ICL TECHNICAL JOURNAL

Volume 5 Issue 4 November 1987

Published by
INTERNATIONAL COMPUTERS LIMITED
at
OXFORD UNIVERSITY PRESS

ICL TECHNICAL JOURNAL

The ICL Technical Journal is published twice a year by International Computers Limited at Oxford University Press.

All correspondence and papers to be considered for publication should be addressed to the Editor.

1987 subscription rates: annual subscription £32 UK, £40 rest of world, US $72 N. America; single issues £17 UK, £22 rest of world, US $38 N. America. Orders with remittances should be sent to the Journals Subscriptions Department, Oxford University Press, Walton Street, Oxford OX2 6DP, UK.

Printed by H Charlesworth & Co Ltd, Huddersfield ISSN 0142–1557

Contents

ICL TECHNICAL JOURNAL
Volume 5 Issue 4

INTERFACE AND PROGRAMMING LANGUAGES

Foreword

This issue of the ICL Technical Journal is dominated by the theme of standards in communications and allied protocols and the key role which they play in creating systems which are able to interwork readily.

The increasing need to be able to create complex systems to meet the needs of users is a major driving force towards an acceptance of the importance of creating standards. These standards must be internationally accepted and in the public domain.

This has led to the rapid acceptance of open systems standards as the means preferred both by industry and by its users to ensure that systems can be built.

The approach is of particular importance to ICL and its customers since we have chosen to focus our energies on supplying systems solutions to serve specific markets. It is not surprising, therefore, that the company has played a leading role in establishing the world-wide acceptance of this design approach. In this, we are following a commitment to standards which dates back to the earliest days of ICL.

It is gratifying to see the rapid way in which these exceedingly complex standards are being generated and are gaining acceptance. They are appearing as mandatory requirements for the I.T. systems of a growing number of major customers. This is one more measure of the increasing maturity of what is still a very young industry, and one in which our systems designers rightly take pride.

D.J. McLauchlan
Director of Technology and Engineering

NETWORKING AND INTERWORKING

Open Distributed Processing

J.B. Brenner
ICL Marketing and Technical Strategy, Bracknell

Abstract

The subject of Open Distributed Processing (ODP) standardisation is at a formative stage in ISO. An analysis of the nature of distributed systems is presented in this context, together with a survey of current research findings.

1 Introduction

The purpose of this paper is to explain some of the technical thinking that underpins the Open Distributed Processing (ODP) standardisation[1] that started this year in ISO, the International Organisation for Standards.

Readers of the ICL Technical Journal will probably already be aware of the existing standardisation for Open System Interconnection (OSI)[2,3]. The ODP standardisation extends far beyond the scope of OSI. It is concerned with how to build distributed systems, and how to integrate software across them. Its focus is to be an ISO Reference Model for Open Distributed Processing, scheduled for completion in 1989.

The European Computer Manufacturers Association (ECMA) has been studying this area for several years and has recommended a set of technical assumptions[4] on which to base the Reference Model. The author is Convener of the ECMA group directly concerned (TC32-TG2); and the content of this paper is closely related to the ECMA work.

The pace of change in information technology is very rapid, and probably nowhere more so than in the field of distributed systems. The impact of some of these changes amounts to paradigm shifts in which the changes overwhelm the basis for existing patterns of thought, and different patterns must take their place. As explained in Kuhn[31], there is great difficulty in recognizing and accommodating to such discontinuities. This is what we face here when trying to visualise Open Distributed Processing in the 1990s.

The subject area has traditionally been viewed mostly in terms of networking and data communications, rather than programming-in-the-large, and languages and compilation systems. This emphasis is now being reversed.

Another big change is that integration of multi-media information (e.g. voice telephone + data) is likely to become commonplace during the lifetime of Open Distributed Processing standards. The integrated handling of real time voice, video and image is a relatively new subject.

A further broadening of horizons is inherent in the enormously large and varied field of application of distributed systems. Table 1 gives an indication of its breadth and diversity.

Table 1 Some fields of application for distributed systems.

Administrative Systems	Radio/TV/Hi-fi
Business Management	Office Systems
Command and Control	Process Control
Engineering Computation	Scientific Computation
Factory Automation	Telecommunications
Image Manipulation	Technical Design
Knowledge Engineering	

Our belief is that across an immense field there is a convergence of techniques, such that it is possible to construct a relatively small core of concepts sufficient to achieve a near universal basis for distributed information systems. The existence of this completeness of distributed systems techniques is substantiated in the "Survey of Techniques" in section 4. This gives a summary of research results and practical experience in distributed systems which ECMA proposes as the technical basis of the standards. The sources which it references are a basic reading list for the whole distributed systems subject area.

But before looking at these research results, we need to explore the nature of the subject area. This is done in two stages. "Understanding the Problem", section 2, seeks to identify the essentials of distributed systems. "Modelling the Problem", section 3, introduces techniques for modelling distributed systems. These are major steps towards construction of the Reference Model for Open Distributed Processing.

Finally, a view of the technical content of the emerging architecture and standards is summarised in the "Expectations", section 5. This emphasises that Open Distributed Processing standardisation will be an evolutionary development, making good use of existing standards.

2 Understanding the Problem

2.1 Introduction

We need to develop an understanding of the fundamental issues particular to distributed systems, so that these can be recognised in the Reference Model for Open Distributed Processing.

A preliminary point is that no special distinction of scope or applicability is intended by calling the ISO work item "Open Distributed *Processing*", not "Open Distributed *Systems*".

2.2 Definitions

The first task is to define what we mean by "distributed systems" in the context of the Reference Model for Open Distributed Processing. We begin with some general definitions.

> **system:** a composite whole.
>
> **component:** a participant in a composite whole.

The components are the **resources** from which a system is composed, and the **elements** into which it may be decomposed. The structure of this **composition** and **decomposition** is the essence of systems. It leads us to the definition of architecture in this context.

> **system architecture:** systematic and formulated knowledge of the composition and decomposition of a system.

These concepts of components and systems are recursive, in that a system may be a component (**subsystem**) of some larger system, which in turn may also be a subsystem of some other system; i.e. systems can be viewed at different degrees of **granularity**. Typically system composition has a hierarchic structure, with complex components which can be decomposed into less complex components, possibly through many levels. This kind of hierarchic composition and decomposition is fundamental to human understanding, and to the management of complexity.

It is possible for something to be a component of more than one system. Furthermore, if something else interacts with a system, it and that system can be viewed as components of some larger system (e.g. system A and an observer or user of it become components of some larger system B).

The above definitions are general to many kinds of systems (e.g. astronomical systems, biological systems, social systems, mechanical systems), so the next step is to concentrate our view onto information systems.

> **information system:** a system which manipulates information (abbreviated here to "system" when the prefix "information" is apparent from the context).

By this definition, the inherent and distinctive characteristics of information systems are **information** and some means of **action** to manipulate it. Similarly, these are inherent characteristics of each component of an information system.

Some further definitions enable us to distinguish between various levels of abstraction.

abstract system: a system defined without reference to realisation.

logical system: a system defined with reference to its realisation, but without reference to its physical realisation.

physical system: a system defined with reference to its physical realisation.

Any realisation of an information system may be viewed at all of these levels of abstraction. A physical system is defined mostly at the boxes and wires level. A logical system is defined in terms of functional units, software modules, interface specifications, protocols, etc. An abstract system is an implementation-independent view of a system. This abstraction is the essence of the system, without extraneous detail. The term "abstract" as used here does not have the connotation "unreal" or "unrealisable".

The next step is to define separation properties with respect to the two distinctive characteristics inherent in information systems, i.e. *information* and *action*. This is where we begin to identify properties that are fundamental to distributed systems.

information separation property: a property of a set of two or more entities, such that any *information* integral to an entity is disjoint from that integral to the other entities.

action separation property: a property of a set of two or more entities, such that any ability for *action* integral to an entity is disjoint from that integral to the other entities.

These separation criteria are used here to define the separation of components within a distributed system, and this gives us precise definitions for distributed information systems.

distributed system: an information system, of which all the components visible at the chosen level of abstraction and granularity have, with respect to each other, the *information separation property* and the *action separation property*.

open distributed system: a distributed system conforming with particular open standards.

The separations which by this definition are axiomatic to distributed systems are illustrated in Fig. 1. The example is a distributed system composed from the separated components A, B, C, and D, with various bindings between them.

The example also illustrates that a component of a distributed system, such as component D, is not necessarily separate from further components, such

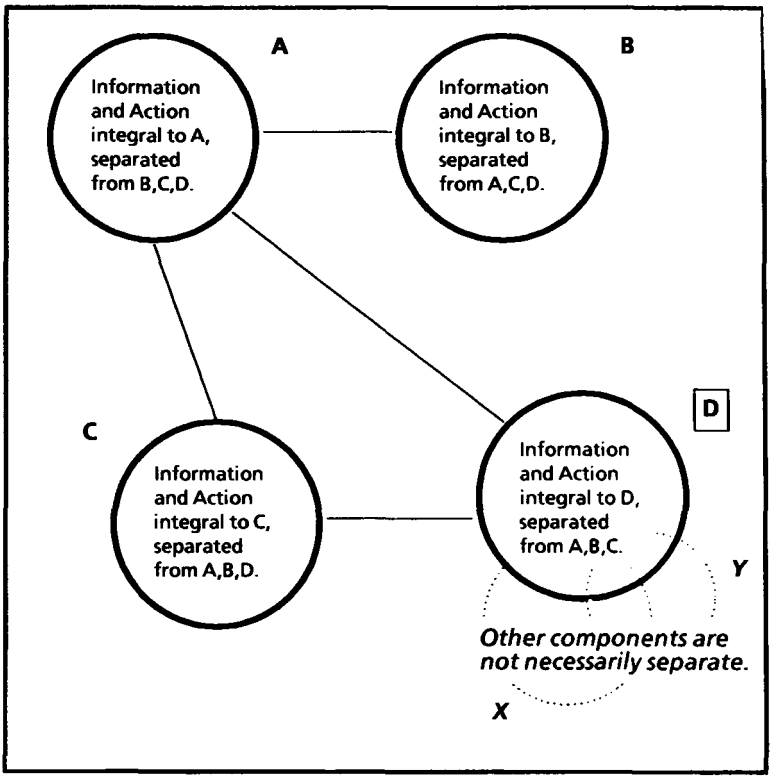


Fig. 1 Axiomatic Separation Properties of a Distributed System

as X and Y, which were deemed not to be visible at the chosen level of granularity. Mutual separation with respect to each other is the necessary and sufficient condition, and global separation from all other components cannot be required as the general case.

Our definition of (open) distributed systems is applicable at all the levels of abstraction; i.e. there can be **abstract distribution** (the disjointness is manifested as separation defined within the specifications), **logical distribution** (the disjointness is manifested as separate implementation modules), and **physical distribution** (the disjointness is manifested as physically separate units). It also includes **temporal distribution**, in which the disjointness is by displacement in time (e.g. interaction is via store and forward mechanisms). Distribution at one level of abstraction does not necessarily imply distribution at others; e.g. a logically distributed system may consist of physically co-located components.

This tight definition of distributed systems is not applicable to all kinds of distributed systems. Some useful and important distributed systems do not have these separation properties. For example, there is no significant

separation of information in the combination of a remote terminal and a computer which echoes onto the terminal screen the characters entered on the terminal keyboard. Similarly, there is incomplete separation of information where software is partitioned into physically separate modules with global variables common to them (distributed global memory). A case where there is no significant separation of action is a remote slave device such as a monitor display. The legitimacy of such configurations is not an issue here, and these examples are all about realisation details not visible at our level of abstraction.

Abstract distribution is the most general case, and is appropriate to an architectural and implementation-independent view of distributed systems. Being at this level of abstraction, and being concerned with the fundamentals of separation, the Reference Model for Open Distributed Processing is essentially about system modularity and structure, not spatial or temporal dispersion.

2.3 Separation

In addition to the **axiomatic separation properties** (i.e. the *information separation property* and the *action separation property*), the components of distributed systems may have other **derived separation properties**. The complete list of separation properties and their characteristics and derivations is given below.

Information separation property. Independent existence of components is inherent in this axiomatic separation property.

Action separation property. Potential for autonomous action by components is inherent in this axiomatic separation property.

Explicit communication property. Explicit communication for interactions between components is inherent in the two axiomatic separation properties, individually and in combination.

Location property. Distinctions of location are inherent in the combination of the two axiomatic separation properties; also locations might change (relocation).

Identity property. Distinctions of component identity (i.e. the ownership, authority and accountability of components) are inherent in the two axiomatic separation properties, individually and in combination.

Isolation property. Potential for isolation of components by control over their accessibility is inherent in the combination of the two axiomatic separation properties. This, together with the identity property, is a basis for security.

Concurrency property. Potential for parallel activity is inherent in the action separation property.

Partial failure property. Potential for a system to continue in operation after the failure of individual components is inherent in the combination of the two axiomatic separation properties.

Incremental change property. Potential to incrementally add, or remove, or replace components is inherent in the combination of the two axiomatic separation properties.

Heterogeneity property. Potential for diversity of implementation is inherent in the combination of the two axiomatic separation properties, and is enhanced by the incremental change property.

From these considerations, two issues emerge. The first is how the consequences of distribution are to be treated; see §2.4 and §2.5. The second is how to exploit the features of distribution to achieve desirable quality attributes; see §2.6.

2.4 Distribution Transparency

A major system and application design issue is whether or not to hide spatial separation and its consequences (e.g. explicit communication and partial failures). The term distribution transparency is used here for discussing the visibility of separation within distributed systems.

Arguments for distribution transparency. It can be advantageous if all the consequences of distribution are made transparent. This hides complexity, simplifies the task of applications designers and enhances the reusability of existing system components. Evolution of existing products based on centralized systems is then inherently straightforward. A successful experiment with such transparency is Unix United[14].

Arguments against distribution transparency. Full distribution transparency, which completely conceals distribution, can be relatively expensive in terms of the underlying implementation effort and performance overheads. Moreover, it denies designers the opportunity to exploit the consequences of distribution via explicit fault management and the decentralization or replication of control, or data, or both.

System design choices lead to different transparency requirements; and full distribution transparency is not always necessary. Therefore, the Reference Model for Open Distributed Processing should structure these transparency choices and not pre-empt them.

2.5 Kinds of distribution transparency

It is well established that distribution transparency is made up of a number of separate elements[41] which are described here in terms of the conditions necessary to achieve full distribution transparency:

Access transparency: Concealing the use of communications when accessing remote resources (such as programs, data and devices).

Location transparency. Enabling the use of a resource, independent of the placement of that resource in the distributed system.

Migration transparency. Enabling the migration or reconfiguration of resources in a distributed system.

Replication transparency. Enabling the use of multiple instances of a resource for such purposes as enhancing dependability and performance.

Concurrency transparency. Avoiding inconsistencies due to parallel execution, by using concurrency control techniques.

Fault transparency. Concealing faults by using error processing techniques.

Performance transparency. Minimizing the performance penalties associated with using remote resources.

Scaling transparency. Concealing variations in system behaviour due to scaling up to large or busy or turbulent systems, and scaling down to small or placid systems.

Transparency issues are demonstrated in the following example, in which a network of small personal computers is used by a group of currency dealers. The dealers need to share access to a simple database of currency prices and deals in progress. The structure of the program in each personal computer is shown in Fig. 2. There are two modules in the program: a database access module that manages access in the shared database, and a currency application module.

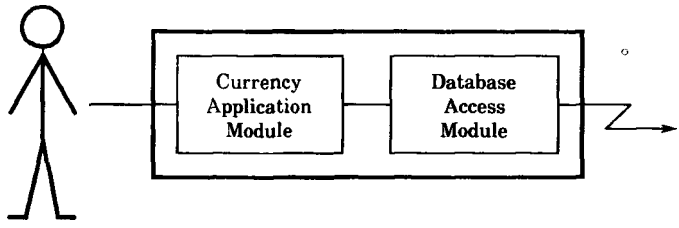


Fig. 2 Program structure of distribution transparency example

A design aim of this system is that the currency application module should not be affected by the distributed nature of the system. This requires that the database access module should provide full distribution transparency (i.e. it completely hides from the application module all distribution of the data-

base). Depending upon how the system as a whole is structured, different strategies will be adopted for the database access module.

Centralized structure. The database is on a central database server which manages concurrent access to the database. The database access module communicates with this remote server, and conceals this use of communications, thereby providing *access transparency.*

Partially distributed structure. The database is on a central 'remote disc' server. The database access modules in each personal computer must now cooperate with one another to preserve consistency of the database. They hold locks, which they set and release in some coordinated way. They conceal this from the currency application module. They thereby provide *concurrency transparency* (which was previously provided for them by the centralised database), in addition to the *access transparency.*

Fully distributed structure. A copy of the database is stored on the local disc of each personal computer. The database access modules in the personal computers each arrange that their copy of the database is kept in step by using, for example, Birman's 'Bulletin Board' protocols[7]. They hide this replication, and thereby provide *replication transparency.* With the database replicated, it is possible for the system to remain in operation despite the failure of individual copies. The database access modules do the necessary error processing to conceal the faults, and thereby provide *fault tranparency.* The database may be too large to store on each local disc, in which case it can be partitioned and each local disc keeps only some fraction of the whole. By handling and concealing this discontinuity of scale, the database access modules provide *scaling transparency.* Responsiveness will be enhanced if the personal computer for each dealer stores locally the database partitions that he uses most frequently. The database access modules thereby minimise the performance penalties associated with using remote resources, and provide *performance transparency.* Responsiveness may be further enhanced if the partitioning can be reconfigured when usage patterns change. The database access modules conceal this, and thereby provide *location transparency* and *migration transparency.*

It is evident that as the degree of distribution increases, the database access module has to provide a greater number of transparency attributes to meet the transparency requirement of the currency application module. Open Distributed Processing standards should specify how to achieve these transparency attributes.

2.6 Quality attributes

The commercial and technical viability of Open Distributed Processing standards will be critically dependent on quality attributes. Systems designers using the standards should be able to achieve high levels for the quality attributes of importance to them, and should be able to make

tradeoffs between quality attributes and other commercially important factors, such as cost.

The following quality attributes are considered to be highly important in this context:

Dependability. Every improvement in dependability (i.e. reliability, availability, maintainability, safety, security)[34] is a potential increase in applicability.

Efficiency. Every improvement in performance with a given resource is a potential increase in applicability and a potential reduction of cost.

Scaling. Every improvement in the ability to scale up and to scale down is a potential increase in applicability and in-service flexibility.

For all quality attributes there are threshold levels to be achieved for viability in particular fields of application; e.g. for process-control applications there are usually critical requirements for response times, continuous operation, stability at peak loads and safety.

Experience abundantly demonstrates that the most demanding aspect of system design is achieving the quality attribute targets. Qualitative factors most therefore have a dominant role in the design of the Reference Model for Open Distributed Processing.

Potential for manipulating quality attributes is inherent in the separation properties and distribution transparencies of distributed systems. For example, reliability and availability may be improved by exploiting the partial failure property and fault transparency and replication transparency; security may be enhanced by exploiting the isolation property; and performance may be improved by exploiting the concurrency property and concurrency transparency. Qualitative excellence is an inherently achievable goal for Open Distributed Processing Standardisation.

2.7 Generic Functions

Some functions are common to all distributed systems, independent of the field of application. They can be classified as **Generic Functionality** and **Generic Attribute Controls**.

The **Generic Functionality** may be classified as follows.

Supportive services. There is necessarily an infrastructure of common supportive services to overcome the obstacles inherent in separation; e.g. directory services, authentication services and time services.

Management mechanisms. There are common management concepts and functions such as domains of control and control points relating to

them. Similarly, there are management functions which should be common to all components; e.g. accounting, security, configuration control, fault management and diagnostic controls.

Data storage mechanisms. There should be generic functions for remote data access, data distribution, data consistency, etc.

Human-computer interfacing mechanisms. There should be common mechanisms for information presentation, dialogue structure, and user-oriented modelling of applications.

Generic Attribute Controls are concerned with how well the distributed system operates. Quality attributes and transparency attributes should be manipulated on the basis of system design policies relating to transparency, efficiency, dependability and scaling. These qualitative control mechanisms should be generic to all components of distributed systems, and should be distinct from, and orthogonal to, the specific functionality of components.

2.8 Summary

This analysis of the nature of distributed systems may be summarised as follows:

All systems are *composed from components.*

Distributed information systems have 10 distinctive *component separation properties*, axiomatic and derived.

The visibility of the separation is defined and manipulated as 8 kinds *distribution transparency.*

The separation properties and distribution transparencies provide opportunities for distributed systems to have *enhanced quality attributes.*

All distributed systems have essentially the same needs for comprehensive *Generic Functionality* and *Generic Attribute Controls.*

With this understanding of what is to be modelled by the Reference Model for Open Distributed Processing, we can now consider how to do the modelling.

3 Modelling the Problem

3.1 General

The main content of the Reference Model for Open Distributed Processing should be an architecture of abstract distributed systems; and this is essentially about separation and its consequences.

The view taken here is that a well formed abstract architecture for this purpose should be constructed from two ingredients:

a **theory** which captures the separation properties that are the essence of distributed systems;

a **framework of abstractions** within which to position and use this theory, and thereby to provide a language for describing and understanding the structure of distributed systems.

Both these ingredients should be consistently and completely applicable across the whole field of abstract distributed systems (and thereby all modular systems). The architecture should have a basis in formal notation that avoids the dangers of ambiguity and inconsistency.

This structure should draw from the theoretical and practical results summarised in the Survey of Techniques in section 4. Some specific proposals are advanced here.

3.2 Object Theory

The concept of "objects"[29] is now generally held to have a crucial role in the structuring of modular systems.

The term **object theory** is used here for discussing object concepts in the context of system modelling, as distinct from use of essentially the same concepts in programming languages[23] and operating systems[12]. Object theory provides a theoretical basis for the structuring of **abstract systems**. It is independent of whether or not object-oriented programming languages or object-oriented operating systems are used in implementations. Object theory objects visible at the level of distributed system granularity may be termed **coarse grained objects**, as distinct from fine grained objects that are only visible in implementation software.

In this object theory, an **object** in an abstract system is any component which is of significance to an observer and contains information and the means of acting upon it. Objects **encapsulate** the internal representation of their information and the implementation of their actions, so that these are not externally visible. An object is therefore a model of a component with the **axiomatic separation properties**, plus crucially important hiding of implementation detail. Object theory is mainly about the modelling of interactions between such separate objects.

Information within an object is typically **persistent**, but may be **volatile**, as required. The observable actions of an object are its **operations**. This is a term taken from **abstract data type** (ADT) theory. Viewed this way, an object is a **data abstraction**, and the operations on it define its behaviour (strictly, its behaviour type). ADT theory and concepts have a formal mathematical basis, and are pervasive throughout high level languages. However, as we will see, object theory consists of more than just data abstraction.

The external visibility of an object is a set of individually named **interfaces** to its operations. An interface provides a **service** consisting of a defined set of operations. Objects interact by sending and receiving **messages** which carry information between them. These are generally termed operation **invocations** and operation **responses**. The content and dialogue structure of these message exchanges must conform to the specification of the interface being used.

Objects offer their services to other objects by **publishing** their interfaces in some appropriate way. Objects gain access to the services provided by other objects by means of the published interfaces of these other objects. The terms **export** and **import** are used for this way of establishing access. Object theory includes **client/server** and **producer/consumer** relationships between interacting objects. There may be many-to-one and many-to-many relationships. These are all aspects of **object bindings**. As a practical matter, client/server models are already coming into general use in OSI standardisation; e.g. in standards for Office Applications[19] and the OSI Directory Service[28].

For some objects many instances may exist. So a distinction is made between object types (sometimes called classes) and object instances. An **object type** is a description of the generic observable characteristics of members of a set of objects with identical behaviour type. An **object instance** is one of a set of objects of the same object type. Object types are not necessarily completely different, and are usually derived from existing object types. Accordingly there are **subtype inheritance** concepts (this inheritance of behaviour description is not to be confused with the inheritance of implementation methods in some object-oriented programming languages).

In addition to their specific interfaces, objects may have a generic **management interface**, consisting of a set of management operations common to all objects. Objects can therefore be **managed objects** which are the component resources of **managed systems**.

Objects may also be characterised by various **object properties** which position them in an external framework of abstractions (see §3.3).

In summary, Object theory is a particular combination of:

> **data abstraction + type concepts + subtype inheritance concepts + object interface concepts + object binding concepts + object properties concepts.**

Object theory can be applied to most aspects of computer systems; e.g. human/system interfacing, system/system interfacing, software/software interfacing, software/hardware interfacing. It is a useful way of structuring because it can prevent the designer's intentions from being distorted by visibility of implementation mechanisms. Moreover, people have an opportunity to understand their information systems without needing to understand the technicalities of the implementation mechanisms.

The primary test of whether Object theory should be at the heart of the Reference Model for Open Distributed Processing is: does it model the separation properties? A secondary test would be: does it have a sufficient formal basis? Another would be: does it fit comfortably into the world of high level languages? Also is it sufficiently general to include important kinds of interactions that are outside the scope of most programming languages today; e.g. voice interactions? The answers are all in the affirmative; therefore object theory is expected to be the fundamental theory used in the Reference Model for Open Distributed Processing.

3.3 Framework of Abstractions

The framework is intended to provide a consistent way of positioning objects in models of information systems. This explicit positioning should help the designer to explain why an object has been introduced into the design of his particular system, and how it relates to other objects. It also helps us to understand and to relate together different systems, because their component objects can be directly compared via their positioning within the one common framework of abstractions.

The framework can be visualised as a multi-dimensional design space. Each object is positioned relative to every dimension (except that for many purposes the position on a dimension will be the null "dont care" value). These are linguistic dimensions rather than geometrical dimensions; the framework is a language for describing the design space, rather than a cartesian grid.

The dimensions chosen should be those inherent in *all* objects. This is an important consideration in the choice of dimensions summarised below.

Topology dimension. This dimension is concerned with the configuration of objects in the model of a system. In this dimension there are zones corresponding to the three levels of abstraction already discussed: physical topology, logical topology and abstract (or service) topology. Configurations may have tree and network mesh topologies, with subdivision into separate or nested domains.

Composition dimension. This dimension is concerned with object composition and decomposition. As the level of composition increases, the objects in a model become more complex but fewer in number. There are recipies to apply structure to a group of objects (i.e. components) and thereby compose a higher level object (i.e. a system or subsystem), and vice versa for decomposition. These recipes include client/server relationships, producer/consumer relationships, and replicated object troupe concepts.

Infrastructure dimension. This dimension is concerned with the organisation of the object-support infrastructure. Most objects are at the far end of this dimension, in that they sit on top of the abstract machine and

object-interpreter environment supported by this infrastructure (the *nucleus* described in §5). At the other end of this dimension are the heterogeneous objects that come from outside the architecture and provide the basic resources (heterogeneous language systems, operating systems, processors, storage, communications networks, etc). Next to these are positioned homogeneous objects which overlay the heterogeneity to provide a homogeneous abstract machine and object-interpreter. Then there are local objects which provide specialised local support for an interpreter; and finally, global objects (e.g. directories) which are not tied to a particular interpreter.

Human interface dimension. This dimension is concerned with the visibility of an object from the viewpoint of human users of systems. It allows a separation of concerns between various aspects of the very complex matters of human interfacing, and between these and all the rest of a system. At one end of this dimension is the human user (modelled as an object); at the other end are resource objects that should know almost nothing of the complexities of human interfacing (e.g. file servers, and the majority of objects). In between are zones in which intermediary objects are concerned with: the interface devices (e.g. keyboards and screens); the presentation of information (e.g. windows, menus, command line parsers, etc.); the dialogue structure along the whole of this human interfacing dimension; and transposing of the underlying resources into cognitive models suitable for the human user (e.g. the desktop paradigm).

Communications dimension. This dimension is concerned with classifying the communications functionality of objects. Most have none (other than that inherent in all object interfaces) and would therefore be at layer 7 of this dimension, the whole of which would be expressed in terms of the OSI layering.

Evolution dimension. This dimension is concerned with controlling changes in the architecture (e.g. version 1, version 1.1, etc.).

Temporal dimension. This dimension is concerned with the life-cycle of a system as it is built up and modified to meet changing requirements.

Each object can be visualised as having a label (a properties list) visible on its surface, declaring its position on these dimensions, and declaring its name(s).

4 Survey of Techniques

4.1 Introduction

Now that we have some idea of the general structure of the architecture, we can populate it with more detailed technical content.

The standards should draw upon known techniques rather than indulge in unnecessary invention. Fortunately, there is a large body of mature tech-

niques that have been developed and explored in the distributed systems research community. Many have yet to appear in standards, but most are already appearing in proprietary forms. These techniques have originated from a considerable experience with system design[33], and are well matched to the technical requirements of Open Distributed Processing.

4.2 Interaction style

Object theory defines interactions as Operations at a level of abstraction that is independent of important issues of notation and implementation.

A notation and interaction style already well established in the world of open standards is that of the **OSI Remote Operations** technique, for which there is now a draft joint CCITT recommendation and ISO standard[25], based on the existing CCITT Recommendation X.410[13]. This is a means for rigorous definition of the structure and syntax of the invocations and responses of operations on remote objects, with automatic derivation of the OSI protocols to support these interactions. The technique is now used for almost all OSI application protocol standards. Another relevant notation for specifying object operations and interfaces is the LOTOS formal description technique[27], which allows formal definition of the semantics and temporal ordering of operations, not just their syntactic structure. These two techniques should be used to complement each other, and not as mutually exclusive alternatives.

An asynchronous **message-passing** style of implementation is generally applicable to most kinds of remote interactions, and has been used in many successful distributed systems designs. **Remote procedure call** is an appropriate implementation technique for synchronous remote interactions, see §4.3.

4.3 Remote procedure call

Much of the research into distributed systems has focussed on bringing software engineering techniques to bear on the problems of building, operating and managing distributed systems.

A major breakthrough came with the maturing of remote procedure call (RPC) techniques, bringing together programming languages and datacommunications via process-to-process communications, as in Birrell and Nelson[9]. In an RPC system a program (the client) can call a procedure, defined at the language level, which is executed by another program (the server), which is potentially remote.

The major feature of RPC is that it allows the programmer to construct application protocols in terms of his normal language construct for interactions between modules, i.e. the procedure call (and function call). RPC also provides flexibility in the configuration of distributed applications, since the

choice of physical co-location or remoteness for sets of procedures need not affect the programmer, and can be deferred.

An RPC package inherently consists of three parts:

an RPC protocol
a runtime library
a program module linker

The RPC protocol is responsible for the transfer of RPC request and response messages between client and server. RPC protocols are optimized for short response times and minimizing the number of packet exchanges. This is in contrast to traditional general purpose protocols which are usually optimized for continuous bulk transfer. RPC protocols mask failures so that, in the absence of catastrophic failure, remote calls are executed exactly once (i.e. like local calls). RPC protocols can be based on simple recoverable connections with ultimate responsibility for recovery vested in the client. RPC protocols may also be built using conventional connection-oriented interconnection.

The runtime library consists of two parts: **marshalling** and **dispatching**. Marshalling is the process of taking the arguments and results of a procedure call and assembling them into packets for transmission by the RPC protocol, and then disassembling them. RPC marshalling is optimized to improve responsiveness by minimizing the complexity and overheads of buffer management. Dispatching is the process of selecting the correct procedure to invoke on receipt of a remote call.

Despatching relies upon a **binding** being set up between client and server. A server program (or support software associated with it) **exports** information about the procedures it offers; and similarly a client **imports** such information. The imported information specifies procedure identifiers to be inserted in requests for decoding by the despatcher.

The job of the program module linker is to automatically replace calls to remote procedures by calls on the local marshalling routines and RPC service, but without explicit programmer involvement. These calls are often realised via what are termed **stub procedures**. The linker also generates the runtime binding information needed for imports and exports, and will exploit the local programming language modularization features to delimit client and server procedures. Complete stub procedures can be generated automatically by software tools systems.

A notable feature of the RPC protocol in Birrell[10] is that it is very well integrated with an authentication and encryption system providing secure communications.

Since procedure calls wait until a result is returned, RPC systems are ideally

based on a lightweight process structure so that many remote procedure calls can be active simultaneously in systems where asynchrony is required.

ECMA has recently completed a remote procedure call protocol standard[20], which uses the OSI Remote Operations notation and protocol.

4.4 Consistency

Distributed processing has led to much research into the problems of **consistency**. In distributed processing there is true parallelism, and the execution of processes in separate machines may overlap. Correctness requires that each process should see a consistent view of the data structures and state of the computation, and therefore the parallelism must be constrained.

To this end, much work has been done on **transaction-based systems** using the concept of transactions from the database world[24]. Transactions are **atomic actions**, their effects are all-or-nothing. In the ARGUS system[36] sequences of program statements, including calls of remote procedures, can be labelled as an atomic action. The ARGUS compiler and runtime system are jointly responsible for providing **stable storage**, and for managing read/write **locks** and running **two-phase commitment** protocols to achieve atomicity. Atomic actions in ARGUS may be nested so that a programmer can build atomic actions around any sequence of statements, including **nested sub-actions**.

Following on from the ARGUS work, other research groups have recognized that greater parallelism can result by using application-oriented locking strategies rather than by automatic nested compositions of read/write locks. An example is the TABS system[46]. Another approach for enhancing parallelism is that of **optimistic concurrency** strategies where processes are allowed to proceed until a conflict is detected and recovery invoked[32].

Another characteristic of distributed programs is the potential to increase dependability and performance by replicating parts of the program. Birman's ISIS system[6] provides an efficient implementation of **resilient objects**. Such an object is implemented as a group of replicated co-operating objects. If a member fails or becomes overloaded, another in the group will take over. The system is based on a suite of optimized **atomic broadcast protocols**[7]. A similar scheme is Cooper's replicated distributed programs[17]. This is an RPC system that supports active replication of both client and server in order to achieve increased dependability.

The treatment of replication in distributed systems is now being made systematic by the recognition that the style of interaction between replicates is the **advisory** style typical of producer-consumer interactions, in comparison to the **imperative** style typical of client-server interactions. There are also well defined concepts, terminology and techniques for **fault-tolerant systems**[34].

4.5 Operating systems

As well as language oriented developments there have been many innovations in operating system technology to accommodate distribution.

The ACCENT system[42] is an example of a network operating system. The inter-process message system of the ACCENT kernel is extended across a network by a 'network process'. To their users, remote services are made to appear as local ports. The network process is responsible for bringing the remote ports into the local address space and for isolating inter-process communication from the details of network communications.

The V-system[15] has followed a different approach. In order to optimize performance, communications are an integral part of its kernel. Because of the efficiency of this, the V-system is able to use the local area network for page swapping between disc-less workstation and file servers. The V-system includes the notion of a 'process group' as a set of processes that can be addressed as a single entity, even if they are distributed across several sites. This notion therefore provides system level support for some of the replication techniques mentioned in §4.4 above.

A number of operating systems have been based on the object-oriented model of computation: the disjoint address spaces of multiple sites matches the encapsulation of state concept that underpins objects[29]. The best known example of this approach to operating systems is Eden[12]. All programs in Eden are objects with well-defined external interfaces. Remote interactions take the form of one object invoking an operation at another, using RPC-like protocols. In Eden objects have **logical addresses** so that they can be accessed without knowledge of their location, enabling dynamic reconfiguration of the system. Eden specifies a number of generic functions that can be applied to any object. These generic functions are mostly to do with unified management of objects.

The Cambridge Distributed System[38] explored the possibility of dynamic instantiation of services upon demand, using a pool of uncommitted processors. Requests for service are directed to a **resource manager** which finds an appropriate free processor, loads it with the required service, and transparently reconnects the user to the newly made service. The operation of this processor pool is dependent upon remote debugging, automated service management and remote access security.

The LOCUS system[49] is an example of how a derivative of Unix can be implemented as a distributed operating system. The LOCUS kernel goes to great lengths to insulate the applications programmer from the effects of distribution. This **distribution transparency** has the great merit that applications previously written for ordinary Unix can run unchanged in the distributed environment, which offers more capacity and dependability than a single node Unix system. The negative aspects of complete distribution

transparency are that applications cannot exploit the distribution they cannot see, and that the needs of system management (particularly failure diagnosis and reconfiguration) are in conflict with the transparency.

There has been much debate on what kinds of distribution transparency should or should not be provided in distributed systems (see §2.4 and §2.5). This debate reinforces the recommendation in §2.4 that standards be flexible enough to support variable distribution transparency.

4.6 Protocols

Many aspects of protocol design have been revisited using systems engineering techniques rather than traditional communications engineering. The application of the 'end-to-end' principle[44] has led to a focus on reducing the processing and buffer management overheads at network nodes. The outcome has been a move away from strict layering of protocol implementations, with simplification of protocols so that they can be moved out of general purpose processors into network interface units. This simplifity also enables small machines, such as personal computers, to support complete implementations of the protocols.

At the present time attention is being given to the requirements of very high bandwidth networks (e.g. 100Mbits/sec LANs), and high bandwidth combined with long delay (e.g. satellite channels)[35]. In these networks, many of the assumptions that are fundamental to traditional protocols are being undermined. Fast networks can bombard a node with data faster than the data can be processed. By the time the situation is recognized and the processor reacts, the amount of data in transit may be immense, leading to severe buffering problems (especially at intermediate gateways), with consequent instability in congestion control algorithms. Preliminary work suggests that **rate controlled** protocols, which inherently avoid overcommitment of network nodes and consequent feedback oscillations, will be more stable and achieve better throughput[16].

4.7 Multi-media interactions

Communication can be classified as **isochronous** (for real-time voice or video, etc.) and **anisochronous** (for text, data, stored graphics and stored image, etc.). Several systems have been built that provide for interactions in which all these forms of information are handled in an integrated fashion, for example multi-media conferencing applications[21,5].

The integration of isochronous and anisochronous interactions is stressful for communications and processing. Progress in this area is predicated upon real-time performance guarantees from networks, processors and operating systems, and consequently many of the performance optimisation techniques described above will be essential.

Multi-media interactions can be accommodated within object theory. The fundamentals of object theory (e.g. encapsulation, the modelling of interfaces, and the way in which objects form relationships via the publishing of interfaces) are invariant to such details.

4.8 Heterogeneity

There has been considerable attention to the problems of accommodating heterogeneous systems on a network. An important theme has been common services to provide links between islands of homogeneity. For example there have been many designs of system-independent file servers[39,47] and comprehensive work on directory services and authentication[11]. The latter is based on practical experience with very large distributed systems.

Important experience has been obtained in software tools to support RPC between programs in different languages, executing in different kinds of computers, and communicating across different networks[30,22].

4.9 Security

Most attention in the area of security has been directed towards the use of encryption for communications security. A substantial survey is given by Voydock and Kent[48].

As well as being a means of achieving data integrity, encryption has been used as a means of 'sealing' data in authentication protocols[37].

Encryption has been made an integral part of RPC protocols to defend them against a wide range of network level attacks[10]. The design and operation of authentication services[11] has been explored in some detail. Work has also been done on the exploitation of separation in distributed systems to achieve isolation and enforcement of security policies[43].

4.10 Large systems

The research community has not confined itself to laboratory-sized systems. Many of the research systems have grown to considerable size and operate in service on a production basis. An example of this scaling up of research experiments is the Grapevine mail system on the Xerox Internet[8,45], which provided many lessons for future systems. Several research projects have been based on extensive wide area networks, including the ARPA network in the USA and satellite systems[35].

A number of academic institutions are engaged in setting up large distributed computer workstation networks as an integral part of their infrastructure to support teaching and research; e.g. Project ATHENA at MIT and the Information Technology Center at Carnegie-Mellon University[40].

5. Expectations

Given the above technical assumptions for Open Distributed Processing standardisation, we can now speculate about the technical content of future Open Distributed Processing standardisation.

The likely abstract architectural structure of the Reference Model for Open Distributed Processing is illustrated in Fig. 3. The components of systems are modelled and encapsulated as **objects** (a), for which there are concepts and formalisms for object derivation, specification, composition, decomposition and inter-relationships. Each object is positioned in a universal multi-dimensional **framework of abstractions** (b), which defines object properties, both absolutely and relative to all other objects. The objects interact via some kind of object-support **nucleus** (c).

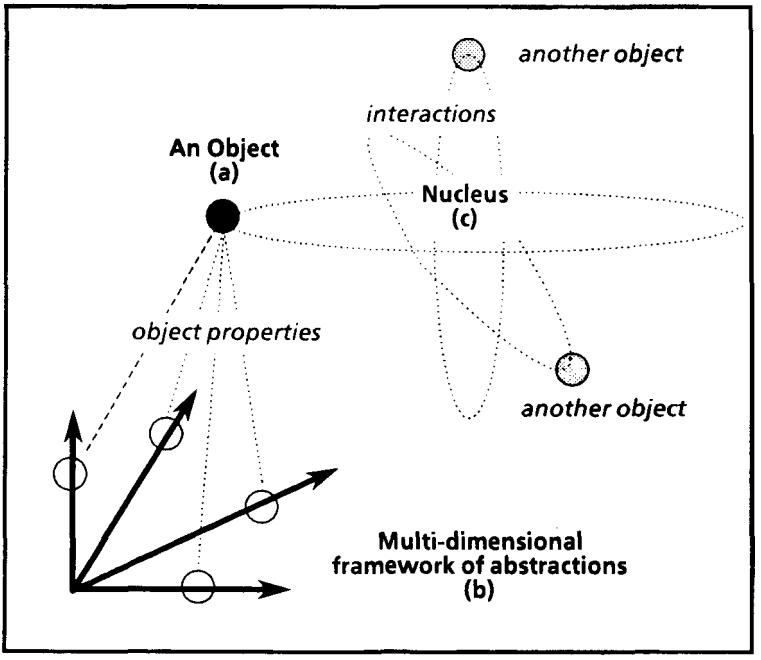


Fig. 3 Open Distributed Processing abstract architecture

The nucleus is the sole means for an object to publish its interfaces, names and properties, and to find out about those of other objects, and to access other objects. It also provides the means of manipulating the distribution transparencies and quality attributes of interactions, and is oriented towards object implementation in high level languages.

A vital constraint on the nucleus is that it should evolve from existing open

standards, and be able to co-exist with proprietary standards. The concepts and protocols in the nucleus should be oriented towards high level programming languages, and be network-independent, operating system-independent, and language-independent. They would thereby have the potential to co-exist and interwork with other kinds of networking. A key to achieving this is seen to be the OSI Remote Operations technique[25]. By an interesting combintion of forsight and good fortune, it is directly on the evolutionary path into the new kind of language-oriented integration of distributed systems.

In ECMA this nucleus has been called DASE, the Distributed Applications Support Environment[18]. An ECMA DASE standard is scheduled for completion in 1988.

Acknowledgements

Most of the ideas presented in this paper have been refined and articulated via the Advanced Network System Architecture project, Alvey ANSA[26], and in particular Andrew Herbert.

Clarification of the technical approach owes much to the expertise and the patient committee work of my ECMA colleagues over several years, especially David Robinson of GEC Marconi.

I would also like to thank Graham Crisp of Plessey for his exposition to ANSA on Object Theory, which I have drawn on here.

References

1 ISO.; "Proposed revised text for the NWI on the Basic Reference Model for Open Distributed Processing". ISO/TC97/SC/21 N1889. April 1987.
2 HOULDSWORTH, J.; "Standards for Open-Network operation", ICL Tech. J., November 1978.
3 BRENNER, J.B.; "IPA networking architecture", ICL Tech. J., 1983, **3** (3), 234–349.
4 "Proposed Technical Assumptions for Open Distributed Processing". ECMA, Geneva, April 1987.
5 AGUILAR, G., GARCIA-LUNA-ACEVES, J., MORAN, D., CRAIGHILL, E. and BRUNGARDT, R., "Architecture for a Multimedia Teleconferencing System", ACM Computer Communications Review, **16** (3), 126–136 (August 1986).
6 BIRMAN, K., "Replication and Fault Tolerance in the ISIS System", ACM Operating Systems Review, **19** (5), 79–86, (December 1985).
7 BIRMAN, K., JOSEPH, T. and STEPHENSON, P., "Programming with Shared Bulletin Boards in Asynchronous Distributed Systems", Technical Report TR86-776, Department of Computer Science, Cornell University (August 1986).
8 BIRRELL, A., LEVIN, R., NEEDHAM, R. and SCHROEDER, M., "Grapevine: an Exercise in Distributed Computing", Communications of the ACM, **25** (4), 260–274, (1982).
9 BIRRELL, A. and NELSON, B., "Implementing Remote Procedure Calls", ACM Transactions on Computer Systems, **2** (1), 39–59 (February 1984).
10 BIRRELL, A., "Secure Communications Using Remote Procedure Calls", ACM Transactions on Computer Systems, **3** (1), 1–14 (February 1985).
11 BIRRELL, A., LAMPSON, B., NEEDHAM, R. and SCHROEDER, M., "A Global

Authentication Service Without Global Trust", Proceedings IEEE Security and Privacy Conference, Oakland, California, USA, 223–230 (1986).
12 BLACK, A., "Supporting Distributed Applications", ACM Operating Systems Review, **19** (5), 181–193 (December 1985).
13 CCITT Red Book, Volume VIII, Fascicle VIII.7, Data Communications Networks, Message Handling Systems, Recommendation X.410, CCITT, Geneva, 1985.
14 BROWNBRIDGE, D.R., MARSHALL, L.F. and RANDELL, B., "The Newcastle Connection or UNIXes of the World Unite!", Software – Practice and Experience, **12** (12), 1147–1162 (December 1982).
15 CHERITON, D., "The V Kernel: A Software Base for Distributed Systems", IEEE Software, **1** (2), 19–42 (April 1984).
16 CLARK, D., Presentation on "Rate Controlled Protocols", ARPA Internet End-to-end Task Group Protocols Workshop, University College, London, (August 1986). Unpublished.
17 COOPER, E., "Replicated Distributed Programs", ACM Operating Systems Review, **19** (5), 63–78 (December 1985).
18 "Distiributed Applications Support Environment (DASE)". ECMA/TC32/TG2/86/91, ECMA, Geneva, December 1986.
19 "Framework for Distributed Office Applications". ECMA, Geneva, June 1987.
20 "Basic Remote Procedure Call (RPC) Protocol using OSI Remote Operations". ECMA, Geneva, December 1986.
21 FORSDICK, H., "Explorations into Real-time Multimedia Conferencing", Proceedings of IFIP TC 6 International Symposium on Computer Message Systems, Washington, D.C., USA, 331–347 (September 1983).
22 GIBBONS, P.H., "A Stub Generator for Multilanguage RPC in Heterogeneous Environments". IEE Transactions on Software Engineering, Vol. SE-13, No. 1, Jan. 1987.
23 GOLDBERG, A. and ROBSON, D., "Smalltalk-80. The Language and its Implementation". Addison Wesley, 1983. ISBN 0-201-11371-6.
24 GRAY, J., "Notes on Database Operating Systems", in Bayer, R., Graham, R.M. and Seegmüller, G., (eds.), "Operating Systems: An Advanced Course", Springer-Verlag, 1979.
25 ISO 7092. "Draft International Standard: Information Processing Systems–Text Processing–Remote Operations Part 1: Model, Notation and Service Definition"; "... Part 2: Protocol Specification". ISO, Geneva, March 1987.
26 HERBERT, A.J., "The Advanced Network Systems Architecture Project", ICL Tech. J., November 1987.
27 ISO 8807. "Information Processing–Open Systems Interconnection–LOTOS–A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour", DP8807, ISO, Geneva.
28 ISO DP 9594. "Directory Standard.". ISO, Geneva, September 1986.
29 JONES, A., "The Object Model: A Conceptual Tool for Structuring Software", in Bayer, R., Graham, R.M. and Seegmüller, G., (eds.), "Operating Systems: An Advanced Course", Springer-Verlag, 1979.
30 JONES, M.B., RASCHID, R.F., THOMPSON, M.R., "Matchmaker: An Interface Specification Language for Distributed Processing". Proceedings 12th. ACM Symposium on Principles of Programming Languages, Jan. 1985.
31 KUHN, T., "Structure of Scientific Revolutions" (2nd edition), University of Chicago Press, 1970.
32 KUNG, T. and ROBINSON, J., "On Optimistic Methods for Concurrency Control", ACM Transactions on Database Systems, **6**, 213–266 (June 1981).
33 LAMPSON, B., "Hints for Computer System Design". ACM Operating Systems Review, **17** (5), 33–48 (October 1983).
34 LAPRIE, J., "Dependable Computing and Fault Tolerance: Concepts and Terminology", Proceedings 15th Annual International Symposium on Fault Tolerant Computing, Ann Arbor, Michigan, USA, 2–11 (June 1985).
35 LESLIE, I., NEEDHAM, R., BURREN, J., COOPER, C. and ADAMS, C., "The Architecture of the Universe Network", ACM Computer Communication Review, **14** (2), (June 1984).
36 LISKOV, B. and SCHEIFLER, R., "Guardians and Actions: Linguistic Support for

Robust Distributed Programs", ACM Transactions on Programming Languages and Systems, **5** (3), 381–404 (July 1983).
37 NEEDHAM, R. and SCHROEDER, M., "Using Encryption for Authentication in Large Networks of Computers", Communications of the ACM, **21** (12), 993–999 (December 1978).
38 NEEDHAM, R.M. and HERBERT, A.J., "The Cambridge Distributed System", Addison-Wesley (1982).
39 MITCHELL, J. and DION, J., "A Comparison of Two Network-based File Servers", Communications of the ACM, **25** (4), 233–245 (December 1982).
40 MORRIS, J., SATYANARAYANAN, M., CONNER, M., HOWARD, J., ROSENTHAL, D. and DONELSON-SMITH, F., "ANDREW: A Distributed Personal Computing Environment", Communications of the ACM, **29** (3), 184–201 (March 1986).
41 POPEK, G., WALKER, B., CHOW, J., EDWARDS, D., KLINE, C., RUDISIN, G. and THIEL, G., "LOCUS: A Network Transparent, High Reliability Distributed System", ACM Operating Systems Review, **15** (5), 169–177 (December 1981).
42 RASHID, R. and ROBERTSON, G., "Accent: A Communications Oriented Network Operating System Kernel", ACM Operating Systems Review, **15** (5), 64–75 (December 1981).
43 RUSHBY, J. and RANDELL, B., "A Distributed Secure System", IEEE Computer, **16** (7), 55–67 (July 1983).
44 SALTZER, J., REED, D. and CLARK, D., "End-to-End Arguments in System Design", ACM Transactions on Computer Systems, **2** (4), 277–288 (1984).
45 SCHROEDER, M., BIRRELL, A. and NEEDHAM, R., "Experience with Grapevine: the Growth of a Distributed System", ACM Transactions on Computer Systems, **2** (1), 3–21, (February 1984).
46 SPECTOR, A.Z., BUTCHER, J., DANIELS, D.S., DUCHAMP, D.J., EPPINGER, J.L., FINEMAN, C.E., ABDELSALAM, H. and SCHWARZ, P.M., "Support for Distributed Transactions in the TABS Prototype", IEEE Transactions on Software Engineering, **SE-11** (6), 520–530 (June 1985).
47 SVOBODOVA, L., "File Servers for Network-based Distributed Systems", ACM Computing Surveys, **16** (4), 353–398 (December 1984).
48 VOYDOCK, V. and KENT, S., "Security Mechanisms in High-level Network Protocols", ACM Computing Surveys, **15** (2), 135–171 (December 1978).
49 WALKER, B., POPEK, G., ENGLISH, R., KLINE, C. and THIEL, G., "The LOCUS Distributed Operating System", ACM Operating Systems Review, **17** (5), 49–70 (October 1983).

The Advanced Networked Systems Architecture Project

Andrew Herbert
ANSA, 24 Hills Road, Cambridge CB2 1JP

Abstract

The Advanced Networked Systems Architecture Project is an Alvey project involving STC/ICL, BT, Digital, GEC, Hewlett Packard, Information Technology plc, Olivetti, Plessey and Racal.

The purpose of the project is to produce standards for the next generation of distributed applications for digital networks, exploiting advanced results from the fields of systems architecture, computer and networked systems research and modern networking technology.

1 Introduction

The goal of the ANSA project is to lead the IT industry towards establishing definitive international standards for advanced networked systems by the 1990s.

ANSA is concerned with networks that support distributed processing applications. These are applications in which discrete components of the overall application may be located in more than one system, at more than one geographical location, or where there is any reason which necessitates explicit communication among the components.

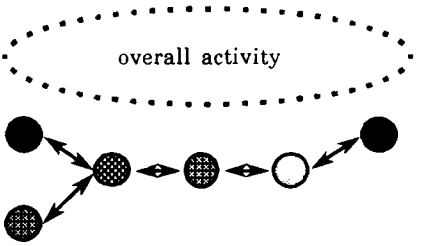


Fig. 1 Distributed processing

The important feature of this definition is the focus on systems supporting some overall application, such as the automation of an office or factory. This

distinguishes distributed processing from the less intimate networking style of open systems interconnection (OSI) between autonomous systems with separate, individual objectives.

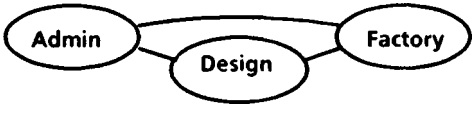
Distributed processing has a very wide field of application including:

administration systems
business management systems
command and control systems
factory automation systems
image manipulation systems
radio/tv/hi-fi distribution systems
office systems
process control systems
telecommunications systems
scientific computation systems

2 Integration through distributed processing

Distributed processing is important not only within each of these fields of application, but also as a bridge between them. Many organizations already have information systems that use distributed processing to support a variety of internal functions. For example, a manufacturing company may have a distributed word processing system in the administration department, a network of CAD workstations in the design shop and an automated factory floor. There are organizational advantages to be gained from the integration of these separate functions into a single, large scale, distributed system oriented towards achieving maximum effectiveness for the company.

- linking together independent systems in an organization



- linking together autonomous organizations

Fig. 2 Integration through distributed processing

Integration is necessary between separate organizations as well as within a single organization. Electronic trading, for example, leads to a distributed system spanning traders, customers and banks. This level of integration is much harder, since no single authority can control the entire activity of the

system. Instead ways must be found to maximize the ability of independent authorities to interwork, without jeopardizing their concerns and interests.

These forms of integration will inevitably lead to heterogeneous systems containing a wide variety of computer and networking technologies, supplied by a multiplicity of IT vendors. The reasons for this are several:

- Many distributed applications are highly specialized and require different environments, with conflicting constraints in terms of such factors as real-time response, throughput, security, reliability and so forth. For example, a safety critical application in a factory may require fault-tolerant and replicated hardware, whereas in an office environment continuous availability is less of a concern.
- Distributed systems grow and evolve over time. New computers and services are connected, new applications are installed. All of these are required to coexist and interwork with older equipment. It is very common to discover distributed systems continuing to use an obsolete item of equipment because it supports a key application which cannot easily be moved to newer components. The owners of distributed systems will come to expect gradual (and continuous) evolution of their systems, rather than the periodic total replacement approach of the past.
- It is unreasonable to assume that any single vendor will be able to offer a complete solution to all distributed application needs – in addition to the major systems vendors there will always be specialist vendors in niche markers (such as CAD) and start-up vendors selling innovative system components.

3 Homogeneity through standards

It has been accepted for some time that the solution to the problem of heterogeneity lies in the use of open standards, such as OSI. Open standards are agreed in a public forum and represent an industry consensus. Open standards help the customer by giving him access to a greater number of vendors and help the vendor by enabling him to sell competitively to the entire information technology marketplace.

Standards are about agreeing interfaces between system components to achieve compatibility. This begs several questions. Which system functions are candidates for standardization? What sort of compatibility is desired? At what level of abstraction should an interface be defined?

Figure 3 shows some possible answers to these questions.

- Human-computer interface standards prescribe how applications should appear on the user's terminal and how the user should interact with the computer. These sorts of standards can bring great uniformity to a wide range of quite distinct applications and reduce the degree of user training necessary before a new application can be used. This is illustrated by the

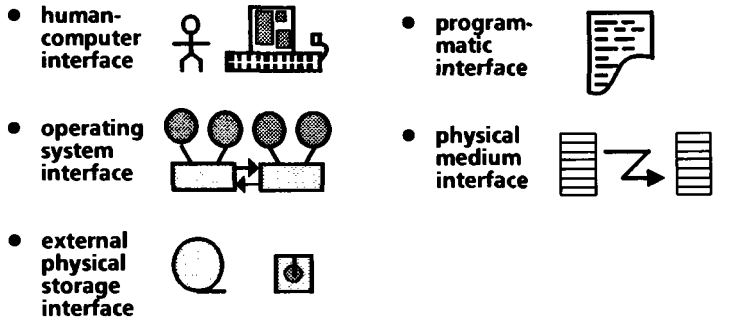


Fig. 3 Kinds of interface abstractions

success of window, icon and menu based systems such as the Apple Macintosh.

- Programmatic standards specify the languages in which applications should be written and the subroutine libraries to be provided. These standards provide application portability across a wide range of systems.
- Operating system standards provide a common application environment by defining application services such as I/O, filing and database. Operating system standards enable interworking between applications written in diverse languages. This level of standardization is very powerful, which explains the thrust behind initiatives like X/Open.
- OSI provides standards for a physical communications interface so that different systems may interchange data and support common services such as transaction processing. It thus enables interconnection of heterogenous computer systems via digital communications networks.
- Standards for the organization and encoding of data on external physical storage such as discs and tapes are another example of an information interchange interface, but in this case not dependent upon a communications link between systems.

Thus it can be seen that all of these different levels of abstraction have their merits, since each achieves a different sort of compatibility. The system designer seeks to achieve a balance between all these levels so that the systems he designs offer the greatest compatibility for the customer. The designer therefore needs to know the relationships and dependencies between interfaces at different levels of abstraction and how to build system components that support standard interfaces.

The sorts of consistency requirements that face the system designer are shown in Fig. 4.

- At the top level the designer needs to be assured that his design does meet the overall system objective and that he has proposed the most cost-effective solution.
- When assigning interfaces for a function, the designer is concerned to

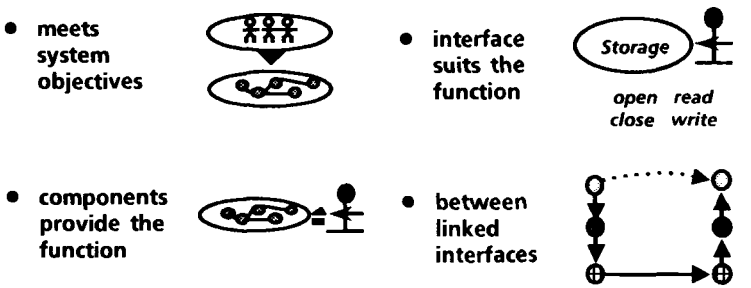


Fig. 4 Consistency requirements

choose an appropriate interface. For example, an information storage function could be met either by a filing system or by a database, both of which have rather different characteristics.

- The designer needs to select an appropriate combination of components to achieve a desired interface. He is aided if there is a repertoire of system building blocks and an understanding of how basic building blocks can be combined to meet higher level requirements.
- Finally, and perhaps the most difficult of all, the designer wants to be sure of compatibility for linked interfaces – for example that a programming language library is compatible with an operating system's interface, or that an operating system's interprocess communication system can be extended over a network to link processes in separate machines.

4 Architecture

The solution to these problems of design and consistency is to present the designer with an architecture to help him in his task.

The purpose of an architecture is to provide a common basis for design and to impose a common style on all systems derived from the architecture.

The benefits of an architecture are the family resemblance of the systems derived from it and confidence in the consistency of those systems.

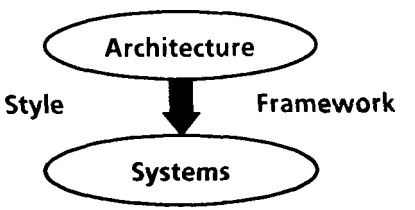


Fig. 5 Architecture

Figure 6 illustrates the major elements of an architecture for distributed processing.

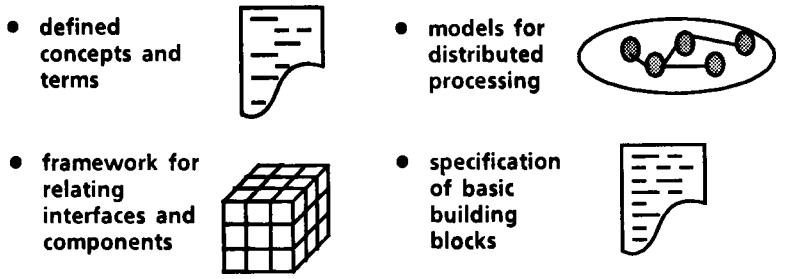


Fig. 6 Architectural elements

- The designer requires a base of well-defined concepts and terms within which he can describe and explain his system.
- To support the description and design of systems he needs formal models of how system components can be related to one another and from which the behaviour of a system predicted.
- A framework of levels of abstraction will guide the designer by positioning system building blocks and interfaces relative to one another. (Possibly the best known example of such a framework is the OSI Reference Model which positions various communications functions into a hierarchical series of layers.)
- Finally, the designer needs specifications for the basic building blocks out of which he can build practical systems.

Distributed processing architectures can be divided into two kinds. There are many which are specific to a particular field of application. For example, a distributed office architecture may be concerned with workstations, file servers and print servers. The concern of ANSA is with the generic aspects of distribution that are common across all fields of application. This is shown diagrammatically in Fig. 7.

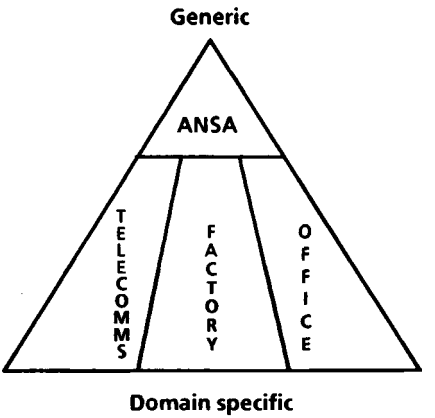


Fig. 7 Kinds of architectures

The benefit of a generic architecture is that it brings the domain specific architectures closer together by providing a common shared framework and common foundation of basic building blocks.

It is important not to confuse an architecture with a system. A **system** is a single solution to a single problem. An **architecture** is an orderly package of solutions to a range of problems. It is naive to imagine that a single system could meet all application requirements. Indeed, this has already been given as an argument in favour of heterogeneity in distributed systems. It is, however, plausible to consider a generic architecture spanning many fields of application provided there is some criterion for deciding what is a generic function.

5 Transparency

The key to understanding generic functions for distributed processing is the concept of distribution transparency.

A system is described as having distribution transparency if it conceals the consequences of distribution from applications and users – that is to say, the users perceive the system to be a single whole rather than merely a collection of independent resources.

The sources of generic functions associated with distribution transparency are best illustrated by examples.

Suppose a hospital wishes to provide hospital staff (doctors, nurses, administrators, technicians etc.) with online access to hospital and patient records. A distributed solution to this information processing problem is illustrated in Fig. 8. Every user has been given access to a personal computer and because of distribution transparency the users are given the impression that there is a single "logical" database common to all the machines, even though in actuality the data is spread out over all the local discs.

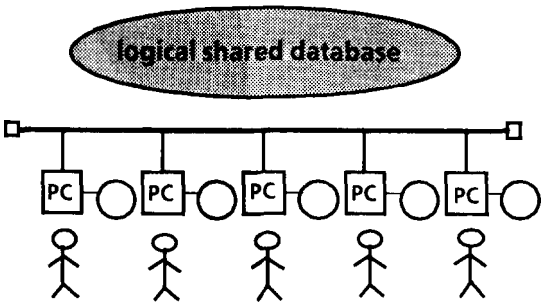


Fig. 8 Distributed processing

The first form of distribution transparency evident in this diagram is access transparency: the users see a single logical database; there is no distinction between local and remote data, although access may be restricted by access controls in support of some suitable security policy.

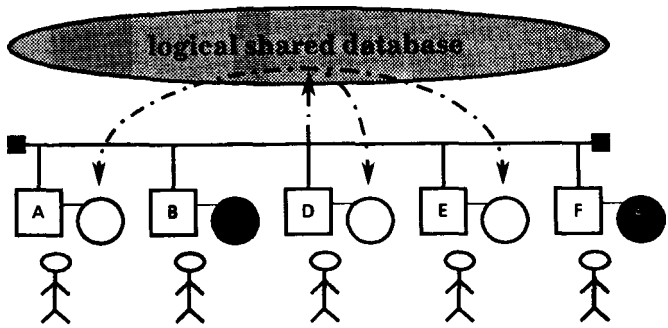


Fig. 9 Access transparency

The next transparency is that of location – the users do not have to know which parts of the database are stored on which machines. Instead, they identify data by logical or functional names and it is left to the system to translate these names to actual database record addresses.

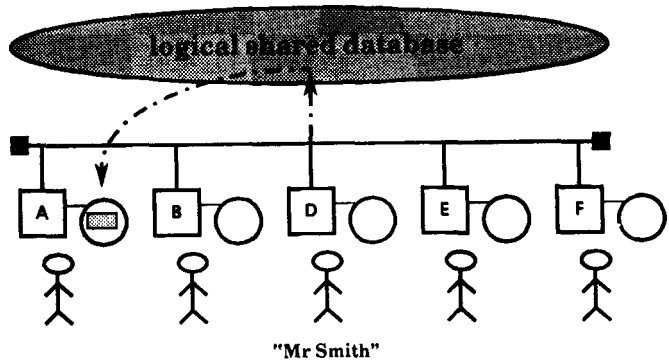


Fig. 10 Location transparency

Since there are potentially many users of the database active at once, the logical database manager must coordinate the parallel activity among the users and ensure that a consistent database image is presented at all times.

In a distributed system there is an opportunity to keep redundant copies of the material in the database and to provide surplus processing resources. This replication may be active in the sense that all the replicates are operating simultaneously to provide increased availability, or voting for

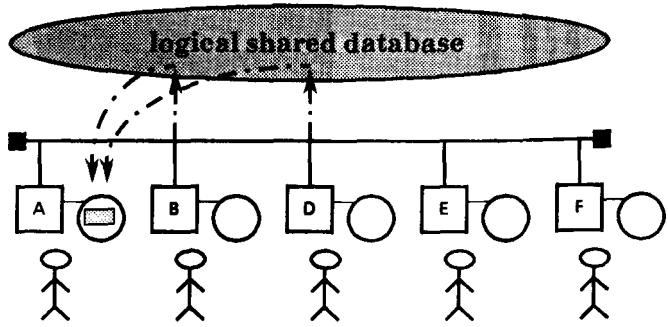


Fig. 11 Concurrency transparency

increased confidence in the results. Alternatively the replication may be passive in the sense that only one replicate is in service and the others are in "standby mode".

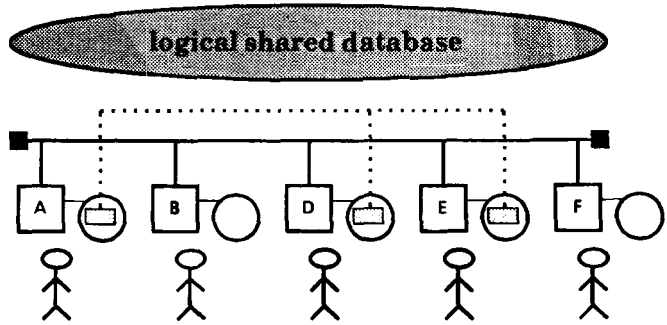


Fig. 12 Replication transparency

This concept leads naturally into failure transparency. One of the applications of replication is to provide continuous service, or to enable recovery and switchover to an alternate.

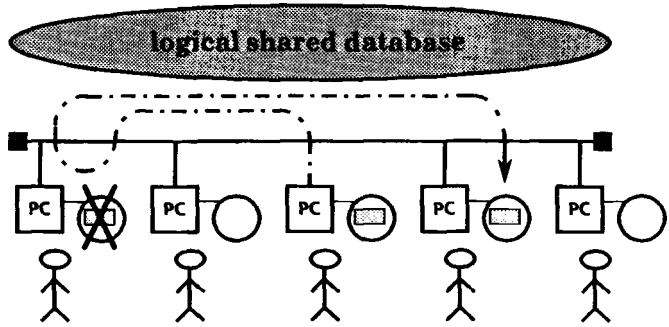


Fig. 13 Failure transparency

Related to replication transparency is migration transparency in which it is possible to relocate parts of the database to adapt to changing circumstances, or to move a service dynamically from one machine to another.

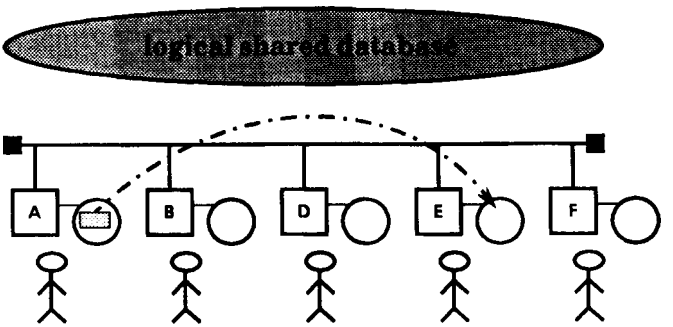


Fig. 14 Migration transparency

By combining these various sorts of transparency it is possible to achieve a significant degree of performance transparency – that is to say, a distributed system that appears to be as powerful to the user as if he had a dedicated machine working just for him. For example, the parts of the logical database most heavily accessed by a user can be kept on his local disc to optimize access time and response by exploiting migration transparency. Consistency processing can often be a background asynchronous task if appropriate algorithms are chosen.

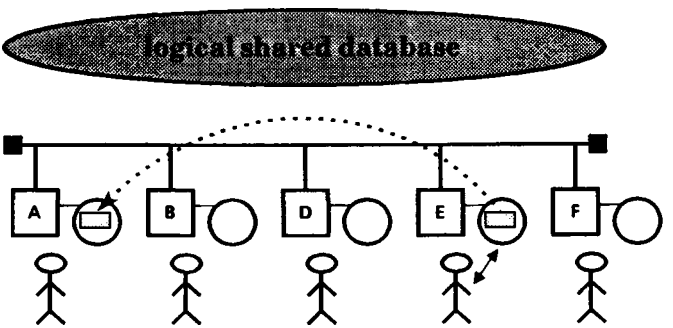


Fig. 15 Performance transparency

Finally, well-designed distributed systems have scaling transparency, that is to say the system can be expanded or contracted arbitrarily without change to its structures and algorithms.

From an analysis of distribution transparency it is possible to draw up the following list of generic functions for distributed systems:

access transparency
- communications and security mechanisms

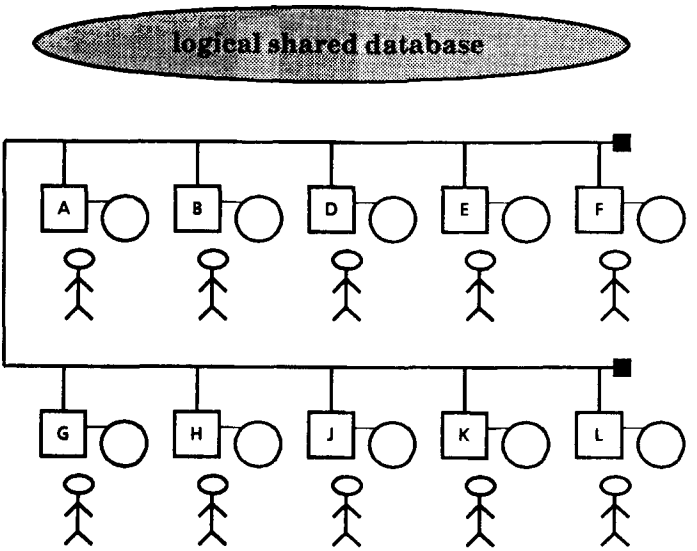


Fig. 16 Scaling transparency

location transparency
– naming, addressing and routing
concurrency transparency
– synchronization and event ordering mechanisms
replication transparency
– active and passive replication mechanisms
failure transparency
– fault management and recovery mechanisms
migration transparency
– configuration and dynamic configuration mechanisms

Performance and scaling transparencies do not lead to distribution mechanisms *per se*, but rather towards strategies for global system optimization.

In different fields of application, different combinations and degrees of distribution transparency will be appropriate and at different levels of abstraction, leading to different combinations of mechanisms or basic building blocks.

Thus the job of a distributed processing architecture such as ANSA is to provide a model and framework for using these building blocks (i.e. transparency mechanisms) at all levels of abstraction.

6 The ANSA culture

The illustration in Fig. 17 shows how the ANSA project is approaching design of the architecture.

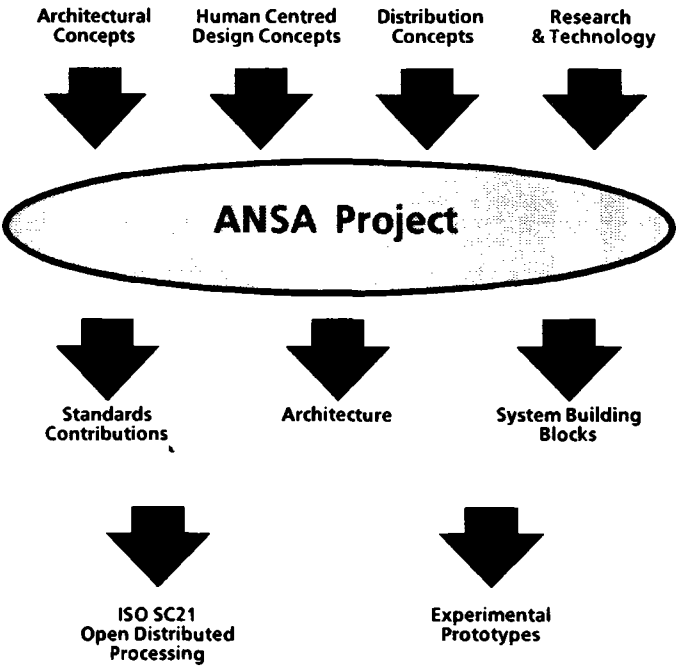


Fig. 17 The ANSA culture

The outputs of the ANSA project are the architecture itself in the form of a published document – the *ANSA Reference Manual* – contributions to the standards process and also experimental prototypes of the mechanisms or building blocks necessary to support the implementation of practical distributed systems.

The standards target for ANSA is the "Open Distributed Processing" New Work Item established by ISOXTC97 SC21 WG1. This has scope and direction similar to ANSA and is the logical progression within open systems standardization from open systems interconnection (OSI).

The initial target for the experimental prototypes is C and Unix since this is an important and widely available environment, enabling the sharing of experience and transfer of technology between the ANSA project and the sponsoring companies.

The inputs to the ANSA project – perhaps best styled as "the ANSA culture" – include the architectural and distribution concepts explained in earlier sections.

A further important thread to ANSA is human centred design. All systems exist to support a human enterprise and to satisfy human concerns. By understanding the needs of enterprises and the concerns of people who use

distributed systems it is possible to understand that many of the variations between otherwise similar mechanisms exist because of different perceptions of the human concerns involved. It is therefore important that ANSA should include these factors in its framework to help the designer select the appropriate mechanism to meet a particular requirement.

An architecture is crucially dependent upon assumptions about technology and its development. The technological assumptions adopted by ANSA are those in Brenner's paper "Open Distributed Processing" pp 613–637. Included among these assumptions is the observation that recent research work in the field of distributed operating systems and communications places many significant and useful results in the hands of the system designer.

7 The ANSA Reference Manual

The *ANSA Reference Manual* is organized as a seven part document as follows:

Part I: Overview

Part II: Technical background
a summary of the concepts that underpin the design of ANSA; essentially a greatly expanded version of this paper

Part III: Concepts and definitions
a complete set of concepts and definitions for distributed system terms; almost an encyclopaedia of distributed processing

Part IV: Models and framework
a formal definition of the models and framework used in ANSA

Part V: System building blocks – description
descriptions and implementation guides to the ANSA distributed system building blocks

Part VI: System building blocks – specifications
detailed formal specifications of the system building blocks described in Part V

Part VII: Examples
illustrated examples of systems building using ANSA, based upon the building blocks of Parts V and VI and the experimental prototypes constructed during the course of the project.

8 Project profile

The ANSA Project is an Alvey project supported by British Telecom, Digital Equipment Corporation, GEC, Hewlett-Packard, Information Technology plc, Plessey, Racal and STC/ICL.

The project team consists of approximately fifteen staff (some consultants, some seconded, others directly hired) working at a single project office in Cambridge, England. This central structure is quite unique among Alvey and

ESPRIT projects and has worked very well for the development of ANSA and the establishment of cohesion among the project team.

The project is funded at approximately £1 million per year through to mid 1989.

The approximate project timescales are shown in Table 1.

Table 1 Timescales

Date	Timescale
June '87	Manual Part 1 to 4 to be written
December '87	Manual reviewed and accepted by collaborators
March '88	Prototype test bench environment available
December '88	ANSA in active use by collaborators
March '89	Major prototype demonstrations working on the test bench
June '89	Manual Part 1 to 5 complete
June '89	ISO ODP Reference Model to first DP
June '91	ISO ODP Reference Model Standard

9 Further information

Further details about ANSA, including copies of the *ANSA Reference Manual* which explains the technical content of ANSA in depth, are freely available from the Project Director, Bill Talbot, or the Chief Architect, Andrew Herbert, at the ANSA Project office in Cambridge (tel: 0223 323010).

Community management for the ICL networked product line

Alan R. Fuller
ICL, Marketing and Technical Strategy, Bracknell, Berkshire, England

Abstract

ICL's Networked Product Line (NPL) is the product realisation of ICL's Information Processing Architecture. NPL Community Management addresses the management of a set of networked products together with the underlying network components, the networked services and their access. This paper outlines the scope of Community Management and describes the way in which the Management of a total Community of networked end systems may be realised.

1 Introduction

1.1 Context

The transfer of all types of information, speech, still- and moving-pictures, as well as simple text can be accomplished over short and over very long distances. The technology to accomplish all of these types of transfer is available at some cost.

For some types of information transfer (for example, speech, and data in the form of text), not only is the technology available and well understood but the cost of providing it has permitted the widespread introduction of facilities based on this technology. Telephone networks and telex networks are undeniably successful.

In contrast, the use of data transfer techniques in the direct support of, for example, distributed office functions is much less widespread. The implementation of a number of distributed office systems (which cater for the information transfer needs in a modest way) has shown that they too are undeniably successful. Nevertheless, one of the most important factors contributing to limit the success of *both* worldwide telephone networks and small distributed office systems is the problems of management.

In each of the above examples, the management problems are not always the same – in the case of large networks, a number of the management problems arise from the sheer size of the network. Simply keeping track of the

maintenance state of the many thousands of network components is a major management problem. In the case of smaller networks of nodes which form, for example, the elements of a distributed office, maintaining the complex information structures in a consistent state is again a major management problem.

The major challenge of NPL Community Management is to help solve the management problems arising from size and complexity.

1.2 Scope

Size and complexity are only two dimensions of management problems. Three further dimensions of interest are skill level, the networked elements subject to the management discipline, and the life cycle of networks and distributed systems.

If one looks at the components of distributed systems, four categories are readily identified as shown below:

- Subnetwork Components
- End systems connected via subnetworks
- Applications running on and systems providing services to end users
- End users.

Further, the management tasks are often carried out by different people because different skills are necessary to accomplish the tasks.

A second major challenge for NPL Community Management is to provide facilities to support people with appropriate skills in the task of managing all four categories of managed components.

Each of the areas is discussed in further detail.

Networks have traditionally been considered as composed primarily of those elements involved in the transmission and switching of streams of information. Typical examples include point-to-point circuits, packet switching nodes, wide band transmission facilities, circuit switches, modems and multiplexing plant.

In a number of instances a valid view of the network can be obtained by aggregating the view seen from end systems attached to the network, as in the case of end systems interconnected by a local area network.

The familiar 'boundary' problems associated with end system-to-subnetwork connection have led to the inclusion of end systems themselves in the problem space to be managed. This initially implied just the communications elements of end systems such as couplers, line drivers and associated software and firmware. Increasingly, all components of the end system are being

considered, for example, closely associated with communications subsystem is buffering for messages being transmitted and the buffering is one aspect of the storage on an end system. If information transfer fails or is degraded as a result of problems with such storage, this too needs to be managed. In much the same way, if information is transferred from a file on a magnetic disc on one end system through a network to a peer system and the 'effective' transfer is limited by contention for disc channel capacity with other processes, this too needs to be within the scope of management.

The applications running on end systems attached to subnetworks have tended to be a 'province unto their own'. This is understandable since most applications are run on a single end system with networked access via terminals. Distributed applications (of which the most important examples are probably management applications!) have a number of characteristics that are such that the applications themselves too have to be within the scope of management.

A familiar distributed application is electronic mail.

Electronic mail involves the distribution of information to named individuals at specified addresses. The information used to achieve the distribution (even when the underlying subnetwork is static) is subject to change. People change locations, new recipients are added to the information network and correspondents leave the network. Further, new facilities may be added to the applications such that only compatible versions in different end systems are able to provide these new facilities. Such characteristics as these mean that not only the distributed applications but their associated datastores need to be managed for the distributed application to be effective.

Finally, the services (and component applications) within distributed systems are provided for the benefit of the end users accessing them. Unfettered access by users to distributed systems is clearly undesirable on a number of grounds: privacy of confidential information, the cost constraints are two important examples. Consequently, the mechanisms whereby users access systems need to be policed, which is equivalent to management of the end users.

All four categories of components of networked systems which constitute a 'Community' have to be managed for total systems management. NPL Community Management is appropriate to all four categories.

The scope of management is equally wide as far as the life cycle of systems is concerned.

The list below gives the typical stages through which either total networks or elements of networks pass during their overall life cycle.

Planning
Definition

Generation
Distribution
Installation
Monitoring
Operational Control
Fault Resolution and Diagnosis
Evaluation/Enhancement Planning.

To ensure that an adequate service is provided, networked systems should be *planned* well from the earliest possible stage. This includes a number of modelling activities so as to ensure correct functionality at each node in the network. Correct interworking capability, correct logical addressing, capacity in terms of storage on the end systems, connectivity, multiplexing capability (where appropriate), bandwidth between subnetwork nodes and appropriate traffic capacity at switching nodes all have to be taken into account.

Once the overall network has been planned, the detailed *definition* in terms of product-specific components has to be achieved and from the detailed definition the precise details of the networked systems is *generated.* This includes selection of options which, although necessary, do not materially affect the overall system. It also includes specification of some options which realise the principles embodied in the modelling phase.

Next, the parametric information has to be transferred to the location where it is to be used – a process of *distribution.* The final act prior to use of the information in a live system is a process of incorporation into the running system – a process of *installation.*

The phases in the life cycle so far can be collectively termed administrative processes; the subsequent stages are conveniently termed operational processes – the main difference between the two sets of processes being the level or degree of unplanned activity involved. Almost by definition, administration is the execution of planned processes whereas operational processes are required to respond to the un-planned event. In the scope of operational processes are *monitoring* processes which ensure the correct functioning of the total set of network components. In the case of unforeseen events, the *operational control* processes are invoked to effect, for example, bypass mechanisms or standby facilities. Once such unplanned events have been signalled, *fault diagnosis* and *resolution* facilities may be employed to determine the cause of any malfunction and bring into operation any remedial action.

As part of the monitoring activity both long term trends in the form of statistics and isolated incidents in the form of network alerts can be important.

The operational aspects relating to satisfactory resolution of a problem span the whole spectrum of managed components. For example, standby facilities

may be invoked, routing tables may be changed, network tasks may be deleted or re-created, the availability of a service may be suspended or resumed and the permissions for a user to access a service may be suspended or re-instated.

The same basic facilities used to plan and define the networked system are employed in the enhancement planning phase of networked systems to cater for growth (or decline in usage) of systems.

All functions in the life cycle of distributed networked systems have to be catered for in order to manage systems in an effective manner.

In terms of people involved in each of the stages of the life cycle, a variety of skills is required. In the early planning stages, imagination tempered with what can be achieved in practical situations is required. In the definition and generation stages, a more detailed product knowledge and methodical (almost mechanical) approach is demanded. In operational management assertiveness coupled with an ability to think laterally is essential to problem resolution.

In each of these areas, the skills necessary are not only different but they are in short supply. NPL Community Management concentrates on assisting the skilled tasks making skilled people more effective rather than de-skilling the tasks – in any case, the techniques involved in making skilled personnel more effective are a necessary precursor to de-skilling the tasks themselves.

2 Principles

2.1 Mechanisms and building blocks

From an analysis of the tasks involved in supporting the total life cycle of distributed networked systems two points are evident:

- a number of tasks require the transfer of information.
- a number of tasks require a means by which pieces of information in different components of the total system are kept consistent.

The two major divisions are inter-related. Further detailed analysis of the needs for information transfer indicate three broad categories namely:

(a) bulk data transfer of files of information
(b) a message passing system to signal, for example, some malfunction or to signal correct functioning of some system – "I am working correctly".
(c) a means by which system components may be operated from a remote point.

The tasks which require consistent information are best thought of in terms of conventional database technology:

(d) the early stages in the life cycle of networked systems require some means by which the network definition can be held.
(e) the operational stages require a central point for information relating to network problems.

Each of these five basic enabling mechanisms is able to support a far more diverse set of network management applications.

(a) Bulk data transfer encompasses the processes involved in transferring files of accounting information, files of statistics for fault trend analysis, files of configuration data and so on.
(b) The messaging system permits the malfunctioning not only of a network level component to be signalled, for example of a modem or a multiplexor, but also that of a total end system or a communications coupler. It also enables the criticality of buffering in an end system to be brought to someone's attention. It enables end system-related problems to be recognised – for example, file store is becoming full. Further, application-detected problems are within its scope and last, but not least, attempted security violations may be raised as an alert.
(c) In much the same way the remote operation of distributed components has a wide range of applicability. Access via a network to networked components enables faults to be diagnosed, possibly by running diagnostic tests including boundary tests. Once the cause of a problem has been determined, the corrective action may be taken remote from the point where the problem occurred.
(d) The sets of managed components (subnetwork, end system, application, end users) work together in a well defined logical manner – users access services and applications, the applications reside on end systems, the applications interwork with each other (which means that end systems intercommunicate) and they are inter-connected by a variety of subnetwork components. Both the logical and physical relationships between all the managed components are required to be consistent.
(e) Once the distributed system is operational and events (not necessarily malfunctions) occur it is highly desirable to have a single point where information is held to assist in the initial stages of problem resolution.

For each of these five basic mechanisms, NPL Community Management has defined a general solution.

2.2 Bulk data transfer

Although the transfer of bulk data is well understood, using for example ISO FTAM as the transfer mechanism, further attributes of the management process are important. For example the fact that a bulk data transfer was successful (or not) is an important management aspect. Further, it is frequently the case that a group of file transfers (and the status of the set) is more important than any one transfer. Typically, the availability of several

files of accounting information is necessary before a process which relies on the complete data set can be run.

NPL Community Management provides such a managed bulk data transfer facility; it is termed Community File Transfer.

2.3 Messaging systems

The wide scope of events which may be signalled by a messaging system has already been described. Not only is it necessary to signal problems – the negative side of an event – but it is sometimes necessary to signal the absence of problems – a watchdog capability – such that as the number of operational problems diminishes the correct working state of a system may be positively determined.

NPL Community Management provides such a messaging facility; it is termed Community Alert Management.

2.4 Remote operation

The provision of remote operation is probably the simplest of facilities. The complexity arises from the diversity of systems for which remote operation is applicable, together with the diversity of end system support tools available.

NPL Community Management provides a remote operation facility; it is termed Remote System Support.

2.5 Consistency of network related data

Although the need for consistency of network related data has already been covered, there are many complex attributes. The consistency needs to cover the logical network components (which services interwork with each other) as well as the physical components (which sub-network elements interconnect which end systems). The realisation of the physical components in terms of hardware, and the logical components in terms of software is an issue of consistency of network-related data. At any instant the total logical/physical/hardware/software description of a distributed system can be held in a database. It is changes to this definition that need to be managed. For example, the network definition yesterday which is different from that today, which is different again from the planned definition tomorrow is an issue which has to be managed.

NPL Community Management provides a database to maintain these many facets of network definition in a consistent manner; it is termed Network Definition Database.

2.6 *Single point of contact*

It is commonplace that a problem received by one user is frequently encountered soon afterwards by another. Many problems have just a single cause. Many problems are not problems in that a fault condition persists – users may simply misunderstand information they are given; the inability to follow the instructions is a problem in their eyes. Research has shown that as much as 80% of seemingly network-related problems can be resolved through use of a single point of contact. The single point of contact may be accessed by telephone or via the network and the simple point of contact is ideally supported in its function by a database used to store the problem reports as they arise. A wide variety of reports may be generated from an analysis of the database relating to who is responsible for resolving a problem, detailed description of the problem, and when, how and by whom a problem has been resolved.

NPL Community Management provides a database to support a single point contact; it is termed the User Help Desk.

2.7 *Integration of different management disciplines*

Historically, different management disciplines have been developed for specific technologies. Typical diverse examples are the management disciplines involved in voice networks and, in contrast, the disciplines involved in managing the software comprising distributed applications. From the description of the basic enabling mechanisms it can readily be seen that they are applicable to a wide range of management disciplines.

As will be seen later, this is only one of several features of NPL Community Management which brings a unifying influence to the diverse management disciplines.

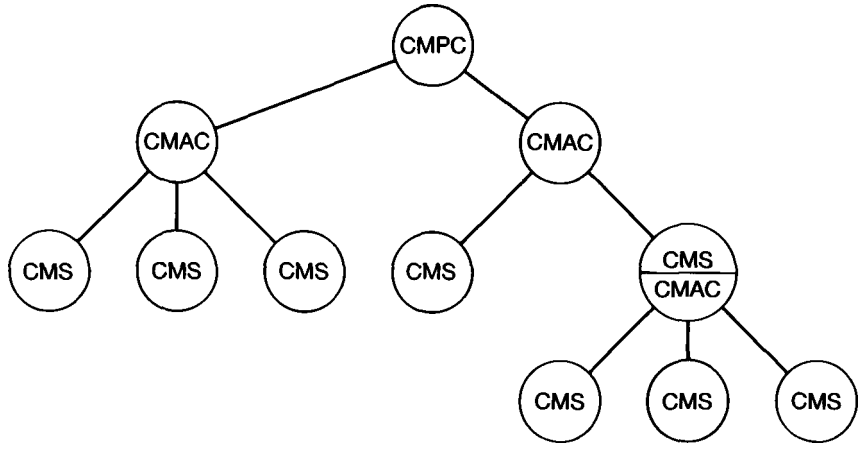
3 Architectural framework

The generic solutions to some basic management problems described in the previous sections can each be used in isolation. For the maximum benefit to be obtained, a number of the mechanisms may be used in combination within the architectural framework within which the mechanisms were conceived.

From previous discussion, the scope of the managed components includes the network, end systems, applications and users – the set of managed components being termed a 'Community'.

The architectural framework is based on the premise that management is a hierarchic discipline. The hierarchy may be 'flat' – a one-to-many relationship; or it may be 'tree-structured' with several branches to the managed 'leaves'. The leaves of the tree are termed Community Management Subsidiaries (CMS) and the entity which exercises control over these subsidiaries is

termed the Community Management Administration Centre (CMAC). This is shown schematically below:



The hierarchy of a single Administration Centre with several Subsidiaries is a 'flat' realisation of the hierarchy and is called a management 'domain'; a series of Administration Centres in a hierarchy can be logically extended to any appropriate depth. In this case, the intermediate Administration Centre is a sub-CMAC and it contains elements of both CMAC and CMS functionality but co-located.

The Community Management Presentation Centre (CMPC) is where the management information is presented and this is logically separated from the Administration Centre where the management decisions are taken.

The former (termed the Community Management Presentation Centre) will be co-located with the CMAC in a number of instances.

The lines between CMAC, CMS and CMPC indicate logical network links. This implies that management information transfers pass over the sub-network connecting the networked elements. This is frequently the case, but a number of system functions can still be adequately managed even when the distributed system elements are not networked on a permanent basis.

The managed components in many instances are the hardware and software components of end systems and subnetwork components. In these instances they are equivalent to Subsidiaries.

The facilities provided by NPL Community Management map onto the architecture in the following way:

Community File Transfer – the transfers occur between the Administration

Centre and the Subsidiary and they are managed by the Administration Centre.

Community Alert Management – the messages are passed between Administration Centre and the Subsidiaries and the major management decisions are made at the Administration Centre.

Remote System Support – Subsidiaries are managed from the Administration Centre.

The *Network Definition Database* and the *User Help Desk* are located on the Administration Centre.

From the architectural approach, a number of points are worthy of note.

The architecture shows an Administration Centre as logically responsible for a number of subsidiaries. It is convenient to partition management functions in this manner, for example to provide one Administration Centre for Directory Management, one for Database Management, one for Modem Management and so on. However, the architecture permits these to be not only separate logical relationships *but* also separate physical Administration Centres. This facilitates a distributed approach to management in harmony with the perceived need to enable users of management facilities to partition management as they desire.

One of the driving forces in this task of partitioning is the desire to permit skilled staff (who are in short supply) to be located where they can most effectively accomplish their management tasks. Should a more centralised approach be desired, the logical management domains may be co-located at a single Administration Centre.

4 Towards open management

One of the real driving forces behind Open Systems Interconnection standards is the need for information transfer between different vendors equipment. Multivendor interconnection is a goal of OSI; multivendor management is a goal of NPL Community Management. For this reason the information transfer mechanisms within Community Management have been firmly based on open multivendor transfer mechanisms. Where fully ratified standards are not available, practical intercepts of these standards have been adopted. ICL is committed to migrate to appropriate standards when they are available.

4.1 For the Managed Bulk Data Transfer

The current file transfer mechanism is the Network Independent File Transfer protocol (NIFTP). The international file transfer protocol (ISO

FTAM) is now well understood and this is the target protocol for Community File Transfer.

4.2 For the Community Management Alerting protocol

No international standard currently exists, although the OSI Management protocol is similar to the Alerting protocol, OSI Management (as currently specified) has a narrower scope in terms of the managed objects over which it exercises control. The same underlying protocol elements are appropriate to both OSI Management and Community Management Alerting Protocol. As the OSI Management protocol reaches maturity, it is expected that there will be significant commonality with the Alerting protocol and the Remote System Support described below. Community Alert Management will employ the OSI Management protocol in appropriate circumstances.

4.3 The Remote System Support facility

This currently employs the NPL/IPA facility Remote Session Access. The Remote System Support Function may be achieved by several open mechanisms. Firstly, via the virtual terminal protocol and secondly via the standardised 'command/response' elements of the OSI management protocols. Both mechanisms will be employed when the standards are mature.

The consequence of basing NPL Community Management on open transfer mechanisms are clear – NPL Community Management is one step closer to open management.

5 The relationship to management standards initiatives

There are a number of significant management standards initiatives which either have had an impact on, or which are closely related to, NPL Community Management. They include:

1 IEEE 802.1 Layer management
2 MAP/TOP Management
3 ISO OSI Management
4 ANSA Management
5 Distributed Systems Management.

In very general terms, management interactions in many instances can be modelled in terms of an application layer protocol which is used to manipulate managed objects in subnetwork elements and end systems.

The IEEE Project 802 defined a protocol for managing objects representing entities associated with the protocols which form one realisation of the lower two layers of the ISO reference model with specific reference to local area networks. NPL Community Management collects information about some

of the managed objects defined by IEEE for products in which IEEE protocols are employed.

As part of the evolutionary process to OSI management standards, the General Motors MAP specification, initially at version 2.1, employed the protocol developed by IEEE and extended the scope of the managed objects to cover protocols realising all the layers of the 7 layer OSI protocol stack. At version 3.0, the specification has evolved to intercept the emerging OSI Management protocol (although the latter is still incomplete in some ways) and a wide range of managed objects has been defined.

Much of the detailed definitions of managed objects within NPL Community Management is aligned with the specification of managed objects defined by General Motors in the MAP specification.

OSI Management has an elegant structure for its protocol (which shows some similarities to the IEEE protocol) and it is structured into a number of component parts. These support different functional areas of management including accounting management, configuration management, fault management, performance management and security management. A further component supports features common to all functional areas of management.

In each of the three areas above (IEEE, MAP/TOP, OSI), the model employed is similar to the architectural model employed for NPL Community Management although the scope of the managed components over which management control may be exercised is somewhat larger for NPL Community Management.

In architectural terms, NPL Community Management aligns very closely with the management architecture developed for ANSA – the Advanced Network Systems Architecture for future networking developed under the auspices of the UK Alvey Directorate. This is an advanced research project funded by the UK Government. However, the ANSA project has to date not defined the protocols employed to achieve the management functions.

In the UK, an ad hoc group has been working on a model for Distributed Systems Management (DSM). The scope of this work and NPL Community Management is well aligned.

6 Conclusions

NPL Community Management addresses today the need to manage distributed multivendor systems. An open approach and a strategy of intercepting standards has been adopted.

An intercept approach inevitably involves an element of risk. It also involves contributing to the rapid progress towards international standards by

making substantial contributions to the standardisation bodies. ICL staff actively contribute to the work in the European Computer Manufacturers' Association (EMCA) and the British Standards Institution (the UK national member of ISO) and through these to ISO itself.

7 Acknowledgements

Some parts of this paper will be recognised by a number of colleagues. Many people from several Divisions in ICL have contributed to this paper – many of them without knowing it! Without a team effort that this paper represents, the paper would not have been possible.

References

1 IEEE Project 802. Local Area Network Standard. 802.1 Part B, Systems Management (Draft M, January 1987).
2 Manufacturing Automation Protocol. Version 3.0 Implementation Release. Chapter 11.
3 ISO/TC 97/SC 21/WG 4 N. Recommendations of the Third SC 21/WG 4 Meeting, Tokyo, June 1987.
4 The ANSA project; Andrew Herbert, This issue of this journal.
5 A report on Distributed Systems Management, April 1987, M. Sloman (editor), Department of Computing, Imperial College, 180 Queensgate, London SW7 2BZ.

The X/OPEN Group and the Common Applications Environment

C.B. Taylor
ICL X/OPEN Technical Manager, Office Systems, Bracknell

Abstract

The X/OPEN(tm) Group formed in 1984 is a unique grouping of major Computer System Manufacturers that is dedicated to the development of "Open Systems" through the creation of a Common Applications Environment (CAE) which is available on machines from all member companies. The foundations of the Common Applications Environment are interfaces from the UNIX(tm) System V Operating System and the C language, but this is being greatly extended to cover all the facilities necessary to provide a comprehensive environment for the support of commercial portable applications. The CAE interface definitions and portability guidelines are published in "The X/OPEN Portability Guide".

At the time of writing the X/OPEN Group has grown to include eleven major companies: AT&T, Bull, DEC, Ericsson, HP, ICL, Nixdorf, Olivetti, Philips, Siemens and Unisys.

1 Introduction

The formation of the X/OPEN Group was a direct result of two major changes in the Information Technology industry in the early 1980s, a growing user resistance to the long term lock-in effects of proprietary operating systems, and the emergence of the "Department" as a large scale user of computer systems.

1.1 Proprietary Operating Systems

Traditionally a computer system has been controlled by a proprietary operating system developed by the manufacture of the hardware. This has unfortunate effects for the purchaser:

(a) The investment made in applications, programmer training and operating procedures tends to cause a lock-in to a particular manufacturer. Changing supplier requires a substantial re-investment in time and money, and systems from different suppliers cannot easily be mixed.

(b) Because of the relatively small population of machines with a specific proprietary operating system, there is little incentive for the software industry to develop applications, and hence the application software tends to be limited to that developed by the manufacturer, or tailor made (and financed by) the user.

This means that most computer manufacturers find themselves caught in the trap of insufficient generally available applications to extend the base of installed systems, and too small a base to tempt the independent software industry to develop applications for it.

1.2 The Departmental Computer

The availability of 8-bit microprocessors in the late 70s gave rise to the Personal Computer explosion. These systems were not only cheap to produce, but also cheap to develop, and many start up companies were created. To reduce development costs these companies tended to use generally available operating systems such as CP/M and MSDOS, which, combined with the few standard microprocessors, meant that there were large populations of compatible systems. This in turn attracted a large number of software writers to develop general applications which further increased the market for personal computers, and a virtuous circle was created. As these systems relied on binary compatibility for application portability, there was later a tendency to standardise on the Intel microprocessors and MSDOS (and derivatives), largely influenced by IBM adopting these components.

This Personal Computer explosion highlighted two things:

- The computing power that could be brought to the desktop at low cost by use of microprocessors.
- The virtuous circle created by Industry Standards when the volume/cost equations are favourable.

The emergence of 16 bit, and more importantly 32 bit, microprocessors of ever increasing power meant that a similar scenario could be applicable to Departmental computing. It was now possible to service the computing needs of a Department at low, and therefore acceptable, cost. Market research firms predicted rapid growth in this area, and this "middle ground", which lacked a dominant operating system regime, was identified as a major area of opportunity by virtually every computer manufacturer in the world.

The situation was however more complex than it was for Personal Computers. The requirements of a Department are for sharing of information between users, concurrent access to the information and the ability to carry out multiple tasks simultaneously from the same workstation. The systems must also be flexible and expandable over a wide range, as Departments vary significantly in size. A simple control program, like MSDOS, was therefore

not satisfactory; a full multi-user, multi-tasking operating system was required; and the wide scope meant that standardisation on a single microprocessor at the binary level was not possible.

The result was that the leading contender, adopted by most manufacturers, was the UNIX operating system (and its derivatives). It had been designed to be multi-user, multi-tasking and flexible, and had been written in the C language to be easily portable to many different hardware architectures.

2 The birth of X/OPEN

A problem with UNIX was that, although it largely fitted the requirement, it had not been rigidly controlled with regard to standard interfaces. Initially even releases from AT&T had not guaranteed upwards compatibility (although by UNIX System V this was committed), and there were several quite different flavours from other sources, notably Berkeley University and Microsoft (XENIX(tm)).

During the first half of 1984, ICL approached the other major European computer manufacturers with the view to ensuring not only that there would be a single standard at the UNIX level, but also that a complete Common Applications Environment should be defined covering the basic operating system, data management, integration of applications, data communications, distributed systems, high level languages, "internationalisation" and all the many other aspects involved in providing a comprehensive interface for portable applications. This effort was to be seen as a complementary plank in the "Open Systems" movement to ISO Open Systems Interconnect (OSI).

Initial discussions were difficult, as amongst the companies all three major flavours – UNIX System V, Xenix and Berkeley – were strongly entrenched, but by mid year sufficient interest had been generated for Bull, ICL and Siemens to start detailed studies. They were joined shortly afterwards by Olivetti and Nixdorf, and the group, known by the codename BISON, was formerly constituted and in full operation by the autumn. This codename lasted until Philips and Ericsson joined, by which time the name X/OPEN had been adopted and registered.

3 Achievement and growth

The first phase of work, covering the basic "system calls and libraries", the C language, an ISAM file access definition, COBOL, FORTRAN and source code transfer, was completed by May 1985, a remarkably short space of time. It was published in the form of "The X/OPEN Portability Guide" in August, the period between May and August being devoted to turning the bald technical specifications into a polished quality publication.

The results of the second phase, covering "Commands and Utilities", the terminal interface, internationalisation, SQL and PASCAL, were published,

along with enhancements to the earlier material, as Issue 2 of the Portability Guide in January 1987.

Over this period there was a major influx of US companies (Digital, Unisys, Hewlett-Packard and AT&T), which was a measure of the early success, added great strength to the Group, and widened its scope outside Europe.

4 The common applications environment

X/OPEN is not a standards creation body; it is concerned with standards selection, refinement and adoption. It is similar in nature to the SPAG (Europe) and COS (USA) activities for OSI, in that it is putting together a tightly defined set of standard interfaces that will be present on all member systems (and wider). The Group is a pragmatic organisation. The members are not interested in the definition of esoteric standards which bear no resemblance to available products. The philosophy is to define portability standards that can be achieved in practice within a short time.

The general policy is to adopt International Standards, and refine their usage, and to adopt de facto standards where an International one does not (yet) exist. Only in areas where there is not even a de facto standard does it do creative work, and even then the general policy is to work with other standards activities (eg /usr/group). In all areas, though, the intention is to have a strong influence on the evolution of the standards.

The overall aim is to put in place a complete environment, "The Common Applications Environment" (CAE), that will exist on systems from all X/OPEN members such that applications can be written to run without change on all the machines. In addition, by placing these definitions in the public domain, it is expected that many other systems will also adopt the interfaces and take advantage of applications conforming to the standards.

5 The X/OPEN Portability Guide — Issue 1

The first steps taken were tentative but were important because they demonstrated clearly that the Group was achieving its aims and because they established a way of working which allowed difficult issues to be resolved by consensus rather than by majority voting.

By mid 1985, the Group had agreed common positions in a number of areas and these were recorded in the first Issue of The X/OPEN Portability Guide: a massive document comprising over 600 pages and setting new standards for the quality of UNIX documentation.

The individual parts are briefly described in the following sections:

5.1 The X/OPEN System V Specification

The most important decision taken during the first year was to agree on a standard "dialect" of the operating system to be supported on X/OPEN systems. Gaining consensus to base the definitions on (a sub-set of) the UNIX System V interfaces was a major step forward, and one that was taken well before publication of the AT&T System V Interface Definition. It is very important here to note that all the X/OPEN CAE definitions are about *interfaces*, and not about the underlying product. For example the Group has never jointly agreed to adopt the AT&T UNIX System V product, but merely some of its interfaces.

The first task undertaken by the Group was the definition of System V system calls and library routine calls. (For readers not familiar with the UNIX Operating System, these comprise a comprehensive set of operating system services invoked directly from within programs written in the C language.) When this work was nearing completion, AT&T published the first issue of the System V Interface Definition (SVID). The Group immediately recognised the danger of splitting the market if the two definitions were not converged, but also found themselves unable to adopt the SVID as published in its entirety. As a compromise, X/OPEN adopted the majority of the SVID definitions, but annotated where necessary to provide a definition to which all members felt able to commit. The differences were in fact small and largely in the presentation. All technical differences were explicitly referred to on the appropriate page of the Portability Guide. A small number of UNIX System V calls, that had not been included in the SVID, were also adopted by X/OPEN, some of which have been added in later editions of the SVID.

5.2 The C Language

Most of the existing UNIX applications and virtually all of the systems software are written in the C language. The American National Standards C Language Committee (X3J11) is working towards an accepted definition for the language. This is still in draft form and evolving. However, the version of the C language defined in the AT&T UNIX System V Programming Guide is commonly supported by a wide range of C compilers and was adopted as the basis of the X/OPEN definition.

As it is possible, in fact quite easy, to write non portable code in C, the Portability Guide includes advice on how to ensure that applications are portable. Also included are guidelines regarding pitfalls that could be encountered when the definition moves to that of ANSI X3J11, once it is ratified and conforming compilers are freely available.

5.3 FORTRAN

The X/OPEN definition of Fortran is simply the ANS FORTRAN-77 standard with no extensions, which is supported by the majority of current FORTRAN compilers.

5.4 COBOL

The work of defining portable COBOL was a little more difficult. The widely accepted standard at the time of publication was the American National Standard X3.23-1974 (COBOL 1974). Although widely supported it was becoming out of date in a number of ways:

- It includes features (such as SORT) which are now regarded as supplied by free standing routines or operating system facilities.
- It does not include facilities for the support of the online user (interactive input/output at a terminal).

The view of X/OPEN was that any COBOL portability definition which excluded screen handling was totally lacking in credibility. All current compilers do in fact support interactive input/output, most of them via extensions to the ACCEPT and DISPLAY statements. Unfortunately, not all of these extensions are compatible, and it was necessary to choose one for the X/OPEN definition. The one adopted was a de-facto standard, the form of ACCEPT and DISPLAY supported by the MicroFocus LEVEL II COBOL(tm) compiler.

It is important to recognise the difference between the adoption of (a subset of) the interface represented by a particular product and the product itself. There are now a number of different compilers which also support the X/OPEN definition of ACCEPT and DISPLAY. These are equally acceptable as components to support the Common Applications Environment. Conversely, the MicroFocus product included other extensions beyond the ANSI standard, which were not adopted as part of the X/OPEN definition.

5.5 Data Management

Data management is a key component of commercial applications and hence also of the Common Applications Environment. In the longer term, it was recognised as being necessary to support full database management interfaces, but at the time that Issue 1 of the Portability Guide was published, there was a state of anarchy among the database management systems with no common interfaces between them.

5.5.1 Indexed Sequential Access Method (ISAM) As an important first step, the Group adopted a set of portable interfaces for record access to Indexed Sequential Files (ISAM).

The C-ISAM(tm) product from Relational Database Systems Inc. (now Informix Corporation) was a clear market leader. In the absence of any international standard, a subset of the interfaces of the C-ISAM product were adopted as the basis of the X/OPEN ISAM definition.

A number of C-ISAM interfaces were excluded because they related too

specifically to that product, and/or were not necessary as part of the standard.

5.6 File Transfer

A major irritant to application software developers was found to be the difficulty of transferring files, specifically text files containing source programs, between different machines. X/OPEN therefore defined standard media formats which would be supported where possible. It was not possible to give absolute commitments in this area because of differences at the hardware level and also in the nature of systems.

Many systems support a 5.25 inch floppy disc drive, but the number of sectors per track varies from 8 to 10; the most common version is generally compatible at the hardware level with the IBM PC-AT, and has 9 sectors per track. For machines with a suitable floppy disc drive, X/OPEN defined a common 80 track format (conforming to ECMA-78) to assist in the moving of source code.

Many systems support a 9 track half inch magnetic tape deck. For machines with a suitable tape deck, X/OPEN defined a common 1600 bpi Phase Encoded format (with optional identification label.)

Many machines also support quarter inch cassette (or cartridge) tapes. These popular low-cost devices are supplied by a number of manufacturers but the recording format differs from one manufacturer to the next. Hence there was, and still is, no possibility of an X/OPEN definition for such devices.

6 A period of consolidation

The next stage was a period of consolidation during which five principal activities took place:

- Initiation of marketing activities to encourage Independent Software Vendors, Government procurement agencies and large users to adopt the X/OPEN Common Applications Environment.
- The growth of the Group to ten members, and the inclusion of major United States based companies.
- Extension of the Common Applications Environment into additional important areas, roughly doubling its scope.
- The development of a verification suite that would verify conformance to the published CAE.
- Working towards the delivery of compliant member systems.

The UNIFORUM conference and exhibition in Anaheim in February 1986 was very important for X/OPEN and clearly showed that the Group had become a major force in the UNIX world. X/OPEN did not have a stand at the exhibition, nor were any papers presented at the conference. However, a

large number of other speakers made reference to the Group identifying it as having a significant influence. Also Issue 2 of the AT&T System V Interface Definition was launched at the show, and it was clear that the X/OPEN Portability Guide had influenced the revision of the previous Issue.

It was at UNIFORUM that X/OPEN was able to announce the extension of the group to the USA, with DEC becoming the first non European member. DEC was rapidly followed by the Sperry Corporation (soon to become Unisys following the merger with Burroughs) and Hewlett-Packard.

On the technical front, an update to Issue 1 of the Portability Guide was published in the summer of 1986, and a second complete edition of the Portability Guide was published in January 1987.

The exact way in which the verification suite will be used has not been finally resolved at the time of preparing this paper. In the short term, its use will certainly remain under the control of the X/OPEN members; there are no immediate plans to issue the verification suite as a product.

7 The X/OPEN Portability Guide – Issue 2

Issue 2 of the X/OPEN Portability Guide was launched at the Uniforum show in Washington in January 1987. This edition, presented in five volumes, included a revised reprint of all the material included in Issue 1, and therefore replaced rather than complemented it.

A number of lessons had been learned from Issue 1. The form of presentation retained the same high quality paper and two-colour printing, but the binding was changed to a form which allowed the material to lie flat. The division into separate smaller volumes made it more manageable for the users, and easier to update and extend.

In comparison with Issue 1, the price has been increased only slightly, although the Portability Guide itself contains more than three times as much material. This is to ensure the widest possible distribution.

7.1 The X/OPEN System V Specification

In Issue 2, the X/OPEN System V Specification is extended beyond the definition of "system calls and libraries" into a number of additional areas, the most important being "commands and utilities", and "internationalisation".

7.1.1 System Calls and Libraries The definition of system calls and libraries included in the first edition of the Portability Guide was reissued substantially unchanged. There were some minor changes to correct errors reported and to reflect the evolution of the IEEE "POSIX" standard, but the level of change was not significant.

7.1.2 Commands and Utilities "Commands and utilities" differ from the "system calls and libraries" in the way in which they are invoked. They are primarily designed to be invoked via a command interpreter.

While the definitions of the system calls and library calls are reasonably stable, the same cannot be said of commands and utilities. The current UNIX commands have been developed in an ad hoc manner over a long period with no consideration of producing an Industry Standard. As a result, the current command definitions are inadequate in a number of ways:

- The commands themselves are often over-complex with a large number of interrelated options, many of which are hardly, if ever, used.
- The definitions are in many cases not rigorous, and are neither an accurate definition of the behaviour of the command nor a suitable basis for verification that the command operates correctly.
- Much of the behaviour is machine dependent and inhibits portability.

In the short term, the X/OPEN Group have prepared a definition of commands based on existing documentation, but, recognising the shortcomings, have liberally annotated the definitions with warnings where behaviour cannot be guaranteed across all systems. While the eventual aim must be to "rebuild the road", an exercise to place warning signs in front of the holes is a necessary and valuable short term expedient.

In the future, X/OPEN will take an active part in the precise definition of a set of standard commands, particularly those which may be invoked directly by (portable) applications.

7.1.3 Inter-Process Communication In Issue 1 of the Portability Guide, System V routines relating to shared memory and inter-process communication were excluded. The Group believed, and continues to believe, that more generalised procedures are necessary in this area. However, there are classes of applications (such as data management) where there is a need for such facilities now. In recognition of this need, Issue 2 of the Portability Guide incorporated the System V inter-process communication interfaces, but included caveats to the effect that some of the facilities are system specific and their presence cannot always be guaranteed across all X/OPEN systems.

7.1.4 Terminal Interfaces Although the Group continues to search for common high level user interface capabilities, many applications are currently being built using a library of routines (known as the "curses" library) that is being assembled, to achieve a degree of independence from the actual terminal type in use. X/OPEN has defined a set of curses routines which will be supported across all systems, and has given advice regarding the problems that can be encountered when using such facilities with synchronous (block-mode) terminals, or asynchronous terminals connected via networks.

7.2 Internationalisation

From the beginning, the X/OPEN Group recognised the importance of facilities to allow the user of the system to operate in his own native language. This implies a number of major requirements:

- The ability to input and output messages in the appropriate language.
- The ability to process the character set in use correctly according to the language and country involved (character type testing such as "is it a number", or "is it alphanumeric", conversions between upper and lower case, collating sequences, etc.).
- The ability to support the user's cultural conventions (date format, currency symbols, etc.).
- The ability to mix languages, so for example a French secretary using a machine in Italy can write a letter to a German company.

X/OPEN has defined a set of application interfaces for international operation. They were derived from the interfaces of the Native Language Support system developed by the Hewlett-Packard Company of Palo Alto, California. They have been further enhanced by X/OPEN and have been modified to be compatible with the Internationalisation proposals of the Draft Proposed American National Standard for the C Programming Language.

The initial system is defined to operate with 8-bit coded character sets, but the definitions are believed to be upwards compatible to 16-bit code sets later. Different codesets can be used to cover the many different geographical groupings. All codesets must have 7 bit ASCII in the LHS (top bit of character 0), but this is consistent with the rules for most internationally defined codesets.

7.3 Programming Languages

The C language, COBOL and FORTRAN definitions were included in Issue 2 of the Portability Guide with very little change from the previous issue. PASCAL was introduced for the first time.

The C language definition was changed in a small number of areas to reflect the evolution of the draft ANSI X3J11 standard.

The presentation of the COBOL definition in Issue 1 proved to be over-complex. The actual definition published in Issue 2 was substantially the same as that in Issue 1, but the presentation of the material was simplified considerably. Advantage was taken to eliminate some obsolete constructs (defined so in the new 1985 COBOL standard) from the X/OPEN definition. Advice relating to the use of COBOL in a UNIX environment was introduced.

The X/OPEN Group has adopted the ISO standard for PASCAL without any extensions.

7.4 Data Management

Data management was significantly enhanced by the inclusion of an embedded SQL definition for access to relational databases.

The ISAM definition was unchanged in Issue 2, with the minor exception that some routines which were defined as optional in Issue 1, pending a thorough examination, have now been adopted as mandatory components.

7.4.1 Relational Database Language (SQL) The availability of high quality relational database management systems with good performance, and the fact that the relational model is inherently simple and easy to understand, are such that they are now the preferred form of database system.

Initially, the available relational products incorporated proprietary application interfaces. However, a standard was developed by the American National Standard Institute, Relational Database Language (SQL) Committee (X3H2), and the rate of conformance towards this standard has been dramatic. This can be attributed to its adoption as a procurement standard by the US government and other influential bodies.

Immediately following the publication of the first Issue of the Portability Guide, the X/OPEN Group carried out a survey of the available principal relational databases, and it was clear that the emerging SQL standard would be a suitable basis for a portable definition.

The group carefully tabulated the capabilities of the market-leading products against the ANSI standard as it moved towards adoption, and, over a period of six months, addressed the many places where one or more product deviated from it. At a number of stages, the conclusions of the working group and the proposed definition were reviewed with the principal suppliers of the systems, and the final output was the X/OPEN definition for embedded SQL.

It would have been ideal had the group been able to adopt the ANSI standard (X3.135-186) in its entirety, but as all the major product suppliers do not fully conform, this was not considered practical. It was also believed that the standard would change in some of these areas. Furthermore it was considered desirable to include certain extensions to reflect common usage.

The ANSI standard allows two levels of compliance. The X/OPEN definition is based on level 1. To achieve level 1, some functions have to be present, but the actual method of implementation is not defined. The X/OPEN definition has included a common method of implementation.

In areas where existing popular products do not comply with the ANSI definition the X/OPEN definition includes warnings to application developers indicating facilities which are not universally supported in a consistent manner.

The full relationship to the ANSI standard is given in an appendix to the X/OPEN definition.

8 1987 and beyond

The immediate future for the X/OPEN Group looks extremely healthy.

The growing importance of X/OPEN was underlined at Uniforum 1987 in Washington where AT&T's membership of the Group was announced and people were besieging the X/OPEN stand for details of the Group and how to join.

In February 1987, X/OPEN was able to demonstrate the practical application of the X/OPEN philosophy at the headquarters of the European Commission in Luxembourg. A single application, complying to the X/OPEN Portability Guidelines, was successfully ported to machines from all of the member companies in a high-profile demonstration to press and major users from across Europe.

The Group now has a membership of eleven, with other companies eager to join. The policy of the Group is that membership is open. However, the actual expansion of the Group will be controlled to avoid any impact on effectiveness from an uncontrolled membership explosion. The other factor which could restrict the size of the Group is the "membership fee". The Group has an ambitious technical programme for the years ahead and this needs to be funded from members' contributions. To ensure that this does not prevent the Group gaining valuable input from smaller companies, it is intended that a series of liaison groups will be established.

At the time of writing, it is not possible to forecast when the third edition of the Portability Guide will be published or what it will contain. However it is possible to give some general statements of the direction that the work is taking.

8.1 IEEE POSIX

The IEEE has a committee, known as P1003, working on a standard for a "Portable Operating System for Computer Environments" which defines a set of interfaces that are largely based on those of the UNIX operating system. The shortform name for this standard is "POSIX(tm)". A document covering the System calls and Libraries aspects was published widely as a "trial use" standard in April, 1986, and there have been several working drafts since then.

POSIX was developed from the earlier work carried out by the /usr/group technical committee in the USA, which was also a major input to the AT&T System V Interface Definition, and to the development of the X/OPEN definition. The intention is that POSIX will be used as the basis of an ISO standard.

In January 1987, along with a large number of companies and other organisations, X/OPEN expressed its support for the "POSIX" activity, and declared its intent to converge with "POSIX" as and when it achieves "full-use" status. X/OPEN, as an institutional member of the IEEE, is playing an important role in moving POSIX towards final acceptance.

As stated earlier in this paper, the current definition of "commands and utilities" is not adequate. Rather than work independently, the Group intends to co-operate with the IEEE P1003.2 sub-committee to produce an agreed standard for "The Shell as a Programming Language". This co-operation will include major technical input and the allocation of skilled resources to work with IEEE.

8.2 Networking

The provision of appropriate networking interfaces is seen as a high priority by the Group. However, working in this area, X/OPEN found itself trying to run faster than the International Standards bodies, which in this case is not desirable.

1987 is likely to see the publication of an X/OPEN definition for an application interface to the ISO/OSI level 4 transport service.

Other work in the networking area is proceeding, but to give details at this stage would be premature. Areas being investigated include:

- Application to application interfaces (OSI level 7, peer to peer).
- Application interfaces to X.400 mail.

8.3 Feasibility Studies

X/OPEN is currently examining the requirements for standardisation in a number of areas. These include:

- User Interface (in particular graphics windowing systems).
- Transaction Processing.
- Security.

It is likely that working papers ("White Papers") will be issued for comment relating to a number of these areas during 1987.

9 Conclusion

The X/OPEN Common Applications Environment is rapidly becoming a reality with major importance for users and application developers.

Users can buy departmental systems with confidence. The manufacturer no longer has control over the user's future. At any time a change can be made to another supplier, or systems from different suppliers can be mixed, without penalty because there are now at least 11 major suppliers of X/OPEN compliant systems.

Users can safely invest in the development of applications in the knowledge that the investment is protected. A serviceable application will not have to be reimplemented just to change to a larger or more up-to-date machine, and there will be a wide portfolio of application products available from the independent software vendors.

Independent Software Vendors can see X/OPEN systems as a very significant coherent market, rather than as a series of small incoherent proprietary market fragments. This will make it much easier to recognise a sound business case for the development of applications.

Many of the definitions included in the X/OPEN Portability Guide are also supported by products on other operating regimes. Currently, the language and data management definitions are widely supported. The availability of applications using these interfaces in UNIX environments will indirectly benefit users of other regimes because of the reduced cost of porting. The UNIX Common Applications Environment will become a standard porting base from which ports to other environments will become more widespread.

The X/OPEN networking facilities supporting ISO/OSI protocols will ensure that X/OPEN UNIX based systems can communicate not only between themselves, but also with other machines both large and small. Thus a whole range of departmental applications which are able to complement mainframe systems will become available.

The X/OPEN Portability Guide, Issue 2 is available as a five volume set ISBN: 0-444-70179-6, from booksellers or directly from the publisher:

Elsevier Science Publishers, Book Order Department,
PO Box 211, 1000AE Amsterdam, The Netherlands.

Acknowledgement

Mike Lambert, the X/OPEN Chief Technical Officer, and I are founder members of X/OPEN, and have been joint editors of the Portability Guide. We have both written several papers describing the aims, objectives and achievements of the Group and have freely interchanged material. This

current paper is no exception and I am indebted to Mike for much of the wording.

Security in distributed information systems: needs, problems and solutions

C.W. Blatchford
Manager, ICL Secure Network Systems, Bracknell, Berkshire

Abstract

The paper discusses the risks to which information held in computer systems, especially in distributed systems, is vulnerable and indicates the means that are being developed by national and international official bodies and by the IT industry to ensure that the desired integrity and confidentiality of this information are preserved.

1 The problem

The criminal may be considered as the individual most responsive to the changes taking place in any society. Protect society with guns, then the criminal will attack with guns. Store valuable assets in 'robust' safes then the criminal will use dynamite. Control society through information held in computers and telecommunications systems, then the criminal will turn to information technology itself through which to perpetrate the crime.

Ignorance of such advanced technology has been considered an effective barrier against the malefactor. This is no longer true. The emphasis on the harmonisation and standardisation of computing procedures; the large number of young trained personnel; but above all the ubiquitous, user friendly workstation with its access to many services around the world could pose significant threats to society.

Comprehensive research being undertaken by key computer users and vendors alike has revealed some disquieting facts that constitute threats to Computer Systems.[1] In summary

(i) Some 30%, or more, of computer users, in one large UK sample, had experienced some form of computer crime.
(ii) Over 75% of crimes are perpetrated by 'authorised insiders', where control systems of user identification and authentication apparently have not been compromised.
(iii) Crimes are increasingly in the form of malicious damage to the information, there being no obvious financial benefit to the perpetrator.

This makes detection of the crime easy but apprehending the criminal very difficult.

(iv) Crimes are increasing in technical content, making resolution (and even quick detection) less achievable without the involvement of high cost, scarce, information consultancy resources, and/or fundamental system changes.

The information technology industry has worked closely with the other interested commercial sectors (Accounting firms, etc.) to establish stringent administrative procedures surrounding the information asset. Such procedures, even where comprehensive, require effective implementation, careful monitoring and regular updating. This requires consistently applied human resource, but here is a paradox: crimes are committed by 'authorised insiders' who can be the same people as are being asked to police the systems. What can be done?

The Information Technology industry is exploring facilities that will allow the consistent operational enforcement of the detailed security policies through the basic systems' hardware/software components. The computer vendors are increasingly turning for technical guidance to the Defence authorities, where such problems have been recognised and tackled in a rigorous fashion.

These Information Security or 'Infosec programmes' can give direction on how to create technical solutions to combat the computer crime in the commercial world. The creation of 'Commercial Security Cells' in both UK and USA, organisationally aligned with the defence community, is helping to focus the problem and recommend guidelines and even acceptable solutions to the market place.

2 Definition

The theory of computer security is imprecise, relying on a common understanding of the problems of computer crime. The balance between information systems functionality, assurance and underlying mechanisms needs to be better understood. Some broad classifications, however, are possible.[6]

Security concerns may be analysed into:

(a) Privacy – ensuring the confidentiality of information (and associated processes). Information has not been read and understood by the unauthorised.

(b) Integrity – ensuring that information and processes have not been exposed to alteration or destruction, whether accidental or by intent. This may range from 'simple' malicious damage through to complex information manipulation to effect a financial or asset fraud.

(c) Denial or deterioration of service – using associated information

processing resources so that the legitimate user may not have access to, or use of, a desired, pre-agreed level of a specific service.

The IT vendor requires the generic categorisation of computer crime so as to achieve focused hardware/software solutions, supported as necessary by user-driven administrative controls.

It is difficult to categorise computer crimes precisely. Specific recorded incidents are a mixture of intellectual creativity coupled with one or more of the following 'security' exposures.[2,7]

(a) Data diddling – changing information, the input/output procedures being driven through the work station: creating the false transaction, a dummy file, etc.
(b) Salami techniques – the thefts of small amounts of similar assets without apparently impacting the whole – the unauthorised collection of small amounts of data, the rounding to the nearest penny.
(c) Trojan horse – unauthorised, covert placement of computing code that can manipulate information or processes. These may either support, or be supported by, (d) or (e).
(d) Logic bombs – unauthorised computing to be executed periodically, based on some threshold level (date, value), that will cause some system malfunction.
(e) Trap doors – 'holes' in existing application or operating systems, utilities, etc. that allow additional authorised code to be eased in later but which by their nature will simplify unauthorised, logical access into the computer system.
(f) Superzapping – unauthorised use of computer programs that will by-pass controls to modify or disclose any of the contents of the system.
(g) Asynchronous attacks – interrupting or changing the sequence in which transactions are processed so as to create unacceptable conditions. This may be in a single computer or, more likely, in a distributed network environment.
(h) Scavenging – searching for a 'debris' left over from previous processes – from the waste bin to the residue of earlier application programs in the computer.
(i) Data leakage – the physical or electronic removal of data or copies of data or processes from the electronic service.
(j) Piggybacking – replacing an authorised use or service, etc. with an unauthorised one in an unterminated operation.
(k) Masquerading (impersonation) – by humans, computer services, hardware devices (computer, telecoms, etc.), assuming the identity of another (usually authorised) subject.
(l) Wiretapping – electronic tapping of data within a single physical location or via a gateway (public or private networks).
(m) Electronic eavesdropping (originally a subset of Scavenging) – unauthorised picking up the radio emissions from an electronic device and reconstituting the signal to obtain information.

(n) Simulation and modelling (a pseudo-crime) – the parallel interpretation of restricted data or processes in an unauthorised manner (the 'home mini-computer' with company data).

This list is not exhaustive – it could become longer, depending on the ingenuity of the criminal mind: thus the use of Artificial Intelligence or Knowledge Engineering techniques may result in many more, novel, criminal opportunities.

Any comprehensive taxonomy of the types, structure and impact of computer crime is classified by the relevant national authorities. Non-Government data available to the commercial sector is limited. Analysis of threats against architectural/technical solutions is underway.

3 User-corporate security policy

Security to protect against computer crime must be considered integral to an overall corporate administrative programme. Such a programme should have three major concerns:

- the Information policy (what information is required, how is it stored, for how long, how is it labelled, who owns it, how is it processed, etc.
- the Security policy (who/what can access what, when or how, in the context of both information and processes).
- the Electronic Resource policy (what objects are distributed, where to effect the processing).

An understanding of the importance of the information asset as the lifeblood of any organisation, of its relationship to corporate resources and the vulnerability of computer in an increasingly competitive business world is essential before the corporate security programme can be defined. A hierarchy of security statements is necessary including

(a) Policy – does the user wish to protect the assets, and from whom?
(b) Standards – to what extent is the user willing to protect and at what cost?
(c) Procedures – can the solution be administered by humans and supported by technology to an agreed level of functionality and assurance?

A corporate security policy may take a number of forms – but as a minimum it should address:

(a) Integrity/confidentiality of information
(b) Integrity/confidentiality of services/control functions provided by the system (both human and machine)
(c) Independent guarantees of operations/transactions (usually in real time)
(d) Authentication of users (not just end 'human' users)
(e) Access Authorisation Control, to services, functions, information

(f) Audit/Monitoring of services, functions, information.

The policy must be supported by Corporate Management and not left to lower management to define. The policy, if rigorously implemented, may have a profound impact on the human, as well as electronic systems, relationships and organisation.

One survey into Corporate security policies in 1985 showed that fewer than half of organisations in the UK had such statements and in no more than 12% had Corporate Management given a positive lead in operational enforcement. Recent, well publicised crimes (especially by authorised people inside the Financial sector) will have more clearly focused management attention.

4 Security solutions

Past computer security solutions have depended upon robust administrative processes with *ad hoc* implementation of selected computer technology (information encryption, etc.). More sustainable solutions will require a consistent understanding of computer security principles embracing all aspects of physical, administrative, and logical access control and of data storage and structure.

The proliferation of information systems down to the local user or workstation level has reduced the opportunities for stringent administrative and physical control. Greater reliance must therefore be placed on the logical access control systems (and supporting encryption processes), the security rules definition and the machine-based enforcement.

A 'shorthand' checklist has been prepared to facilitate validation of suitable security procedures. It is based on the language of the 3'A's and 5S's.

4.1 Logical access control

The 3A's – Authentication, Authorisation and Audit (Accountability) – The logical access controls over the information system.

4.1.1 Authentication: Establishes the validity of a claimed identity.[3] It is the procedure by which the system user or service identifies and validates the corresponding party in any two-way communication. It is primarily used for agreeing the human user but increasingly is necessary to check the service or computer in the communication process. Crimes have been committed by the correct end user being duped into interacting with a 'false service' on the computer and giving out information which could be collected, subsequently, by the criminal to perpetrate a fraud ('scavenging for authorised passwords').

Human authentication mechanisms have fallen into two prime catagories.

(a) Authenticating against 'What you have got': the credential – signature, token, knowledge (e.g. secret password), etc.
(b) Authenticating against 'What you are': physiological properties – fingerprints, retina scan, voice, etc.

The human signature is the familiar mode of authentication. There are major legal and commercial processes built around its use. The wide use of IT may undermine its value because it can be reproduced automatically. It may be considered to fall into both categories within computer authentication systems. It is a credential that can be learned, but masquerading is very difficult in the dynamic action of signing, even for the most experienced forger.

The advent of voice pattern analysis and keystroke dynamics opens up other control opportunities. Both have the characteristics of continuous, covert verification during an entire workstation session rather than verification at a single instance. This is significant progress in achieving authentication that protects against 'piggybacking' in addition to human masquerading.

Research into the area of keyboard dynamics has established that we each have a unique mode of operating a workstation. An initial profile, compiled from some 100 typed words, is stored in the computer for future operational comparison. It is of significant interest in financial and electronic office systems (Electronic Mail) where users are primarily a stable population of company staff.

The communicating of the resulting confidence in the identity of the human user to ensure accountability for actions is a major system problem requiring a consistently applied set of control procedures. Authentication between communicating services or computers is based on token exchange, usually in an encrypted form. The international standards and protocols for this process are currently being developed between vendors as part of the Open System Interworking and Secure Network Architecture activities. The OSI Association Control Service Elements standard recognises a graded authentication from weak to strong depending upon the application of encryption technique.

4.1.2 Authorisation: Access to data, processes and resources may or may not be authorised; the statement of what constitutes 'authorisation' is the cornerstone of the Information System Security Policy.[4,5] Access to an object potentially implies access to the information that it contains.

Access is a general word for the variety of types of interaction between a subject (an active entity in the form of a person, process or device) and an object (a passive entity – files, services, devices, etc.) that cause information flow or to alter the system state; and these can be as complex in a computer system as they are in any human administrative world.

A comprehensive authorisation model may be considered fundamental and integral to the architectural framework of any large information system,

whether in one computer or distributed around a complex network.[6] Historically, user authorisation structures have been vendor-dependent, usually built around the features of specific operating systems and associated computer utilities. This can restrict the view of the control processes to many disparate 'DOMAINs'. Large users, as they merge applications to create large distributed information systems, will gradually move to a homogeneous understanding of the Authorisation process. This single, open, but secure 'DOMAIN' will be driven by a 'Universal Authorisation Model'.

Authorisation rules are made in the context of systems-perceived characteristics possessed by the entities involved, of the state of the processing environment at the time, and of the type of operational access to data and processes that is requested. The characteristics of the entities are represented by the following classification of electronic data:

(a) Authorisation attributes associated with the subject (*privilege attributes*). The subject's name(s), its perceived rôle in the system and its trustworthiness
(b) Authorisation attributes associated with the object (*control attributes*). The object's name(s), its perceived rôle in the system and its degree of sensitivity or required integrity
(c) Context within which the request is being made. The time of day, the communications route involved, or the accesses currently being made to other objects by this and other subjects.
The overall systems architecture and the supporting Open Standards will constrain and possibly modify the context of the access, especially in a policy of controlled entities such as protected operations or applications.
(d) The kind of access being requested: read, modify, use, know-about, etc.

The rules of the authorisation policy are applied to values from these four categories and the result is essentially either 'access permitted' or 'access denied'. The algorithm representing the rules is typically complex, involving combinations of multiple entries from each category.

The existing ad hoc Authorisation methodologies available from vendors may be considered to bracket a spectrum of control opportunities. The *extreme* categories range from an object access authorisation being given to a subject (capability) through to a subject name and access authorisation assigned to an object (Access Control list).

In these identity based authorisation modules, the subjects or objects in the system have specific individual entity or data attributes. They *do* 'know' each other administratively and have corresponding identity labels attached (e.g. Joe can access File A). Increasing research is going into the administrative and control benefits of authorisation models where the subjects and objects in the system *do not* know each other. Labels, based on subject clearances and object classification, are used to effect dynamic coupling in the system

(e.g. Joe is Top Secret and therefore can access Top Secret File A). The Information Security policy may be applied as a set of rules against these labels to establish a complex hierarchy/matrix of relationships, at differing levels of control granularity.

The Defence community use the identity-and-label based authorisation models to support a further classification of Access Control into *Discretionary* (identity based, either subjects and/or groups) where a subject can pass on access permission to another unless constrained by *Mandatory* controls (label based, clearance/classifications). The non-defence sectors are currently exploring more flexible interpretations of Mandatory and Discretionary principles.[8]

Early simple, standalone information systems usually follow an identity based authorisation policy. Large distributed systems, operating with a volatile population of subjects and objects depending upon corporate organisational needs, will support label based processes, against specific rules, to allow effective multi-domain administration.

Experience with use of the VME High Security option[5] (in 1987/88) will feed into an Open System interpretation of the Universal Authorisation framework. The indication, however, is that many of the apparent administrative differences between Discretionary/identity based and Mandatory/label based processes could become superficial in some systems implementations.

It has already been established that one specific implementation of a Mandatory authorisation policy will give users *unique* non-hierarchic individual clearances. The clearances will then become equivalent to user IDs and the corresponding object classifications are simplified Access Control Lists (ACL).

The civil world will build upon guidelines, especially on the Defence Community – DOD – Trusted Computer Security evaluation criteria (ref. 'Orange Book'), to create a flexible rule-based labelling system for wide, networked application.[6,8] ICL has already undertaken preliminary studies into the extension of labelling and control to cover processes and procedures in addition to data and resources.

Functionally rich Authorisation models working within a distributed services environment require protection over the transmission of these labels. There needs to be a high level of assurance that the labels are held, managed and applied correctly and thoroughly against a set of protected rules. The adoption of operating systems that allow stringent protection of processes that manage and apply the rules in physically discrete networks or nodes is considered essential to a secure information processing environment (ref. VME-HSO).

4.1.3 Audit: (Accountability) is the recording (either historical or, increasingly, real time) of what is going on in the system. It is inextricably bound up

with the system resource management processes (e.g. community/network management, etc.). In addition to the obvious recording of the key end user transactions, database access, etc. it must monitor the integrity of the underlying security services. This will include:

(a) Changes to the security policy (privacy, integrity, etc.)
(b) Use of the logical access mechanisms (authentication, authorisation, access to and use of audit itself)
(c) Security violations at the logical access control (both actual and attempted)
(d) Job initiation and termination in the system
(e) Rule/label based system changes to security controls (including the subject/object clearances, classification labels, etc.).

The volume of historical data accumulated by the Audit process is becoming a significant operational issue. Industry/discipline profiles are being compiled to highlight those information, process or resource/network management issues that need to be actively and speedily addressed. Analysis tools are fundamental to good security management.

There is an increasing interest in real time security monitoring with the immediate notification of the malefactor in the system to a security manager; the subsequent operational action is a sensitive yet critical issue in information systems management.

There has been limited research into the full capability of computers/networks to detect anomalies in their usage and operation. The development of 'real time' intrusion detection systems using expert techniques will be given a high priority in conjunction with selected Accounting firms. This will become integrated into existing networking alerting products.

4.1.4 The packaging of the Access Control: The Authentication, Authorisation and Audit processes act as a troika, each supporting and complementing the others. For example – the Authentication process may identify the user (from some token/biometric properties, etc.) but if the communication has come from an unlikely physical location, the process could then allocate a less comprehensive privilege set. This could subsequently modify the subject labels and hence the ability to access objects in a rule-based authorisation model.

In some 'insider Crimes' the authentication and authorisation processes may *not* be compromised yet a crime has still been committed. The Audit process must then be able to store information in a highly confidential, tamperproof facility – the information system equivalent of the 'Black Box Flight Recorder', the information being made available only to the nominated, authorised 'Human Auditing' function.

The Secure Architecture will define the scope, role and positioning of the 3

'A's in the ICL Networked Product Line solutions. The current hierarchical architecture sets the 3 'A's as the prime responsibility of end systems operating environments, usually at the corporate level supplemented by the local work station as the network access capability.

The USA DOD Trusted Computer Security Evaluation Criteria – 'Orange Book' – and its MOD interpretation have set guidelines on the evaluation criteria that could be used to assess the degree of trust one could place on a computer system to protect classified data.[6] The scale, as defined, applies primarily to trusted, commercially available systems, covering functional and assurance requirements in general purpose operating systems. Specific, special environments (e.g. communication processors, process control computers and embedded systems) will be more adequately covered in further 'colour books'.

ICL has recognised the advantage of multilevel, secure, multipurpose operating systems running at the approximate equivalent of B1 plus on the scale, to support both end user applications and vendor-supplied operating software against the authorised, technically competent, criminal. This will reduce the chance of successful technical crimes from masquerading, Trojan horses, debris scavenging, superzapping, etc. by protecting the services and mechanisms of the 3 'A's.

The major ICL operating systems will include secure variants (e.g. VME HSO and Secure UNIX).[5,7] These will facilitate such protection by effectively partitioning any services associated with the 3 'A's in a processing node from any end user application software.

4.2 Secure communication

Distributed systems require effective and secure information management both within and between administrative domains. The controls can become quite complex if implemented fully and would embrace the storage and retrieval of information in addition to the communication between remote services.

4.2.1 The 5 'S's: Secret, Sealed, Signed, Stamped, Sequenced.

These terms were originally coined to support the secure communication of information, either by human or electronic means, between remote locations.[9]

(a) Secret – the sending of the message in a form that cannot be interpreted by the unauthorised (privacy/confidentiality)
(b) Sealed – the message had been structured to detect (and possibly rectify) tampering (integrity)
(c) Signed – the identity of the message sender can be guaranteed (authentication, etc.)
(d) Stamped – the confirmation of receipt of the message (service related issues)

(e) Sequenced – the message(s) have been received in order consistent with the wishes of the sender (integrity, service related issues, etc.)

These 5 'S's can be combined with the 3 'A's within a single information processing environment to create a blueprint for an integrated model of the security controls (as in the protection of control attributes and privileges in label-based Authorisation models).

Effective physical security (e.g. management of magnetic media), robust message syntax and protocols, 'hashing' of information, and encryption, should achieve the necessary level of security control.

4.2.2 Open Systems security: The Open System, ISO 7498/2 security addendum[10] defines the Security services required at each level and will influence the protocols and mechanisms necessary to achieve the level of secure association/connections between communicating systems.

The addendum satisfies the goals of defining the security services and outlining where they could be placed; however, the addendum is not exact enough for an implementor of security in the OSI architecture. It would be too expensive to provide all security services at all possible layers allowed in the addendum. In addition, if one implementor chose to implement a service at one layer and another implementor chose to implement the same service at a different layer, the goal of compatibility between peer layers of OSI would not be achieved. Standards for implementing the services are not yet fully specified; however ICL is working with standards bodies to ensure a realistic intercept strategy.

Priority has been set to achieve:

(a) Authentication (signed) – Peer entity and Data origin.
(b) Confidentiality (secret) – User data, Traffic flow. The services only being differentiated by layer, not by mode (connection/connectionless of operation).
(c) Integrity (sealed) – Connection/connectionless with recovery.

The services are being set at as low a layer as possible in the OSI model.

ICL encryption of the information will be used to support confidentiality/integrity requests. Authentication (at all layers) will be a consistent approach with possible industry specific standards implementation.

4.2.3 The ICL positioning of network security services: The level of desired control may influence the positioning of the selected subset of security services and the mode of operation of the transmission service.

The ISO Transport layer 4 has been selected for initial security implementa-

tion within ICL. The early development of layer 4 Security standard is facilitated by the stability of the Transport layer and its fundamental importance in networking.[10,11]

A Transport layer Security Service can provide

(a) Integrity Service – an integrity service using a connection-oriented mode can achieve the protocol richness of both 'sealing' and 'sequencing' the information packets.
The Financial System applications (e.g. EFTPOS) can be used to prototype the necessary control syntax (e.g. a 'hashed' message authentication code computed for each Transport data unit).
(b) Confidentiality Service – data can be maintained as secret in a network using encryption as part of an overall encipherment service. This is normally confined to the user data so that heading and control data are left as clear text to facilitate routing.
It is possible to determine who is communicating and how much data is being transferred, thus highlighting opportunities for malicious damage, etc. This traffic flow analysis can be overcome by encrypting at OSI layer 2. This would allow confidential communication hops but gives no protection in the intervening gateways.
(c) Peer Authentication Service – the signing/stamping processes can be supported by security protocols that have been defined for OSI Transport layer, assuring that peer layers have been mutually identified and that connections between them are current and not a replay of something earlier. This process can be duplicated throughout the ISO layers. The encryption procedures use random numbers and keys to avoid replay and the setting up of the control parameters (encrypted tokens, etc.) uses a separate connection process to maintain privacy.

4.2.4 Data Encipherment Service (encryption): An Encipherment Service consists of the data encryption/decryption algorithms, key distribution processes and physical/administrative packages and controls.[10,12]

There are two prime forms of data encryption:

(a) Symmetric, where the algorithm and key are the same for sender and receiver – so that a mechanism for regular key change and distribution is necessary to ensure secrecy. This is the *secret key system.*
(b) Asymmetric, where the algorithm is the same for sender/receiver but where the mathematics embody certain properties that allow multiple keys – some private, some public. This is the *public key system.*

The latter procedure is currently based around the difficulties of factorising large numbers (over 80 digits), the best-known implementation being the Rivest, Shamir, Adlenan (RSA) algorithm. As a privacy/secret mechanism, public keys may be prone to future mathematical breakthroughs which could invalidate the process. The method is, however, very valuable in establishing

valid authentication between remote location humans/messages (digital signatures).

The symmetric encryption system is more difficult to manage. A consistent systems architecture, a comprehensive encipherment service with regular changing of encryption algorithms can, however, be applied. The major control characteristics of such a system are the type and extent of the key distribution procedures. A limited number of low usage computer services, physically well protected, supporting a small controlled population of human end users, may be able to be managed with few changes of key. Physical insertion of keys from a portable key gun by the security administrator in some defined period may be adequate. However for most widely distributed systems the basis of control is through a transaction/session key, electronically distributed to the communicating services, as and when required. The maintenance of the confidentiality and integrity of such a key is considered a prime requirement.

The main operational components and protocols of a 'hierarchical' multipurpose, high security symmetric encryption key system, to meet both LAN and WAN encipherment services, will include:

(a) Physically secure implementation of master keys in all communicating devices and/or associated controllers (Human action).
(b) OSI protocols consistently applied to ensure adequate peer-to-peer authentication.
(c) Secure OSI Connections specifying the class of Service Request (initially at layers 4 and 3).
(d) Key Distribution Centre(s)/KDC – either centrally or distributed
    - under master key management
    - detection of replay and intrusion attempts
    - KDC accountable session keys (and integral device address information).
(e) Multidomain, remote secure peer-to-KDC communication and privilege transfer (especially within WANs or LAN/WANs).

ICL have developed a key management protocol – the Baxter/Jones solution.[12] It is currently being proposed as one of a number of suitable key management standards to the Open Standards community.

Baxter/Jones is suitable for both Local (Closed Domain), Wide Area Network (Open Domain) and any combination of networks in a complex customer network. Its use is consistent with the Secure Distributed Services/ Network Architecture.

Future encryption will be implemented in VLSI. The algorithms will not be published for debate in the 'public domain'. They will be registered for inclusion in a list of security services under the coordination of a recognised ISO body.

4.2.5 Service management: Security Management is management of both Services and the supporting mechanisms. It covers the management of security and the use of the security services to effect management control.

The quality of security required for any communication is driven directly through the layer management entities (LME) from a trusted Management Information Base (MIB). This may be stored in a secure end system (Multilevel Security, minimum B1 on the DODTCSEC scale) or on a standalone secure file server.

The current ISO Transport Level Connection security is specified qualitatively by selecting one of four protection parameters.[11]

(a) no protection features
(b) protection against passive monitoring (privacy/secret)
(c) protection against modification replay, addition and deletion (integrity/sealed).
(d) both b and c.

The driving of the Layer Management Entity from the (LME) management information base (as a constituent of Association Management) will allow additional parameters to be specified (e.g. for routing control, etc.). This procedure is characteristic of the ICL flexible approach to ISO standards implementation.

The Secure Distributed Services architecture will force the System Management Process from a closed proprietary system environment into the Open Systems public arena. Parallel work in secure government networking will facilitate this move.

4.3 Secure distributed services architecture

A consistent security philosophy, supported by the overall logical access control methodology, can be the basis for agreeing a consistent architecture for distributed services and networked information.

In Open Systems, security is recognised as a pervasive culture: only *authorised* data services or data entities must be allowed to connect and communicate. Adequate security is secondary only to the basic activity of communication itself. In this respect, security can be used as the validation of the structure of the information system, for both the user applications and the underlying enabling products (operating systems, utilities, physical devices, etc.) supplied by the vendor.

4.3.1 Architectural phasing: ICL has adopted the following architectural phasing for Secure Distributed Services/Networking, recognising the control characteristics of existing products and services.

Phase 1 – Secure Open Systems with a philosophy of trusted end user systems (computers) complemented as necessary by trusted communication networks. This supports the current ICL architectural blueprint of Corporate, Departmental and Local processing environments.

End systems will be associated with each other through a set of recognised correct routes, some protected by encryption, connection details of which are maintained in the end systems. The final security control of any single object remains with the host domain where the access rules are explicitly stated and maintained. There may be many separate discrete implementation of security policies with administration specific to each end system.

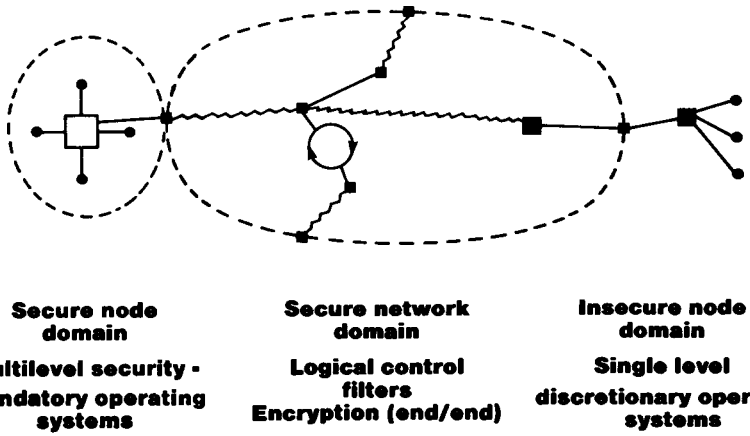


Fig. 1 Secure distributed architectures, Phase 1

The following technical facilities are required

(a) High Security/Trusted Operating Systems (DOD TCSEC-B level) both at the node and, increasingly, in a network configuration.
(b) Authorisation model offering a range of control opportunities from identity to capability based. The label based options would embrace both subject privilege attributes and object control attributes, with some flexible interpretation of identity/service relationships for 'commercial' Orange Book implementation.
(c) Standard 'Open Systems' association processes between communicating system (ISO 7498/2 with secure protocols). The mode of operation being flexed by the security needs (connection v connectionless issues, etc.). The current networking protocols at all layers in the OSI model are being reviewed to establish a capability for transfering security attributes in the form of protected labels.
(d) An encipherment service with selected encryption techniques to protect control/privilege transfer and, if necessary, end user information transfer between application services.

These vendor-supplier enablers must be complementary to the range of end user application software.

Phase 2 – A new architectural definition for information processing systems is being sought, especially with the advent of the distributed, end user driven automated office. End system services originally managed 'hierarchically' in a large centralised computer will in future be extensively distributed through the network. This Open System approach should eventually lead to homogeneous application of security policies and standards across most, if not all, end system domains in a network; yet still allowing some local control over specific mechanisms such as authentication or encryption algorithms.

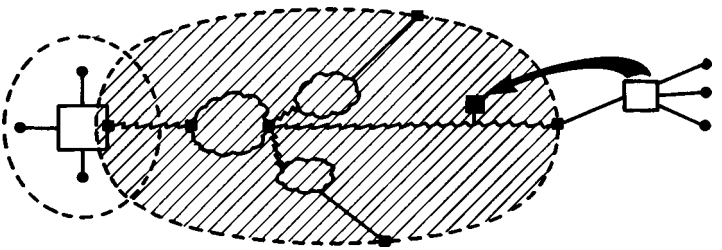


Fig. 2 Secure distributed architectures, Phase 2

The International standards community, primarily through ISO/ECMA with substantial ICL support and commitment, are defining this Phase 2 distributed processing environment. The analysis is currently underway;[4,10] early output from the various working parties indicates the following directions:

(a) Logical segregation of services into: PRODUCTIVE – end user contextual (applications, data bases, electronic mail, etc.); SUPPORTIVE-systems contextual enablers that are transparent to the end user but ensure the correct management and control of the application processes (services include authentication, directories, network management timers, etc.). Security services are considered primarily supportive.

(b) A USER SPONSOR, a high integrity software package, residing in a local workstation, that acts as the agent for the end user in interfacing with the various productive and supportive services. It sets up the environment, provides interfaces to the underlying facilities (including secure routing in association management), manages the menu selection and task specific service agents and audits.
This sponsor has two prime security responsibilities: monitoring the user's continuing presence (break in/out procedures, time out, etc.) and

organising relations with the various supportive security services between use of productive applications.

The user sponsor is supplied with other information to support the desired level of control: terminal attachment, route to the sponsor, identification of the individual (parameters include details of role, identity, access privileges, route, etc.).

(c) Specific Security Services – These are ISO layer 7 application level services and associated protocol. This is the current atomic list. In any specific implementation some may be merged (ref. Authentication and Attribute allocation in a PERSON SERVER facility).

- Authentication – accepts and checks subject credentials.
- Attribution – provides subject-related access privilege data and object-related access control data.
- Association Management – provides a secure way in which a subject accesses and acts on an object.
- Security State – records the current security conditions of the subjects within the system.
- Authorisation facility – actioning the 'Universal' authorisation model in some trusted environment.
- Interdomain (Gateway) – controls communication between domains of differing security policies in a distributed system. (These gateways may reflect different services authorisation models or mechanisms.)
- Security Audit/Recovery – current/past use of the security facilities in the systems and the procedures available to the security administrator to take corrective action (increasingly in real time).
- Encryption support – those encipherment services classified/graded for Productive Services and Support Services to maintain integrity and confidentiality.

4.3.2 Policy implementation: The Security Policy is translated into specific controls. These are then structured into Management Information Bases either end system hosted or distributed around the network (ref. OSI Management/ISO 7498 security addendum). Operational Management is via standard application level protocols.

The security policy for these security services may be more rigorously applied, and at a higher level of control (e.g. encryption as mandatory between distributed security services) than for the end user application services. The functionality of the services, their relationship, the positioning vis-à-vis the ICL NPL range and other regimes, the assurance associated with the underlying ICL mechanisms (operating systems, encryption) will give a uniquely powerful, Open, yet Secure, distributed services architecture.

Some secure distributed systems are being developed around physically separate, standalone servers divorced from the end systems hosting the user applications and productive services (e.g. as in cryptographic service management and end user authentication). Other solutions are recognising the

inherent power and processing economics of the large nodal/end systems computers by hosting the various open systems services under a rich secure operating system (e.g. VME High Security Option).

The rigorous application of such architectural principles to computer systems will gradually improve the overall integrity and reliability of information processing, not withstanding the external threat. The aim is for users eventually to have simple security blueprints, flexed by industry or discipline, which if followed will guarantee the quality of the administrative controls. A secure architecture endorsed by the international standards community, the vendors and, above all, the various national governmental authorities would create a firm base on which to build the information processing applications of the future. An architectural solution will protect us all from the technically competent, yet criminally minded insider.

5 Physical security – the new problems

Any secure architecture based on a combination of human and machine administered logical access rules and cryptomathematics needs to face a real world security test – can it protect the electronically stored information from technically less sophisticated attack or from those new types of crimes currently emerging in our society?

The last two years have seen a resurgence in the fundamental issue of physical security, i.e. in the physical and electronic characteristics of the equipment itself. There is a recognition that

(a) Servers in a distributed network may be physically small, easily portable and intrinsically vulnerable to physical tampering (the storage media, encryption chips, control panels, etc.).
(b) High speed electronic devices will radiate emissions which can be detected and, in the case of simple 'serial' signals, reconstituted with low cost, easily built eavesdropping equipment.[13] The Central Government/Defence solutions of administrative separation and physical barriers with equipment radiation hardening (to 'Tempest' standards) may not be politically acceptable or economically viable in the commercial sector.

In addition, are there fundamental changes in the way in which information is stored and processed that could change the long term control needs? The burgeoning portable intelligent token or 'SMART CARD' is of particular relevance. This can combine the properties of a physical access control key, a robust standalone active logical authentication service and large volume of personal, historical data.

6 Conclusion

Changes in society dictate that information technology and security should be the two fastest growing industry sectors. They need to be aligned more

effectively than in the past. Only then can we resolve the many complex social issues.

In Security, any mechanistic solution can be considered a temporary expedient. The philosophy, architecture, standards and specific products, at best, just increase the relevance and length of this control cycle in addressing criminal acts. The Market Research undertaken into the Security needs of its customers by ICL in 1985[1] indicated that some 25% of users where already stating that control (Assurance) was as important as functionality in information systems; but the two were expected to achieve equivalent status in procurement decisions by 1990. This has been recognised in the ICL Information Security Programme.

Acknowledgements

Good security, as a pervasive culture, will impact all ICL products and services, hence the development of the material behind this article has been a Team effort of many ICL Divisions. The Technical Security Standard subgroup (TS3) especially Tom Parker the Chairman, and other key members, A.C. Gale (NS), D. McVitie (MS), B Fairmaner and A. Somerville (M & TS) deserve special mention.

References

The ICL Security strategy embodied in this article has benefited from many sources – only the most significant, externally presented and published material is listed.

1 ICL-UKL (Northern Region): Corporate Information Systems and Security (Market Research), 5/1986.
2 BLATCHFORD, CLIVE W., ICL-M & TS: Towards the Secure Electronic Office, Computer Crime Forum, 4/1987.
3 USA Department of Defense: Password Management Guideline, CSC-STD-002/85.
4 PARKER, THOMAS A., ICL: Defence Systems, Security in Open Systems, A Report on the Standards Work of ECMA's TC 32/TG9, DOD/NBS X Annual Conference, 9/1987.
5 ICL: Introduction to the VME High Security Option, ICL Manual R30159.
6 USA Department of Defense: Trusted Computer Systems Evaluation Criteria, CSC-STD-001-83 (plus 85 update).
7 BLATCHFORD, CLIVE W., ICL-MTS: Secure UNIX and X-OPEN Stockholm, 6/1986.
8 CLARK, DAVID D. (M.I.T.) and WILSON, DAVID R. (Ernst & Whinney): A comparison of Commercial and Military Computer Security Policies, IEE E 1987 Security Conference, 4/1987.
9 BRANSTAD, DENNIS K.: Considerations of Security in the OSI Architecture, Institute for Computer Sciences and Technology, National Bureau of Standards Gaithesburg Maryland, 9/1986.
10 ISO TG97-ISO 7498/2: Security Addendum and the various working papers from: °ISO TC 97/SC 21/WG1, Architecture; SC 20/WG3, Encryption; SC 18, Text and Office Systems.
11 ISO 8072: ISO Transport Service standard.
12 JONES, R. and BAXTER, M., ICL: Defence Systems, The Role of Encipherment Services in Distributed System, Advances in Cryptology, Eurocrypt '85 Lienz Austria.
13 VAN ECK, W.: Electromagnetic radiation from video display units (Neher Laboratories), 4/1985.

Cryptographic file storage

David King
ICL Defence Systems, Winnersh, Berks RG11 5TT

Abstract

The significance of this paper is that it describes a cryptographic system that has been built. It is based on work by Gifford[1] although it derives its architecture from DSS (Distributed Secure System)[5]. The architecture is LAN-based and comprises amongst other things a file server and a key server. The files are stored in 'post-box' fashion and integral use is made of public key encryption to govern read and write controls.

Keys are categorised as either subject or object keys and are stored in key envelopes on the key server. Methods of controlling access to keys are based upon Gifford's sealing primitives. Keys are placed on a key server via a compilation process whose source is a specification of protection in a language called PROSP. The procedure is illustrated in an example and the file access procedure comprises a sequence of requests and replies between both servers, the special interface to the network and the workstation. It is possible to represent a variety of protection policies using the PROSP script. This simplifies verification by taking advantage of fail safe properties of a cryptographic system. Further developments might be to include a more refined set of access modes.

1 Introduction

This paper describes a system which has been implemented by the author in order to illustrate the ideas for using cryptography to protect information within a system rather than between systems.

The system is called PROSP and it is believed that a system of this nature can obtain a higher security assurance at a lower cost than similar systems based upon active controls only.

The implementation is similar to the Distributed Secure System in architecture[5] for the reason that physical separation of workstations linked by an ethernet only provides a secure processing base, whereas temporal separation of processing, as found in multi-tasking systems, requires a considerable amount of development effort to attain the same degree of assurance. The distributed architecture plays an essential role in PROSP for providing this additional security assurance.

The next section introduces the background of the system. The LAN architecture of the PROSP system is then described. The storage of files on the file server is covered, and then the key distribution and key recovery procedures, which govern who may access what, are described. Placing keys on the key server is done by producing the keys 'database' from a specification. The PROSP specification language is detailed in section 6, and the paper closes with an example and some conclusions.

2 Background

Protection of information can be seen to be either active or passive. By active protection we mean that access to information is controlled by an intermediary which mediates between the accessor and the object being accessed. This approach has been adopted in most multi-user computing systems. Passive protection is quite different in that information is 'sealed' (or encrypted) and access to it is governed by knowledge of a cryptographic key to unseal it. It is believed that the term was first introduced into the literature by Gifford[1].

The PROSP system is a hybrid system manifesting both active and passive protection, and it has been of considerable interest to see the extent to which passive protection can provide controls in a system which permits the sharing of information. A particular advantage of using passive protection is that the absence of a mediating party eliminates the threat of it being bypassed.

3 Architecture

PROSP is a distributed system based upon a Local Area Network of Honeywell Micro-System Executives (running Concurrent CP/M) arranged in an ethernet. Attached to the network are a number of workstations, a file server and a key server. The workstation provides users of the system with processing and some temporary local storage, while the file server stores files which can be shared by the user community. The key server is a repository of cryptographic keys which are used by the system for protecting and controlling access to information on the file server. It stores all its keys in encrypted form. The system master key grants access to all of these keys.

In addition, between the network and each workstation is a TNIU (Trusted Network Interface Unit) which ensures that all information which is passed on to the network is encrypted, and that no encrypted information is passed on to the user's workstation. It is the only place in the system where encryption keys are stored, albeit temporarily, in cleartext, and thus the TNIU has to be tamper-proof.

TNIUs are not needed to interface betweeen the servers and the network since it is required that information passing on to the file server remains encrypted, and in the case of the key server the keys are stored already sealed, and no further encryption is required. Thus each user can be sure that once

information has passed out from his TNIU it will be cryptographically protected. However, in spite of all traffic on the network being in encrypted form, it is necessary for each TNIU to be mutually authenticated to each server. To implement this, handshaking to establish session keys between each pair of communicating parties has been implemented.

Production of the keys and registration of users is the responsibility of the system administrator. He therefore has access to the system master key and, in addition, the software for installing keys on the key server.

Each user who is registered to use the PROSP system is issued with a key card which contains that user's personal key. The card is placed into the TNIU for the duration of the user's session. The TNIU reads the key from the card and uses it to recover further keys from the key server.

The organisation of PROSP, a name which refers to the system as a whole, is illustrated in Fig. 1.

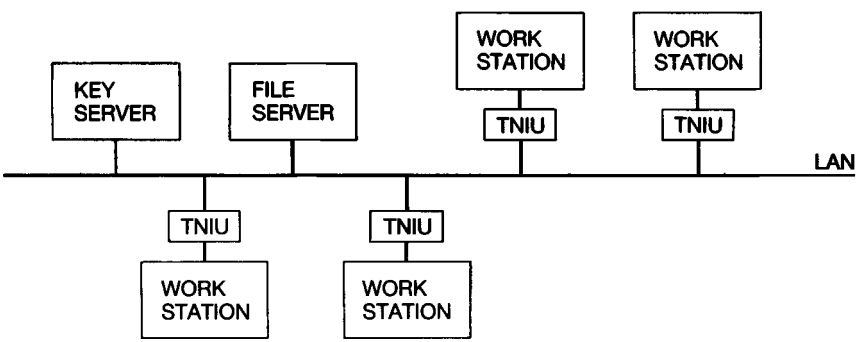


Fig. 1

The aim of PROSP is to provide an implementation of various protection policies through the organisation of cryptographic keys. This is the single most important feature of PROSP, and attention will be paid to this aspect in the remainder of this paper. However we will first outline the basic elements of PROSP in a little more detail.

4 File storage

Files are stored either remotely on the file server or locally on the user's workstation. The purpose of the file server is to enable a variety of users to be able to share information with each other according to a particular protection policy. The files are stored within the server in encrypted form. Access to a file is determined by the user's ability to obtain the decrypting key for the file. The names of the key(s) used for encrypting and decrypting the file are stored with the file itself[2].

In security policies, it is usual to speak in terms of 'Subjects', 'Objects' and 'Relations' governing access rights of subjects to objects. In what follows, we adopt this model, defining subjects of the system to be users and the user processes which run on the users' workstations, and objects of the system to be files. The keys which are used to encrypt and decrypt files are called 'Object Keys'. Files which have the same protection requirements are encrypted with the same object keys. However there are two different ways of protecting a file depending upon whether the subjects who can write to the file are different from those who can read it.

Public key systems provide a mechanism for implementing the latter, in which one key pair half is used for encrypting the data (writing it to the file server) and the other half for decrypting the data.[7]

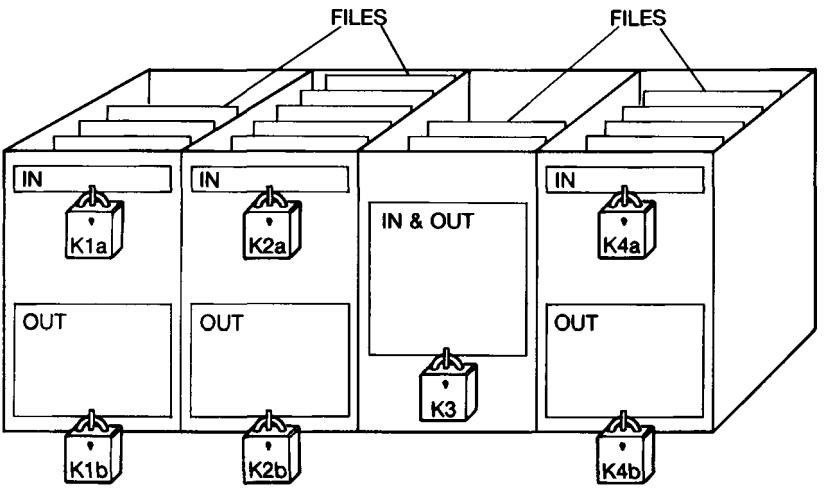


Fig. 2

5 Key organisation

Associated with subjects and objects are corresponding subject keys and object keys. Subject keys are given to users on the basis of their rights to access data. The users are provided with magnetic strip cards which store the user's subject key. Object keys are stored on the key server in encrypted form. We refer to such object keys as being contained within 'envelopes'. See Fig. 3. Access to any object key requires knowledge of a subject key, or a combination of subject keys, which will open the envelope.

Gifford[1] defines Key-And and Key-Or functions. Basically, a Key-And function allows an object key to be accessed if and only if a specific collection of subject keys is available. The object key is encrypted with each subject key in turn, and thus recovery of the object key depends upon knowledge of all

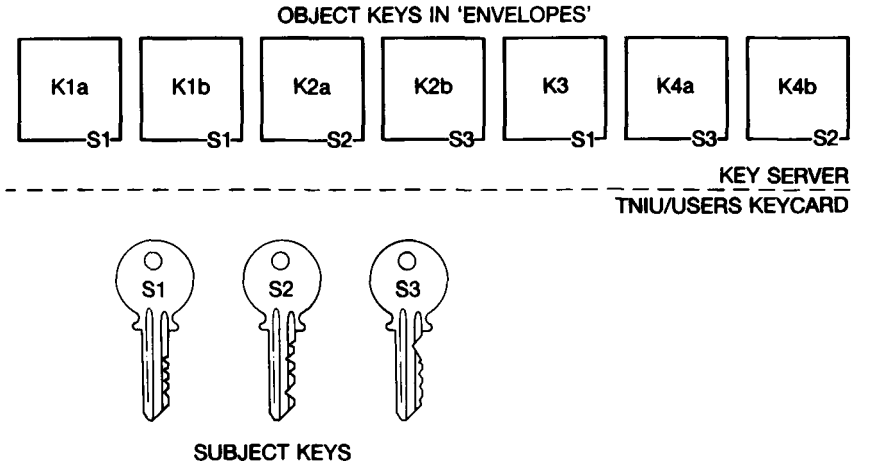


Fig. 3

the subject keys. The Key-Or function permits an object key to be accessed if any one of a set of subject keys is available. The object key is copied as many times as there are subject keys which will grant access to it. Each copy is encrypted with each subject key in turn. Thus knowledge of any of these subject keys will grant the subject access to the required object key. See Fig. 4.

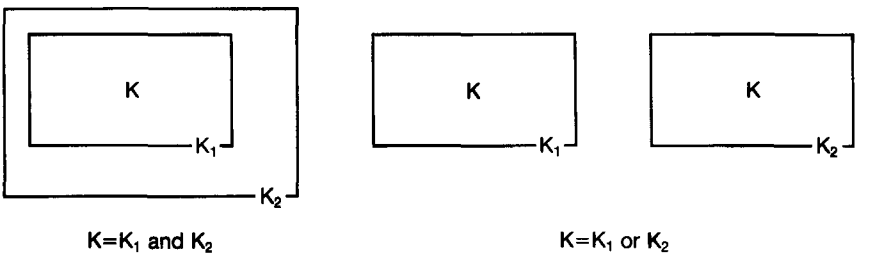


Fig. 4

Other methods of implementing Key-And and Key-Or are considered[3,4]. However, the encryption of keys with other keys to form key envelopes has been adopted and implemented in PROSP.

5.1 Key recovery procedure

The recovery of a key is triggered automatically by the TNIU on receiving a file access request. The TNIU request identifies the name of the object group key, either by asking the file server, or by asking the user via his workstation. The key envelope containing the object group key is then loaded into the TNIU from the key server, and the TNIU attempts to decrypt it with the subject keys it knows about.

This procedure is recursive in that if subject keys are not known, then the key server is questioned to see if the subject key is stored within an envelope. If the object group key cannot be recovered, because some subject group keys are unavailable, then the fact is reported back to the user's workstation.

Clearly, the control of keys upon the key server, and which keys are placed in envelopes that can be opened by other keys, determines which objects can be recovered by which subjects, and is largely determined by the protection policy. The entry of keys onto the key server, and the construction of key envelopes, is performed automatically by translating a specification of the protection policy (in PROSP script) into the keys and key envelopes.

6 The PROSP language

The purpose of the PROSP language is to get keys on to the key server. The language is a specification language and is non-procedural. It describes the protection policy in terms of cryptographic keys and dictates how the keys and the key envelopes are to be produced.

The system administrator constructs a PROSP script to reflect the desired protection policy. This is placed on the key server and compiled to produce the key envelopes. Once this has been done, the users are issued with key cards and the whole system is 'switched on'. This section describes in some detail some of the features of the PROSP language. The language allows the declaration of keys, which during compilation are generated by a key generator associated with the key server, and statements relating to the production of key envelopes to reflect the desired protection policy.

The structure of a PROSP script can be seen in Example 1. The first section is the TYPE section in which keys are declared to be of a particular type, for example 'DES' for Data Encryption Standard and 'RSA' for the RSA public key cryptosystem. The SUBJECTS section lists those keys which are identified with subject groups, and users who are members of a particular subject group will be given access to the key identified with that subject group. Similarly, the OBJECTS section lists the keys associated with object groups. Thus an object within that group will be encrypted with the key associated with the object group.

Keys declared in the TYPE section which do not appear in either the SUBJECTS or OBJECTS section can be used as intermediate keys which in some circumstances can simplify the readability of a specification.

6.1 Declaration of keys

Keys are declared in the TYPE section and the syntax depends upon the key type. Conventional keys are declared as follows:

```
tutor, student: DES;
```

Example 1

```
POLICY      Timetables;
TYPE
            tutor,
            student:DES;

            (timetable_read, timetable_write):RSA

SUBJECTS
            tutor,
            student

OBJECTS
            (timetable_read, timetable_write)

RELATIONS
            timetable_read:=student OR tutor;
            timetable_write:=tutor

END.
```

This statement would declare two keys with the names tutor and student of type DES. The declaration of a public key pair is shown in the following example:

(timetable_read, timetable_write) : RSA;

This statement declares a public key pair, one half called timetable_read, the other, timetable_write.

6.2 Declaring subjects and objects

The SUBJECTS section consists of a list of key names which refer to the subjects in the desired protection policy, e.g.

SUBJECTS—Student, Demonstrator, Lecturer

Single halves of key pairs may be included in the list, and each item must be separated by a comma. The OBJECTS section consists of a list of key names related to objects in the policy, e.g.

OBJECTS—(Exercise, Set_Exercise), Scratchpad

For any public key pair associated with an object, both names must appear in the list. The convention is to use one key pair half for encrypting the object and the other for reading it (the read key is always declared before the write key). Thus both halves are always associated with the object.

6.3 *Relating subjects to objects*

The RELATIONS section contains statements about the access of subjects to objects. The basic statement consists of the name of an object key, the operator ':=' and the name of the subject key, e.g.

Exercise:= Everybody;

This states that a user with access to the key Everybody can access objects associated with Exercise. There are two boolean infix operators, AND and OR, which apply to subjects, e.g.

Staff:= Demonstrator OR Student;
Area:= Classified AND Dept_A;

The object group, Area, can be accessed by a subject with access to both of the keys, Classified and Dept_A. The object group, Staff, can be accessed by a user with access to either the Demonstrator or Student keys.

6.4 *Compilation procedure*

The compiler generates keys according to the names in the TYPE section. Each name is associated with a key, and for public key pairs each name is associated with a key pair half. The keys which are to be used for encrypting information are tagged. These keys are those which are listed in the OBJECTS section and are either DES keys or the second in the pair of public keys. For example in the following objects section:

OBJECTS—(Exercise, Set_Exercise), Scratchpad

Set_Exercise and Scratchpad are tagged as being encryption keys. (Exercise and Scratchpad are using for decrypting information.) The RELATIONS section determines which keys are to be contained in key envelopes. The keys on the left of the symbol, :=, are to be encrypted, (placed in envelopes), with the keys on the right of the symbol according to Gifford's method of implementing Key-And and Key-Or. How this is done is illustrated in the following example.

7 An example

We now turn to an example to illustrate how these syntactic pieces can be put together to create a meaningful protection specification. The example is taken from a typical teaching environment in which there is a lecturer, a demonstrator and a group of students.

A protection mechanism is required to provide an environment in which students can develop solutions to workshop exercises and submit their solutions to a lecturer or demonstrator for marking. The exercises, set by the

lecturer, are to be stored on line for the students to read. The students' solutions are to be handed in to the lecturer on line. In addition, provision should be made for the lecturer or demonstrator to develop their own example solutions.

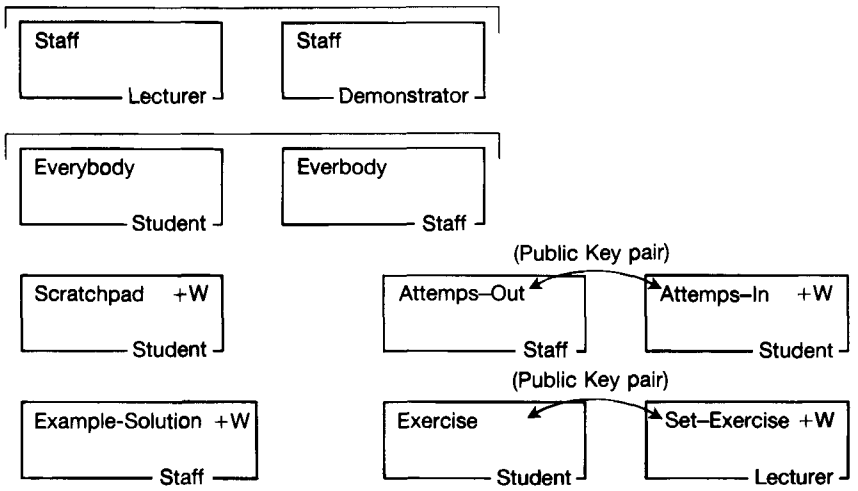
Example 2

```
 1 POLICY        Workshop;
 2 TYPE
 3               (Attempts_Out, Attempts_In),
 4               (Exercise, Set_Exercise):RSA;

 5               Example_Solution, Scratchpad,
 6               Student, Demonstrator, Lecturer,
 7               Staff, Everybody:DES

 8 SUBJECTS
 9               Student, Demonstrator, Lecturer

10 OBJECTS
11               (Exercise, Set_Exercise),
12               (Attempts_Out, Attempts_In),
13               Example_Solution, Scratchpad

14 RELATIONS
                 {Set up generic groups of users}

15                    Staff:=Demonstrator OR Lecturer;
16               Everybody:=Student OR Staff;

                 {Describe mechanism for a lecturer to distribute exercises to students &
                 others}

17               Set_Exercise:=Lecturer;
18                   Exercise:=Everybody;

                 {Now describe the mechanism for students to hand in their work}

                   Attempts_In:=Student;
                 Attempts_Out:=Staff;

                 {Students can share a scratchpad}

21               Scratchpad:=Student;

                 {Staff can store an example answer}

22               Example_Solution:=Staff

23 END.
```

The script for this scenario is shown in Example 2. We now look at what is described in the script. The keys Set_Exercise and Exercise are used to encrypt the exercise and read the exercise respectively. The students can obtain the key Everybody (line 16) and thus obtain the key Exercise (line 18) which they use to read the exercise. The students then develop their solutions

on their respective workstations. When a student is ready to hand in his exercise he sends his solution to the file server encrypted with Attempts_In, which he can access (line 19). Both the lecturer and the demonstrator can obtain Attempts_Out and can therefore mark the work; they have access to the key Staff (line 15) and therefore can access the key Example_Solution (line 22) which can be used by them to prepare an example solution.

The keys and key envelopes which are generated by the compilation of this script are shown in Fig. 5.



+W:
The keys are 'tagged' to enable them to be used for encrypting & writing information to the file server

Fig. 5

8 Conclusion

A system called PROSP has been built which uses the distribution of cryptographic keys as the basic mechanism for controlling access to files. The production of the keys in order to reflect controls for specific protection policies has been achieved by designing a compiler which takes as its source a protection specification in PROSP, and produces a 'database' of keys which reside in key envelopes on the key server.

The system has been tested with various scripts describing a variety of existing protection policies, in particular the Bell and LaPadula protection model[6]. In addition it has been used to investigate new types of protection policy[3].

A weakness of the system is its inability to distinguish between commonly

available file access modes, e.g. append, execute and delete. However, it is believed that the system as described could be integrated with a capability-based system in which capabilities would be associated with access modes. The specification language could be extended to cater for these modes.

References

1 GIFFORD, D.: Cryptographic Sealing for Information Secrecy and Authentication, CommACM, 25(4), April 1982.
2 GUDES, E.: The Design of a Cryptography Based Secure File System, IEEE Trans. on Software Engineering, SE-6(5), September 1980.
3 KING. D.: Passive Protection for Controlling Access to Information in a Shared File Storage System, Ph.D. Thesis, University of Birmingham, Edgaston, Birmingham, 1987.
4 KING, D. and JARRATT, P.: An Algebra to Represent Security Policies for Cryptography-Based Secure Storage Systems, Int. J. Computer Mathematics (Awaiting publication).
5 WOOD, J. and BARNES, D. H.: A Practical Distributed Secure System, SYSTEM SECURITY Online Conference, London, September 1985.
6 BELL, D. E. and LAPADULA, L. J.: Secure Computer Systems: Mathematical Foundations and Model, M74-244, The MITRE Corp. Bedford, MA., May 1973.
7 DENNING, D. E.: Cryptography and Data Security, Purdue Univ. Addison-Wesley Pub. Co., 1982, ISBN 0 201 10150 5.

OFFICE DOCUMENTATION AND AUTOMATION

Standards and Office Information

Gill Ringland
General Manager, Office Information Systems Business Centre, ICL, Bracknell, Berks

Abstract

International Standards for office systems are now being implemented to tackle tasks visible to end users – for instance the transfer of documents incorporating images and word processed source text over electronic mail networks. This paper examines some of the social and technical factors which have given impetus to this activity, and the effect this has already had, and will have in the future, on the rate and nature of the evolution of systems for workers in offices.

1 Introduction

1.1 Standards – the changing perspective

Today, few would disagree with the proposition that international standards are essential if information technology is to be fully exploited. Equally few, however, recognise just how significant the impact of those standards is likely to be. Users who accept the limitations of information islands, and manufacturers who dismiss full open systems integration as science fiction, are those who have not understood the recent pace of events in the international standards world.

The evidence is there for all to see. More has been achieved in the past four years of computing standards development than in the previous thirty. The same pressures which converted the great stationary steam engines of 19th Century factories into today's profusion of specialised electric motors and petrol engines, is at work in the IT industry.

Comparing the DP mainframe with the stationary steam engine is not so far-fetched. Power generated centrally was distributed by a series of belts to machines distributed around the factory. The distribution overhead was such that for small low-powered applications it was ineffective. If the central power source broke down, the whole factory ground to a halt and changing the system or adding new machinery was difficult.

The same way with information technology: the large, inflexible, and general is giving way to the local, the specific, and as small as possible. Chosen to meet very individual needs, departmental applications have become more

specialised. Yet the users expect a wider set of services, not limited by departmental interests, PC power or central DP convenience.

Users want information in understandable, usable forms, together with a wide range of services and applications. Unconcerned about information's origins or system topography, their view is the workstation and the services on the network behind it. They want electronic mail, access to remote or local databases and decision support. They are fiercely protective of their right to choose the very best system to suit their own needs, but they also expect full integration of applications, services and data within the system as a whole.

For the IT strategist it is far from easy to reconcile these interests. The key considerations today are networking communications and application interworking. Both are complex and evolving rapidly. Computers as mere service providers is a new concept and causes a revision of thinking and planning scenarios.

1.2 The role of customers in demanding standards

At the same time as the user perspective of the role of computing is changing, specialisation has become the name of the game for system vendors. Focus on market segments or specific technologies is the only way to achieve the excellence the market demands. Yet isolated solutions are rightly judged unacceptable. So there is growing pressure on manufacturers, from their customers, to develop standards which ensure that specialisation is compatible with system integration.

1.2.1 The need: IT strategists, particularly those in large organisations, have therefore formed the view that the essential, critical success factor within the strategy has to be the interworking infrastructure.

While mainframe DP environments will remain the backbone of an IT strategy for some time to come, the fastest rate of change is in focused departmental systems. Manufacturing, CAD/CAM, R&D, Retail and Office Systems all now require specialisation.

But clearly these segregations do not stand up in a changing environment, or in large organisations. Many manufacturers are also retailers, most have R&D facilities, all control stock and manipulate complex information processes within their office environment. The overlap of such disciplines is very wide. Good management of information is as important to a health authority as it is to a bank. And for each department, information generated in one, say personnel, will directly affect other departments, like accounts.

1.2.2 The initiatives: Manufacturers were among the first large user groups to recognise that information captured on computer systems was an infinitely precious resource. Two initiatives looked at the information flow

within manufacturing organisations and decided that data once captured should be available for use by both humans and machines throughout the organisation.

MAP[1], initiated by General Motors, looked at information generated by CAD/CAM systems and how this could be made available to a variety of manufacturing processes. The second and later initiative, TOP[2], was originated by Boeing. It looked at how information could be made available to all applications and services from CAD/CAM through to Office Systems.

With the pressure of worldwide competition behind them, both MAP and TOP made very rapid headway in selectively applying general standards created by CCITT (Comité Consultatif International Télégraphique et Téléphonique), ISO (International Standards Organisation), and others. This selectivity, with concentration on functionality observable to the end user, has created an invaluable focus for the propagation of systems that work, built to open standards.

Curiously, many of the collaborators in these groups such as Ford and General Motors, Boeing and McDonnell Douglas are deadly rivals. It is worthy of consideration that such titans are willing to subordinate individual advantage to create an environment in which the IT industry can service their real needs. The importance they place on 'open' standards should not be ignored by any players in this industry who wish to remain players in the next decade.

1.3 The evolution of standards

It is unlikely that IT standards will ever be finalised as long as there is scope for innovation. So those who wait until all standards are fully specified and ratified are going to be left on the sidelines.

ICL and other vendors active in the standards arena have adopted an intercept strategy. Armed with early information about any future standard, ICL takes known elements relevant to its particular applications and develops software based on these elements. They are designed so that when the standard is ratified, the proprietary version can easily be updated to conform to the final specification.

This allows product development to take place in parallel with a standards development, benefitting ICL and its customers. And with real systems being field tested in advance of the standards publication, information and ideas are fed back to the standards bodies themselves.

Another good reason for developing conforming systems today is that user driven initiatives such as TOP and MAP have clearly identified what the market wants. This has enabled subsets of wider standards to be created which meet the majority of these needs, so reducing development and testing

requirements. So it is cheaper and quicker to conform than would have been possible previously. Customers know that systems bought today can be upgraded to standard profile compatibility when required.

The business of collaboration between suppliers is greatly simplified, so it is easier to produce more complex and comprehensive systems. Yet at the same time it is easier for vendors to specialise in the areas they know best without fear of being left out of mainstream development.

That there is market demand for fully integrated and compatible systems is not in doubt. The speed with which major users have moved to develop industry specific standards is evidence of their desire to have such systems in operation as quickly as possible.

1.4 The European approach

MAP and TOP are initiatives by private industry in the US to push strongly towards the implementation of pillars of International Standards for building systems in specific environments. The US Government equivalent is GOSIP[3], which is a Federal Government procurement profile for open systems network products for use in the office environment.

Has Europe now been replaced by the US as the forcing house for open standards? The answer must be, to an extent, yes.

In Europe, the CEC has defined an 'Information Architecture'[4] which overlaps partially with the specifications in TOP and US GOSIP, but includes also aspects of application architecture.

The Standard Promotion and Application Group (SPAG[5]) took a lead on electronic mail in 1986 with the first X.400 electronic mail certification process, as a natural extension of its work in establishing implementation profiles since the early 1980s. SPAG will operate a 'green dot' procedure, in X.400 mail as in other protocols, which certifies that a supplier meets the standards. This procedure is in line with the verification procedures of the Corporation for Open Standards Interconnection (COSI), the most similar US-based organisation to SPAG.

The UK government (via CCTA)[6] has recently issued a working paper defining the UK Government OSI Profile (UK GOSIP) which aims to extend the base standards into more precise definitions (i.e. profiles) to meet the requirement that separately procured Departmental systems can work. It is a 'narrow stack' of OSI protocols for administrative IT services, including:

- electronic mail/messaging
- file transfer and management
- terminal oriented interaction
- revisable text and composite document interchange

- graphics interchange
- formatted data interchange

Clearly the convergence of the CEC Architecture, US GOSIP, TOP and UK GOSIP profiles is important to us as a European supplier.

2 'Office standards'

2.1 Social and technical factors

Standards related to office systems have been adopted particularly rapidly. This can be attributed to a number of social and technical factors.

2.1.1 Social factors: During the last few decades, the pattern of employment has changed. Service industries and white collar jobs now provide over two thirds of all jobs in most industrial countries, up from less than half over the last four decades. Traditionally, service industries – retailing, hotels, education – have not been capital intensive. The level of capital investment associated with white collar jobs is widely quoted as £2000 compared with 20 or 30 times that in agriculture.

Since 1955, white collar workers have increased from 10% of the population – these figures are based on US data – to 25% of the population and rising fast. At this rate, the entire population will be white collar workers in due course.

Of course, it won't happen. The pressure to be competitive, to increase productivity in commerce, in industry and in government, means that technology is exploited, and will increasingly be exploited in the office, to improve the effectiveness of information workers.

Now, the personal computer has been mooted as the office automation tool for the professional. Certainly the growth has been phenomenal, from the hobbyist in garages plugging pieces onto a S100 bus, to a £20 000m industry, in 10 years. Is this a pretaste of the future? Will the growth go on at the same dizzying rate? The answer is that the market is flagging. There are signs of market saturation.

Those who need a personal computer will soon have one or more. The hurdles are claimed to be twofold. One is the learning time for the use of applications. The other is the inclination for information workers to incorporate information technology into their work. Industry is improving its offerings to tackle these hurdles, for instance with friendlier software, more vertical applications, better support. But I would like to pick out another aspect. There is one part of the PC business which is booming – personal computers used to improve communications.

Let me give some examples.

Firstly, when we were trialling One Per Desk most of the first triallists already had PCs which were, like 9 out of 10 PCs, on the shelf not in constant use. But One Per Desk comes with an integral modem and links to mail services like Quikcomm.

The usage pattern of those trialled One Per Desks is that more than a year later, 9 out of 10 of them were used more than once per week.

What is the difference? Mostly they are used for sending messages to other people within or outside the organisation using the electronic mail system: that is, they are used for communication.

Secondly, researchers reported by Infoworld estimate that less than 30% of all personal computers are equipped with modems, yet business users overwhelmingly name communications as the capability they would most like to improve in their computing system. The communications technology will clearly be added soon.

So there are two indications that the future of information technology in the office in its widest sense is bound up with communications, and that communications facilities are needed by office and information workers as these social trends evolve.

2.1.2 Technological factors: Figure 1 represents one of the many technologically driven features which have shown exponential trends since the 1940s. It could be the cost of a given unit of processing power, as in Grosch's law, the acreage required for a single transistor as in Moore's law, or as in this case the decreasing number of relative failures per gate as technology evolves from transistors, through SSI and LSI to VLSI.

During the forty years shown here, these changes – in price, in size, in reliability and in processing power of electronic components – have taken information technology from a back room speciality into a pervasive technology central to any business, organisation, or country.

This technological capability is exploited for instance:

- Processing power: by 1995, a single integrated circuit selling for a few hundred dollars will deliver more computing power than that from a clutch of $5 million super computers today;
- Storage: memory chips with Megabyte capacity will sell for a few dollars, and disks will continue to develop higher and higher densities, decreasing costs of storage by 34% a year;
- Transmission Media: the throughput of conventional copper hasn't changed, but high-speed LANs using coaxial cable, optical fibre links with low loss rates, mean that the costs of networking are dominated now by building and laying costs rather than cable costs;
- Scanning and printing: the spread of non impact (e.g. laser) printing

Reliability & cost of technology

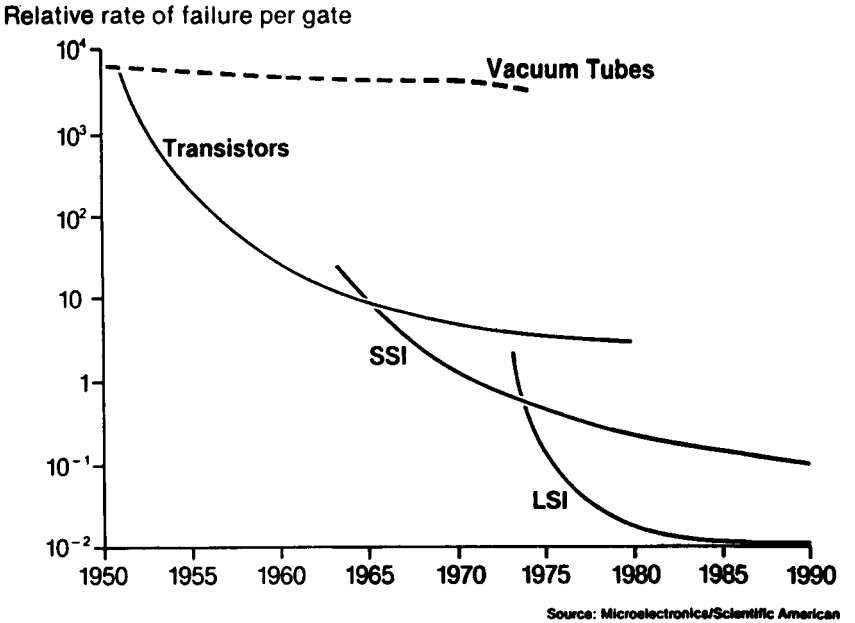


Fig. 1

technology opens possibilities for colour and image printing which are yet to be exploited. The use of technology for scanning of documents on input, using OCR with assistance from image processing technology to replace keyboarding, is in early stages of development.

These projects suggest that the basic technology and components will be there to support users' ambitions: but will the systems be there to support their ambitions? That is, will our systems support the following:

- mixing of text and data in a meaningful way
- integration of voice mail and conventional electronic mail
- speech expression of text
- command recognition
- editable graphics
- access to remote services for data

The X.400 and ODA standards discussed below tackle the first two areas of application.

2.2 Electronic mail

Many applications like spreadsheet or wordprocessing work well in a standalone environment, though results or documents may well need to be

shared. Electronic mail, on the other hand, is a classic example of an application which is singularly pointless without other users. So before the advent of IT standards, the simplest way for organisations to achieve an electronic mail environment was to build proprietary in-house systems.

Very quickly, however, it became apparent that the cost of supporting these was far too high. Each required specific training for users and system engineers alike. Every time extra facilities like security or distribution lists were required, they had to be specifically designed and developed.

Then there was the need to communicate between different systems within the organisation. Even when the mainframe, minis and PCs had the same origins it was difficult enough, and certainly costly.

Once this was solved there was the business of gearing up to be able to communicate with customers, suppliers, bankers and commercial collaborators worldwide. The need to communicate effectively was growing rapidly. With the speed of commerce increasing, the opportunity to speak to colleagues or customers was decreasing. Time zone differences, even within countries like the USA, postal delays and travelling time were rapidly becoming incompatible with effective management.

The only real answer was a single international standard that would enable manufacturers to create systems able to communicate naturally between private domains. Such standards resolve many of the problems experienced with proprietary electronic mail systems in private domains.

One of the remarkable aspects of X.400[7], as the standards have become known, is the sheer speed with which the industry moved to implement them. Once European Computer Manufacturers' Association, (ECMA), and CCITT had established a common understanding, things moved very quickly indeed.

The standards were first published in 1983. It was a mere 18 months before the world's first commercial system was demonstrated at ICL's Networked Office launch at Bracknell in April 1985. In fact ICL had been working closely with CCITT, ECMA, and ISO to develop the standard and had a head start. By the time of the Hanover Fair in March of 1987, fourteen manufacturers and network suppliers including DEC, HP, Bull, XEROX, Olivetti and ICL were able to demonstrate multi-vendor electronic mail in action, based on product offers.

IBM also has announced availability of an X.400 compatible electronic mail product, by the end of 1988.

2.3 Supporting protocols

2.3.1 Wide area networks: Of course X.400 is only part of the solution.

Occupying levels 5–7 of the OSI model, it does not deal with actually getting the information from one place to another.

Increasingly, to create cost effective wide area networks, X.25 is being employed. Traditionally this packet switching protocol has been used by large scale public networks such as national PTTs and high-value private networks, such as those operated by banks. This was because both the intelligent switching and the encoding/decoding units required were expensive.

However, the ability of packet switching networks to reroute data automatically to avoid line failures or traffic congestion and their low cost of transmission, were ideal for electronic mail systems. There are other advantages too. Because messages are encapsulated in identical packages, X.25 provides a high standard of multi-vendor capability. Intelligent switches check for errors and correct en-route, so corruption is minimised.

X.25 technology is now less expensive and looks very attractive as a carrier for wide area X.400 electronic mail systems.

2.3.2 Terminals: There is clearly a need for terminals to recognise and be recognised by other unlike applications and systems. The standard called Virtual Terminal Protocol (VTP) links terminals logically to the system. It describes how they should recognise information presented by other systems and ensures that it is displayed coherently, despite differences in MMI.

VTP is becoming important because with X.25, X.400 and other standards a greater volume of information is moving between unlike systems. With PCs everywhere it is imperative that they be connected to the network irrespective of manufacture. Previously, proprietary protocols like IBM's 3270 or ICL's CO3 were used, which limited the type of terminals that could be connected. But VTP also offers another major advantage. By standardising terminal protocols, it reduces the communication overhead in protocols such as X.25, making them more effective.

2.3.3 ISDN: Based on a set of standards recommended by CCITT Integrated Services Data Network (ISDN)[8] will permit data, text, graphics, and voice to share the same network infrastructure. The type of terminal used will be the sole determinant of the network's function, which means that a facsimile machine, PC, or telephone can be plugged into the same socket. A split bus structure will allow more than one terminal to be used at the same time; thus, a user interrogating a database will be able to discuss the information on the screen with a colleague on the other side of the building or on the other side of the world.

ISDN has recently been endorsed as a European standard. The twelve countries that comprise the European Community have agreed to introduce a single version of ISDN: the agreement is in the form of a 'recommendation',

which was recently passed by the European Parliament. It sets deadlines for the specification and implementation of a limited number of ISDN services. The idea is to encourage member countries to spend, collectively, $6 to $7 billion on ISDN by 1993 in the hope that demand will then take off for what would be a universal set of data, text, voice, and image transmission services – all accessible over, and supported by, the same ISDN access wires.

Phase One of the program calls for all EEC countries to provide, by the end of 1988, a circuit-switched 64-kbit/s 'bearer' service that is transparent to user traffic.

By the same deadline, four basic applications, or 'teleservices', are to be accessible through, and supported by, the 64-kbits/bearer channel:

- basic voice-grade analog-channel capability
- CCITT Group IV facsimile transmission
- Teletex
- A mixed-mode Teletex and facsimile transmission service/capability

In addition to bearer channel support for these services, a number of 'supplementary' services are also to be provided or supported. These are call waiting, caller identification, direct dial-in, and closed user group facility. Phase one further calls for the universal provision by year-end 1988 of adapters for existing analog, X.21, and X.25 customer terminal equipment.

The provision of ISDN services in Europe clearly opens the gates to the exchange of information in a variety of forms, across national boundaries.

2.4 Office Document Architecture

Office Document Architecture[9] probably represents the most significant standards initiative of all those in the office arena. Designed to allow images in both raster and vector form, data, text, voice and video to be communicated between systems from different vendors, it is a very powerful set of standards. By describing information logically both in content and format, ODA ensures it is communication in a form that can be edited, manipulated and used in other applications.

The fallback standard is ASCII. This 7 bit code carried no logical information about the data carried, merely identifying each character space and what goes in it. So if communications were taking place between a traditional 80 character landscape screen and an A4 portrait word processing screen, a large proportion of the data would have nowhere to go and would simply be lost.

By defining logical structures for information, ODA allows the reformatting necessary to achieve a different layout automatically, as shown diagrammatically in Fig. 2.

EFFECTIVE COMMUNICATION REQUIRES UNDERSTANDING OF STRUCTURE AS WELL AS CONTENT

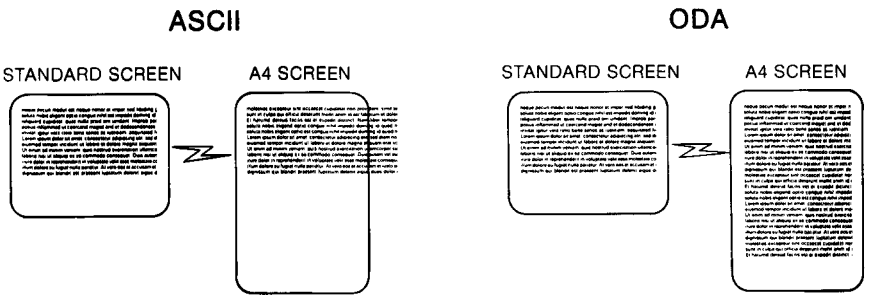


Fig. 2

ODA divides into two parts, descriptions of document content, and document structure.

Under document content, we have today standards for transmitting character text and photographic (or facsimile) images. They will be extended to cover:

graphics – both image forms and forms capable of being processed. These will include constructions formed from elements such as lines, circles and ellipses, and extensions are expected to handle composite constructs formed from such elements;

data capable of being processed (typically numeric data and computable expressions);

sound, especially for spoken annotation of documents.

Under document structure, standards are currently available covering three applications:

imaging: the transmission of a text image in such a way that what the recipient receives is exactly the same as the original;

formatting: the transmission of a text image in such a way that the recipient can re-format the text for his own purposes (for example, to merge the whole or part of it into one of his own documents);

editing: includes all the editing controls in the original, thus supporting amendment of the document by the recipient and automatic revision of the page layout accordingly.

This standard also caters for the inclusion of application-specific information structures: this will be done through use of a specially developed language called SGML (for Standard Generalised Mark-up Language).

Future standards will provide for filing and retrieval, and for particular constructions using multiple modes of data, such as spreadsheets and business graphics.

The status of ODA is that facilities ratified by ECMA in 1985 formed the basis of the ISO and CCITT work. CCITT has an existing standard – teletext – which is a subset of ODA – and is looking at issuing an incremental standard to cover ODA facilities. ISO have issued a DP based on the ECMA standard, and ratification is expected in 1988.

3 Exploitation of standards

3.1 Communication of information

All the evidence points to the fact that the inability of existing systems such as telephone, traditional mail, telex and secretarial services to match the speed and flexibility of today's commercial environment, is proving to be a major source of aggravation in the 'information age'.

A number of technologies and services have sprung up to fill the gap, such as couriers and telephone answering machines. They can only be regarded as stop-gap in nature however. Communications within the office are just as problematical. Historically one could rely on meeting frequently enough in corridors to ensure that colleagues were kept up to date. Now, with devolution of business functions to remote sites, this is no longer the case. Managers are working faster, longer and are more mobile.

Traditional secretarial services falter when managers do much of their administrative work out of office hours or at home. It becomes more difficult to telephone busy executives, even more difficult to arrange a meeting between several.

What's more, the emphasis in organisations of all types is increasingly on management. Most companies that need them have payroll, accounts, invoicing, and other clerical tasks computerised. Many have production processes, retail systems, stock control, distribution and other industry specific routines well served by departmental systems. Yet very few today have comparable levels of IT support for what is probably the most important part of any company.

White collar information workers of all types, from secretaries to professionals and senior executives, in sales, marketing, finance and personnel, need better tools with which to tackle their tasks. They have historically been undercapitalised compared with other workers, and handicapped by inadequate services.

The failure of the PC to help in this situation suggests that the ability to perform tasks more rapidly is only part of the solution. The problem is

communications based. The solution must meet both communications and applications needs equally.

3.2 Information is 'multi-media'

We are already seeing the first manifestations of a new generation of communication tools. Imaging systems, mobile telephones, radio paging, facsimile, electronic mail and viewdata, all tackle discrete elements of the problem. A lasting solution will clearly depend on convergence of the technologies.

But few want to make the first move. Pioneers in any area are notorious for making the headlines, then meeting the official receiver. What everybody wants is the reassurance that when they make a decision about office systems, it will be in line with whatever the industry as a whole decides. Choosing the best system is simply not enough. It also has to be compatible with systems adopted by other departments or companies in the group and with those of customers, suppliers and collaborators.

3.3 de facto standards

One way of ensuring this is to buy the same systems throughout the organisation as those used by business partners and customers. But, even within the same organisation, history usually dictates that several systems will need to co-exist, and specialist requirements will dictate that new systems may be supplied from more than one source.

Another approach is to depend on systems which depend on de facto standards. These will naturally be of a proprietary nature and future systems planning will naturally be dictated by the commercial interests of the individual computer manufacturer. Product ranges, operating systems and specialist applications will be developed or not, according to that vendor's perception of market needs. And the interworking standards will be under private, rather than public, change control.

Most users today find this an unacceptable constraint on their corporate sovereignty. With information, especially computer based information, rapidly being recognised as any organisation's most valuable resource, the freedom to choose how it is exploited is paramount.

Freedom of choice means alternative suppliers, specialist services, specialised applications and a variety of approaches to any problem. Only international standards, under public change control and adhered to by the majority of relevant suppliers, can create an environment in which users can control their own destiny in this way. Only international standards can ensure the degree of integration between the different communication media required.

3.4 Standards as a facilitator for niche and new technology

Several factors make inevitable the emergence of international standards as the single most important development for IT this decade. One such is the rapidity with which computing technology is being developed, it is relevant to a wider range of applications almost every day. It is impossible for any manufacturer to meet every market opportunity, nor is it desirable.

There is therefore an almost infinite number of openings for dedicated specialist vendors to meet new demands. But inevitably, isolated solutions have limited markets, limited futures. Each tends to fulfil its true potential only as part of a larger system. But which, and from which manufacturer? International standards remove the need to answer such questions in detail, by ensuring compatibility with a wide range of systems.

By encouraging new and independent suppliers the standards will help produce more relevant systems, which will encourage market growth. This will in turn encourage systems integrators to develop more comprehensive, complete and usable systems which will also encourage market growth.

By removing the system planning dilemmas common to users and manufacturers alike, standards will speed the arrival of new systems. And by encouraging free competition and reducing development costs, they will cause the price of systems to fall. This too will fuel market expansion.

3.5 OSI and obsolescence

There are other advantages for users. No vendor, however large, can meet a user's entire need. So the ability to mix and match systems from the market as a whole is an attractive option, allowing the users to get the best value for money and the best tool for the job in each area. The risk of incompatibility can be minimised, allowing users to develop systems at their own rate to meet very specific needs.

By increasing systems integration, international standards will minimise the number of terminals on the office worker's desk. They will eventually ensure that all services, whether image, text, data or voice based, are available through a single workstation.

Another phenomenon, IT obsolescence, will be dramatically reduced. Re-use of systems in new, often unpremediated situations will be made far easier to achieve. And systems will be generally more easily reconfigurable to meet new demands, new initiatives or reorganisations.

For instance, the FOCUS Private Sector Users Committee on Standards[10], has enumerated a number of potential benefits. The conclusions of their Report are that open, multivendor standards facilitate:

- Communications within an office and between (internal) offices.
- Communication with other offices and organisations.
- Minimum number of terminals on an office desk.
- Planned migration paths to new equipment, with minimal hardware and software obsolescence.
- Unplanned migration paths which may be needed – through takeover, reorganisation, commercial or other reasons – are easier to accommodate.
- Ability to 'mix and match' equipment from different suppliers, so as to get the best value for money for each item of equipment.

It is clearly impracticable to quote universally applicable savings figures but the FOCUS committee estimated that, in roughly quantified terms, this could mean a 15% saving in overall users' costs.

But to regard such savings as the achievement of the standards initiative would be to miss the point entirely. The real benefits lie in the exponential rise in the value of information owned by organisations, and their greatly enhanced ability to use it effectively.

4 Conclusion

Good management is the key to success for any business. But good management needs accurate, timely information, responsible administrative backups and effective communications. That, quite simply, is what office systems exist to provide, not in isolation but in conjunction with a variety of other application specific systems.

As each new generation of office systems delivers more facilities, handles more information faster, so its relevance to other specialist systems increases. The result is such natural integration into the fabric of the organisation that people come to rely totally on the services provided. The organisation's future and its office systems become inextricably entwined.

During the early stages of an industry's development, time is clearly of the essence. But as our industry matures – and the evidence for the increasing maturity of our industry is the technology curves we looked at earlier – the demands of users for the availability of interworking between kit from different suppliers strengthens the push for standards compared with the strength of the push for ad hoc solutions or proprietary protocols.

The concerted multivendor approach to Open Standards is called Open Systems Interconnection (OSI). The OSI route offers in addition to the above:

- It makes the standards offerings and direction openly visible: no fine print reserving the right to change at the whim of a vendor.
- It allows no arbitrary impositions of direction: all debate and change control are in the public domain.

Open Systems Interconnection Standards offer the freedom to choose vendors, systems, and approaches according to specific needs. They enable users to plan systems and information strategies with confidence. And confidence is the key to the market explosion in office systems.

References

1 MAP: 'Manufacturing Automation Protocol', General Motors Corporation, ADMES A/MD-39, GM Technical Center, Warren, MI 48090–9040.
2 TOP: 'Technical Office Protocol', Robinson P.J., ICL Technical Journal, November 1987.
3 US GOSIP: 'Government Open Systems Interconnection Profile', FIPS PUB draft, April 1987.
4 CEC Information Architecture: Document IX/E-6(86) S10-1140 provides a preamble to the Architecture requirements for 1986–1991.
5 SPAG: 'Guide to the Use of Standards', Standards Promotion and Application Group, North Holland, 1986.
6 UK GOSIP: 'UK Government OSI Profile', Working Paper 3: CCTA, February 1987.
7 X.400: 'X.400, the post facilitator', Elliott D.M., ICL Technical Journal, November 1987.
8 ISDN: 'ISDN on Trial', S. Underwood, Datamition, February 1987.
9 ODA: 'ODA Document Standards', Campbell-Grant I.M., ICL Technical Journal, November 1987.
10 FOCUS Private Sector Users Committee, 'The Benefits of Standards', Department of Trade & Industry Report, 1984.

Introducing ODA

Ian Campbell-Grant
Manager, Advanced Products Sector, ICL Office Systems, Bracknell, Berks

Abstract

The paper gives an introduction and outline guide to the international standard ISO DIS 8613, "Office Documentation Architecture (ODA) and Interchange Format". The aims, general concepts and key principles of this standard are described, and its relation to ICL's previously produced Normalised Documentation Format (NDF) discussed; there is a note on future developments.

Introducing ODA

What is ODA?

ODA, the 'Office Document Architecture' is an international standard (ISO 8613[1]) which represents the latest, and probably the most significant, advance towards a standard means for integrating office systems since the inception of the Open Systems Interconnect (OSI) initiative. ODA is an interchange standard for multi-media documents which has been produced in order to allow such documents to be exchanged between conforming computer systems anywhere in the world.

One feature of the ODA standard is that it allows the document to be presented to the recipient with the same layout as that prepared by the originator. More importantly, because the ODA definition provides for the logical structures of the information to be exchanged, the documents can be edited or reformatted or the information contained can be used within other applications.

Recent OSI standards – notably File Transfer Access and Management (FTAM) and X.400 (electronic mail) have dramatically increased our ability to transfer data between unlike systems. This has highlighted the need for the originator and recipient to have a common understanding of the semantics of this data, in order to provide for *information transfer*, so that the information can be understood, can be manipulated and can be re-used by the recipient.

Sitting above the communication standards in the ISO Open Systems Interconnection model, and employing them, ODA provides the basis for such information transfer. ODA documents can contain information content

represented in the form of character text, raster graphics and geometric graphics. Extensions to the standard are planned to add ODA information content types for computable data and sound, allowing these types of information to be incorporated within such documents.

In addition, ODA provides for the various pieces of information content in the document to have their layout and logical inter-relationships represented. By agreement on a generic document type definition, ODA can be used between any pair of applications which understand this particular type of document. In particular, ODA defines standard rules for editing and formatting applications, so these become standard applications.

ODA is designed to support the needs of a wide variety of different cultures, including the western, Arabic and Japanese requirements. Thus, using the ODA standard electronic documents can be used and transferred worldwide, being imaged (e.g. printed or displayed) or processed (e.g. edited or reformatted) by the recipient according to the intentions of the originator. For example, a document can be imaged without any need for the recipient to first format or reformat the received document.

Not surprisingly, ODA has been enthusiastically received by manufacturers. It's early days yet, as ODA has some way to go in gaining wide user exposure, but the signs are that it is destined to be the basis of the open systems information architecture which is needed for true office automation.

The 1986 DTI report, 'Profiting from Office Automation: the way forward' revealed that lack of product compatibility was of major concern to managers. The importance of achieving complete document (or spreadsheet, or graphics) interchange was highlighted by Roger Pye, who put together the DTI report. He says:

> "Standards are dull but essential. They should mean that all documents are sendable, readable and modifiable in any software setup."

It is this order of compatibility that ODA sets out to provide.

Who benefits?

The ability to communicate between a diverse range of office products, and to edit, amend and process information easily at either end, will bring huge advantages both to users and to manufacturers.

The user gains time. A document can be edited on the receiving system without, as was formerly the case, having to re-format, re-paginate and re-set margins or tabs following amendments. As any secretary will testify, this represents a considerable saving in time and effort.

But there are other benefits, like lower transmission overheads. Text

designed for printing on pre-printed stationery can be sent without the human operator being required to format the information, as would otherwise be necessary. And it offers easy laser printing. The requisite 'stationery information' can be sent with the text, eliminating the need for complicated printer set-up procedures.

Editing is also faster and easier. A modern, easy-to-use system such as an ICL DRS Office System can present the structure of an ODA document, such as a company report, at the user interface. Editing to this standard can then be accomplished by means of menu-driven prompts, so that the process becomes fast and foolproof.

The manufacturer gains too, because ODA's information architecture can be used as a framework for developing a set of office system products such that each can communicate with, and take advantage of, the others' facilities. This saves much reinvention of wheels and reduces overheads.

Thus, manufacturers and system integrators can specialise, confident in the knowledge that their products will integrate within larger systems. ODA also enhances the potential of existing office products. No radical revision of office product architecture is necessary to ensure compatibility with a variety of different systems. All that is necessary is that products have been designed to provide ODA compatibility.

Of course all users, suppliers, vendors, systems integrators, manufacturers or end users, benefit from the security of a single, open standard which guarantees multi-vendor interworking and which is under public change control. It encourages investment in new equipment, unhampered by incompatibility caution. Conformance to the ODA standard will link today's purchase with the products of the future.

The standards bodies

The main standards bodies involved in the development of ODA are ISO, CCITT and ECMA; these are introduced here as they are referred to extensively below.

ISO (the International Standards Organisation) coordinates the work of the various national standards bodies, including BSI for the UK, AFNOR for France, DIN for Germany, ANSI for US and establishes International Standards by agreement between its member bodies.

CCITT (the International Telegraphic and Telephone Consultative Committee), which is part of the UN structure and is the focus for standards for the world's PTTs, is responsible for international telecommunications standards. CCITT is primarily concerned with communications between systems operated by the national carriers. Many of the CCITT standards also get passed to ISO.

Recently in a number of areas, including ODA and message handling systems, CCITT and ISO have been working in very close collaboration to establish equivalent standards.

ECMA (the European Computer Manufacturers' Association) is a trade association representing and producing standards primarily for the computer industry in Europe. A primary aim of much work in ECMA is to assist the rapid development of ISO standards for Open Systems Interworking. ECMA ratifies its own standards too. Although ECMA standards are often identical to those of ISO, they can usually be introduced more quickly.

ECMA is the body that initiated work on ODA and has been working to assist ISO and CCITT to complete the ODA standard.

The Office Document Architecture standard

The International Standards Organisation, from its Geneva headquarters, and the CCITT, operating from headquarters, also in Geneva and neighbouring those of ISO, are jointly formulating the definitive ODA standard.

ODA will be documented as an international standard by the International Standards Organisation in its specification IS 8613 and by the CCITT in its T.410 series of Recommendations[2]. Both of these are planned to be ratified in 1988 and to be exactly equivalent specifications. ECMA-101[3], published in 1985 and forming the first version of the ODA standard will then be updated to fully align with the ISO and CCITT specifications.

For communication purposes, the ODA standard falls within the seventh layer of the ISO reference model for Open Systems Interconnection (OSI). ODA can be conveniently divided into three types of standards:

- document structure standards
- document content standards
- document distribution standards

This division is followed below in examining the ODA standard in more detail.

Document structure standards

Document structure standards provide a general document architecture which:

- governs the interrelationship of a number of different forms of information;
- provides a standard means of expressing specialist concepts and information structures for particular applications such as document editing, formatting, or filing and retrieval – such specialist concepts and structures

include for example, identation, columnar layout definition or retrieval keywords;
- provides a standardised means of describing the structure of generic document types, such that groupings of information appropriate to a particular application, or to a group of applications, can be modelled – in this case ODA can be used between applications which understand a particular document type.

Document content standards

Document content standards enable information to take the following forms:

- character text;
- facsimile images (raster graphics);
- diagrams (geometric graphics).

In addition, extensions to ODA are planned to incorporate:

- computable data (used to support spreadsheet or business graphics);
- sound.

ODA governs the definition and means of manipulating of the document structures and of each of these forms of information within electronic documents.

Document distribution standards

For the case of distribution of documents using Open Systems Interconnection a number of standards are defined to specify means for accessing and transmitting documents, as a whole or in part. These standards cover:

- the encoding standards for documents or parts of documents in terms of an abstract syntax notation.
- remote access to whole documents.
- partial document transfer, such as only the completed fields in a form.
- interactive access to documents.

ODA and existing communications standards

In order to facilitate interworking with standards which are already in use, ODA encompasses some existing standards, by defining them as subsets of ODA.

In order to understand how ODA relates to other communications standards, it is useful to bear in mind the division into standards for document content, structure and distribution.

Document content and structure

October 1984 saw the ratification by CCITT of a standard which made provision for the transmission of a document to include mixed media and imaging control information (CCITT Recommendation T.73). This enables the recipient to image the text with the same layout as the original. The information content types included the CCITT standards for character text and raster graphics content architecture.

The current ISO/CCITT Office Document Architecture standard, includes the T.73 specification as a subset. The ODA character content architecture is a superset of the CCITTs, providing some additional processable character text interchange facilities.

Both the ISO and CCITT raster graphics content architecture include levels which are fully aligned with the CCITT recommendation T.4 and T.6, known as 'Group 3 Facsimile' and 'Group 4 Facsimile'. These levels are defined subsets of ODA.

The character content information type is aligned with that of 'Teletex' (CCITT Recommendation T.62).

Because of these relations to pre-existing standards, the construction of convertors between ODA and telex, Teletex and facsimile will be substantially simplified.

Document distribution

In the case that ODA is used for transfer of documents in OSI systems, ODA is regarded as forming part of the ISO Open Systems application layer. For this purpose, ODA is designed to use levels 1 to 6 of the ISO seven layer model for communications. It has in common with both file transfer (FTAM) and electronic mail (X.400) the use of Abstract Syntax Notation One (ASN.1) in the presentation layer. ODA can use either FTAM or X.400 as a transport system. At a lower level still, ODA can be carried by any transparent communications protocol, such as X.25.

ICL's Normalised Document Format (NDF) and ODA

ICL has long been committed to the concept of ODA and in 1982, recognising both the importance of document interchange and the long timescales to which international standards bodies necessarily work, developed an architecture which formed an early intercept of the work on ODA. This, Normalised Document Format (NDF)[4], made possible the interchange of documents between office products in the ICL product range, and in fact became the kernel from which the international standards have grown.

NDF, first developed in response to a need to interchange documents

between 8800 wordprocessors and 2900 series mainframes, facilitates imaging and editing by the recipient in line with the intentions of the originator. The level of interchange offered is impressive but ICL recognises that NDF is nevertheless a proprietary standard restricting interchange to ICL systems; so it is intended to extend the system to achieve full conformity with the ODA standard, with the aim that users shall be able to connect NDF systems to ODA-compatible systems, retaining NDF characteristics over ODA networks.

ICL has played a leading role throughout the development of the ODA standard, providing technical specialists for the ECMA, ISO and CCITT technical committees and the current chairman of the ECMA Technical Committee TC 29 on 'Document Architecture and Interchange' and the editors of several of the key standards documents, including the ISO 'Future developments of ODA'.

Market acceptance

Aircraft manufacturers Boeing are promoting the Technical Office Protocol (TOP) as the way forward to fully integrated office systems. Using selected standards from each layer of the ISO seven layer Open Systems Interconnection model, TOP seeks to provide a comprehensive model for interworking, from physical connection, through transport to the application levels. TOP enables one set of information to be shared by, for example, the chief designer using word processing software, and the technical drawing office, using a CAD/CAM facility.

TOP, an increasingly important standards initiative in its own right, utilises ODA based document interchange[5]. So ODA compatible systems will also be TOP compatible.

The enthusiastic support shown by the TOP initiative for ODA demonstrates the urgent need for such comprehensive interworking standards. ODA's key role in TOP highlights just how many crucial interworking issues are addressed by the standard. A host of collaborators like Ford, Kodak and Dupont in the TOP initiative leave little room for doubt about market acceptance.

ODA standards: who is doing what

Document structure standards

The standard specifies a document architecture covering three applications: imaging, formatting and editing. The standard provides for the control information required for each application to be transmitted with the document.

- For the imaging application – text can be imaged by the recipient with the same layout as prepared by the originator;

- For the formatting application – text can be reformatted (for example to merge the whole or part of it into another document);
- For the editing application – the document can be amended, keeping within the structure rules specified by the originator for the particular class of document and the text layout can be automatically revised accordingly.

Parts of the ISO standard	Individual CCITT Recommendations
DIS 8613/1: Introduction and general principles	Draft Recommendation T.411: Introduction and general principles
DIS 8613/2: Document Structures	Draft Recommendation T.412: Document Structures
DIS 8613/4: Document Profile	Draft Recommendation T.414: Document Profile
DIS 8613/5: Office Document Interchange Format	Draft Recommendation T.415: Office Document Interchange Format

Document content standards

The standard specifies three content architectures covering character text, raster graphics and geometric graphics. The document structure standards allow for documents to have mixed types of document content, all types are able to be mixed and mutually positioned within a single document.

Parts of the ISO standard	Individual CCITT Recommendations
DIS 8613/6: Character Content Architecture	Draft Recommendation T.416: Character Content Architecture
DIS 8613/7: Raster Graphics Content Architecture	Draft Recommendation T.417: Raster Graphics Content Architecture
DIS 8613/8: Geometric Graphics Content Architecture	Draft Recommendation T.418: Geometric Graphics Content Architecture

How does ODA work?

ODA is a software coding method, which allows documents generated on any office system to be converted to a common form known as Office Document Interchange Format (ODIF). Following transmission in this format, the document is re-converted to an appropriate form at the receiving system. The conversion software is system-specific, so that manufacturers can design their own converters; in order to achieve document interchange it is not necessary that the rest of the design of the system be ODA compatible.

Document structure

ODA supports two parallel and interconnected views of document architecture. The logical structure relates the content of a document to objects such as paragraphs, headings and footnotes. The layout structure relates content to objects such as type fonts and pagination. This structural model is very precise and detailed and is the key to ODA interworking.

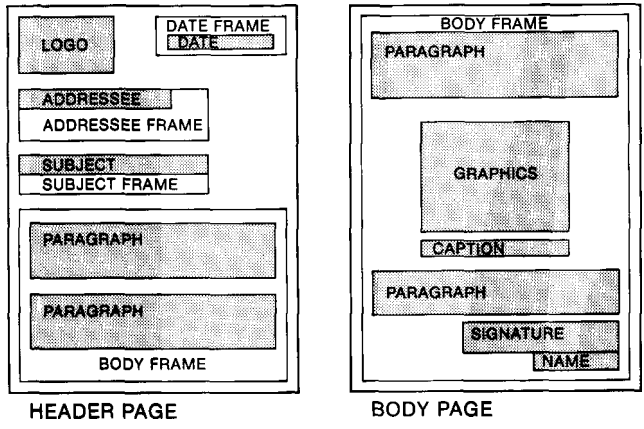
The rules for structuring a document are described by grouping logical or layout objects into 'class descriptions'. So a series of paragraphs (logical objects) forming a section of a document can be described by a logical class description for a paragraph. In the same way, a group of layout objects such as pages with the same column layout and fonts can be described by a layout class description for such a page. Objects are grouped in this way to simplify the creation of documents, improve transmission efficiency and ensure that the internal document structure is maintained following editing.

ODA offers a high degree of flexibility, allowing different rules to be developed for any type of document. So ODA does not prescribe document structures, but provides a framework within which to create them.

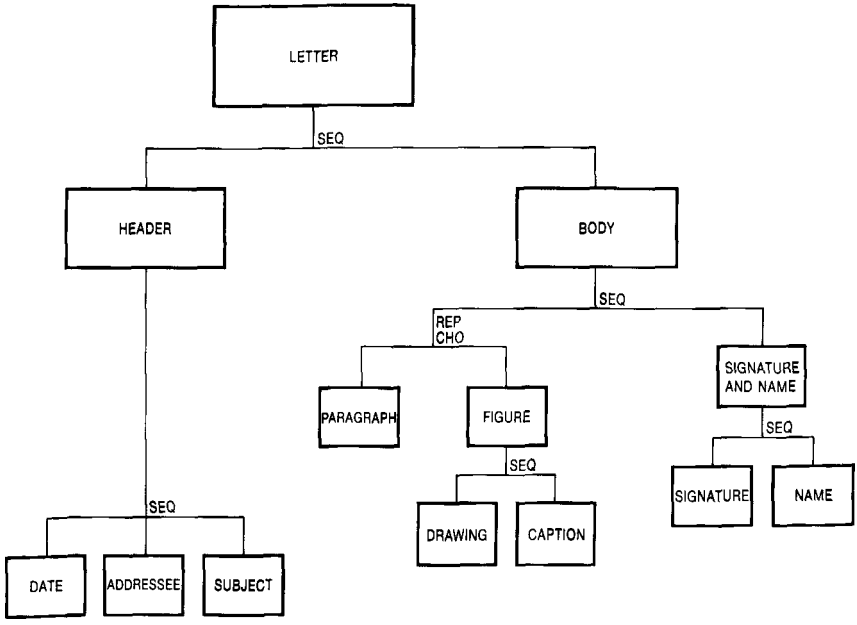
The relationship between logical and layout structures is recorded in a 'layout directive'. Layout directives are important because they allow designers to specify rules for layout, for example that if a document is edited each chapter should always start on a new page, or that the footnotes be at the bottom of the pages.

A second type of relationship is recorded in a 'presentation style'. Presentation styles are used to specify rules for presentation aspects of information,

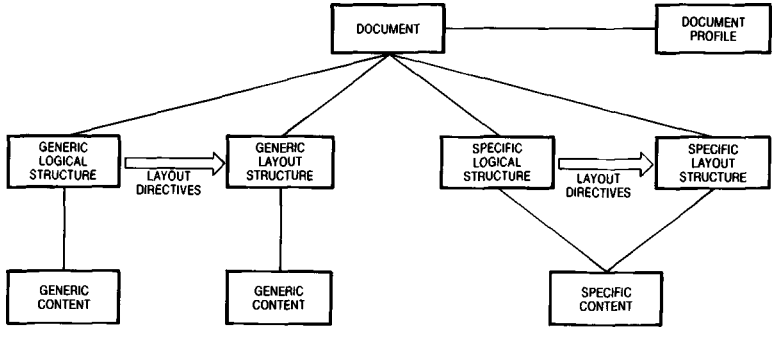
ODA
LAYOUT STRUCTURE



ODA
GENERIC LOGICAL STRUCTURE



ODA STRUCTURE



for example the font which is to be used, or how character text is to be positioned with respect to tabulation points.

Each document must carry with it a document profile. This is designed to hold attributes which relate to the document as a whole, such as its title, author and copyright notices. The profile also specifies whether the document contains specifications for layout or logical structure, or their class

descriptions. It may additionally incorporate information relating to the document's history and details needed for indexing and filing.

Systems design requirements

In product terms, a document processor is needed to coordinate the editing of the various information content types; one editor is used to control editing of the overall document structures and to coordinate a series of other editors, one for each of the forms of information.

To enable ODA to manage the presentation of the document in its original form by the recipient's system and his subsequent editing of it, three types of facility are needed:

- The document editing facility provides for a document to be created or modified. It is required to include both content editors, to cope with creation or amendment of individual pieces of document content such as the content of a figure or a paragraph, and a logical structure editor to cope with changes such as inserting a new figure or deleting a paragraph and substituting two more.
- The document layout facility provides for the layout of a document to be generated, it allows systems to cope with problems such as splitting paragraphs across pages. This process allocates the logical structure to the physical page layout, using the results of the editing facility as input.
- The document presentation facility uses the layout structure and layout class descriptions to define how the document is to be imaged.

Where is ODA today?

With some ODA standards at draft stage, and others already ratified, commitment to using the ODA standards is growing.

ODA demonstrated at Hanover

Despite the fact that the ISO standard has not yet achieved full International Standard status, visitors to this year's CeBIT Hanover fair were able to see ODA prototype systems being demonstrated. In this international collaboration, Britain's ICL, France's Bull, Italy's Olivetti and Germany's Siemens were able to interchange documents between very different word processing systems.

This widely acclaimed demonstration was based on the use of ODA as an interchange format between the word processing systems of the four companies. The WP systems used were independent products of the various companies, not varieties of the same generic product, and the interworking demonstration was achieved without any modifications to the products involved.

ISO 8613 ratification

The ISO sub-committee 18 'Text and Office Systems', which has two working groups focused on the development of document and content architectures respectively, will meet again in November 1987. This meeting is planned to complete ISO 8613 defining ODA.

In February 1988, CCITT plans to ratify the T.410 series of Recommendations which parallel this ISO work, as well as the application profiles allowing facsimile and interactive Videotex to use the ODA standards. In addition, CCITT also plan to agree some further Recommendations, including some complementary distribution standards.

The future for ODA

All three of the main standards bodies (ISO, ECMA and CCITT) have ambitious plans to extend the ODA standards over the next three to four years[6,7]. They will continue to work together within the same area making sure that their standards are aligned before being ratified. All these extensions are being developed so as to be purely additive extensions to ODA, not amending any existing features. The standards planned fall into the ODA areas of document content, structure and distribution.

Document content

Plans to develop a unique method to image colour, independently of the content architecture, are receiving high priority attention, and may well fully complete the standardisation process by mid-1989. This will incorporate provision of grey scale for monochrome imaging.

In the longer term, standards for audio content architecture, designed particularly for the inclusion of spoken annotations in documents will be developed. A new dynamic graphics architecture will enable the processing of moving images. These two new content architectures will take the concept of document interworking into new realms of sophistication.

Structure

High priority is being given to development of rules to govern the transfer of computable data, derived from calculations, to form part of the logical structure of a document. This will assist in the interworking of document processing and data processing applications.

Such support for data will provide for forms, which will allow for the interchange of documents containing objects representing data fields, which may be defined as formatted or processable and which control aspects such as permissible field content and size.

The presentation of data will use existing information content types and data will be able to be reflected either textually, as tables or in forms such as spreadsheets, or graphically, including provision of a business graphics level of facility.

Several other extensions to the standard are also planned, to cater for the efficient interchange of as wide a range of documents as possible. These extensions include:

- providing a general means for ODA to be used for application integration, such that multiple applications can identify and manipulate relevant parts of ODA documents, and such that ODA documents can form the basis for further automation of office procedures;
- providing a high level of facility for the processing of tables and tabluar material;
- supporting multiple sets of annotations and allowing annotations to be optionally imaged with the document;
- providing for different security attributes for designated parts of documents, such as the generic document definition or for signatures within documents;
- allowing documents to include by reference all or parts of other ODA documents;
- extending layout facilities to allow representation of all features associated with high facility print formats, or page description languages.

Distribution

Existing media and file standards will be used to facilitate a new standard governing document interchange on physical storage media such as floppy discs. This is being given high priority and may fully complete the standardisation process by mid-1989.

Also being treated as high priority are application layers protocols for document distribution. These will allow both partial document transfer, and remote interactive access to documents. CCITT may complete ratification of some of these aspects in 1988.

Conclusion

That there is considerable commitment to ODA, from both the manufacturers and the standards bodies, is beyond question. ODA is the first Open Systems standard to provide an information architecture, a framework for developing products which not only communicate but which offer full interworking. ODA opens the way for a new generation of fully integrated office automation systems.

The environment in the more technically advanced societies is moving into one in which information is the commodity, telecommunications is the

means of distribution and the application and interpretation of the information is the value added. Manufacturers and users alike who do not make use of ODA standard will forfeit competitive advantages.

In order to protect IT investments against erosion by rapidly advancing technology, manufacturers and users both need stable interworking standards, changed only by common consent. ODA not only provides this but also combines design freedom with maximum functionality and flexibility: the plans for ODA's future development show that these will be even further enhanced.

References

1 ISO DIS 8613, Parts 1.2, 2.2, 4.2, 5.2, 6.2, 7, 8, 'Office Document Architecture (ODA) and Interchange Format'.
2 CCITT Draft Recommendations T.400, T.411, T.412, T.414, T.415, T.416, T.417, T.418, form the series of Recommendations 'Document Transfer Access and Manipulation (DTAM)'.
3 ECMA-101, 'Office Document Architecture'.
4 PSD 1060, 'NDF 0, Subset one'.
5 'TOP Specification version 3.0 – Implementation release – April 1987' – MAP/TOP users group.
6 'Future developments of ODA, Issue 2', ISO/TC97/SC18/WG3, editor I.R. Campbell-Grant.
7 'Extensions to ODA', ECMA TC29.

The Technical and Office Protocols – TOP

P. J. Robinson
Advanced Products Sector, Office Information Business Centre, Office Information Systems Division, Bracknell, Berks.

Abstract

Today the biggest obstacle to full exploitation of information within organisations is the number of different computer systems with incompatible protocols. Even different systems from the same vendor often have problems in communicating. OSI standards for communication exist, the question is which to choose and whether they can be met commercially. By a selective use of OSI and other standards, TOP aims to provide a framework which allows comprehensive office systems to be built from off-the-shelf components, economically.

Introduction

The Technical and Office Protocols (TOP) is a set of standard communication protocols for specifying multi-vendor distributed information systems used for business and technical operations. Dedicated to improving the effectiveness of office based information systems, TOP has its roots firmly embedded in the manufacturing industry.

It is accepted today that there is more to manufacturing than just making things. Successful manufacturers are as concerned with research, design, marketing and sales management as they are with production processes.

Each discipline uses different information and information in different ways. Yet most information is relevant to other departments, and often it is duplicated.

Often it is input many times, on different systems throughout the organisation. The more often it is input, the greater the opportunity for error and inconsistency. Most information changes with time, and controlling change is a major issue with manufacturers of all types.

Manufacturing processes are increasingly complex, and the rate at which products and markets change is escalating. For large combines the problem is particularly acute. With large numbers of departments each depending on the others for information, the potential for losing, delaying or corrupting data is almost infinite.

Manufacturing application protocols

In 1984, General Motors began an initiative to enable integrated multi-vendor information systems to be created around manufacturing processes. Manufacturing Application Protocol (MAP), as this standardisation initiative was called, concentrated on allowing information from drafting or design systems to interwork with automated production systems. Robotics, process control, stock control and a range of other systems were destined to interwork fully, despite their different vendor origins.

The concentration was natural in an organisation dedicated to high volume automobile production. Aircraft production however, has different priorities, different problems. Aircraft are still basically hand crafted in low volumes. Many of the processes involved are reliant on human skills. But the number of components required for an airliner can run into millions. Audit trails for each must be kept, and the results of the slightest error in handling the vast quantity of information required can have tragic consequences.

A Boeing 747 requires over 250 000 pages of information for users and maintenance engineers. Customers can decide the way they want information presented and in what language. By any standards, document production is a big issue. Add to that the sophistication of the aviation market, the complex R & D, design, pricing and sales organisations required and it becomes obvious why the Boeing Aircraft Corporation decided to initiate a parallel and complementary standards initiative to MAP, for the office.

Technical and Office protocols

Called the Technical and Office Protocols (TOP) programme, the initiative followed MAP's lead by forming groups of collaborators with similar interest. The decision was taken early to work within the developing Open Systems Interconnect standards as far as possible and make all standards produced non proprietary.

At the time the only practical way of creating networked systems was to use components from a single or a very limited number of vendors. If a range of different vendors' systems were required to create a solution, it involved developing custom hardware and software. This clearly adversely affected cost effectiveness and often reliability too.

Users were under pressure to choose systems for their connectivity to existing systems rather than for fitness-for-purpose. This led to 'islands of automation' which were expensive and difficult to integrate.

OSI standards offered a chance to work towards full integration of office systems. The problem was that, not designed to deal with office automation they were too broad, too complex and insufficiently specific to be a complete solution. It was simply too expensive to build full OSI compatibility into all

products. Even if it were technically possible, which in itself was in doubt, it would create serious delays in product development and would be commercially unsupportable.

It became clear that it was possible to identify subsets of the OSI standards which would allow Office Systems networks to be created, without requiring component systems to conform to every standard. These centred around levels 5, 6 and 7, dealing with session, presentation and application respectively.

What was important was that if subsets were to be used, they should meet the majority need for such systems and be agreed and clearly identified. Only then could vendors develop specialist sub-systems which would be compatible within larger networks, and users be guaranteed interworking.

Under Boeing's leadership, the TOP Users Group was formed in 1985. Its first meeting hosted over 200 representatives of organisations worldwide. After extensive seminars and working sessions a structure for TOP technical subcommittees was defined, and vendors and users alike encouraged to participate.

From those committees has sprung the concept of standards pillars, or stacks, within OSI. Each level is addressed with respect to office systems requirements and relevant standards identified. At the lower levels 1–4 dealing with datalinks, networking and transport, the key standards include X.25[1], CSMA/CD CSMA/CD[2], Token Ring[3] and Token Bus[4].

At levels 5–7 the emphasis is on information interchange by electronic mail[5] and file transfer[6]. Above these layers the emphasis is on interchange between applications – for example between document processing systems using Office Document Interchange Format (ODIF)[7] or graphics systems using Computer Graphics Metafile Interchange Format (CGMIF)[8].

In addition, protocols to ensure that terminals display information to standard formats (VTP)[9] are included. Services like Network Management, Directory and Remote File Transfer are all included in the upper levels.

TOP building blocks

Representing a functional specification for discrete subsystems within the TOP standards, the building block approach simplifies the choices faced by system designers and users. It provides for the specification of a commonly used function which may use several different standards across more than one OSI layer.

For example a TOP application system may incorporate the

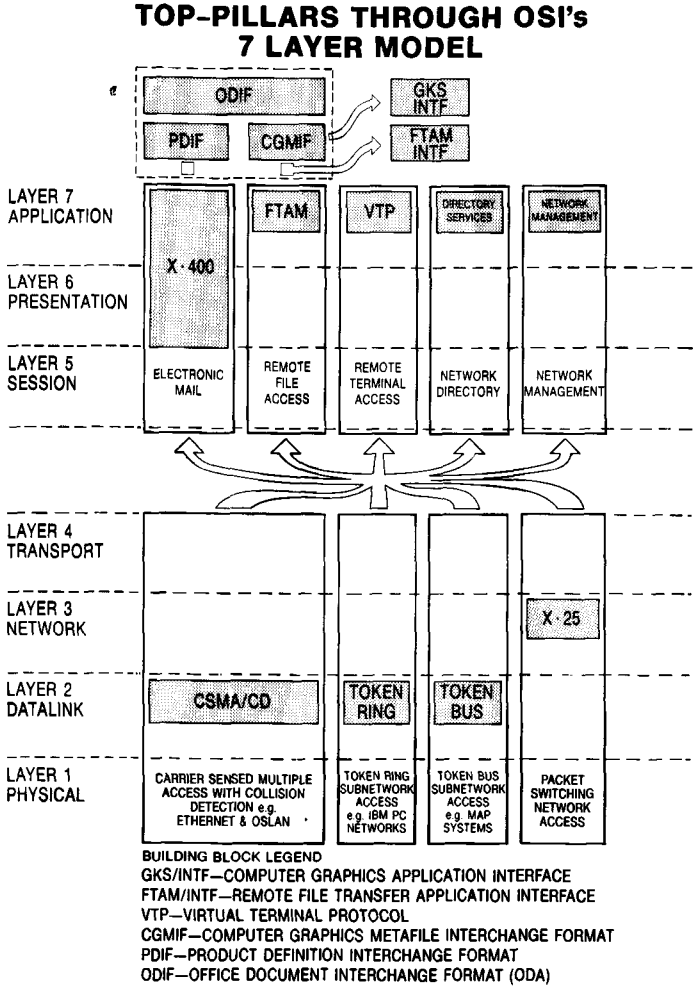


Fig. 1

CSMA/CD Subnetwork block, the Remote File Access block and the Computer Graphics Metafile Interchange Format (CGMIF) block.

Blocks are described by three attributes:

1 *Function* – what the building block will accomplish on a TOP system for the user.
2 *Specification Reference* – a reference to an international standard together with a set of selected options and parameters based on established implementation agreements for Open Systems Interconnection Protocols.

3 *Binding Rules* – the technical constraints that define the associations of the building block with other building blocks in forming valid TOP combinations.

The particular set of options and parameter values of an international standard defined for use in a TOP conforming system is termed an Application Profile (AP).

It is worth examining two of the TOP building blocks in more detail. These are the blocks which include Office Document Interchange Format and the Electronic Mail Building Block.

Office Document Interchange Format Building Block (ODIF)

This provides a common format and encoding for transfer of compound office documents in revisable or formatted form containing characters, geometric graphics and raster graphics. The TOP AP for ODIF:

- enables users to interchange documents such as memos, letters and reports
- refers to the ISO 8613 standard and subsets the full capabilities of this by specifying values for the document architecture, the content architecture levels such as character, geometric graphics and raster graphics subsets, the document profile and the interchange format
- can be used in conjunction with the electronic mail and remote file access building blocks.

Electronic Mail Building Block

This provides TOP-conforming systems with the capability for store-and-forward handling of messages between end users and applications. It offers to the electronic mail user the functions necessary to post and receive interpersonal messages.

Messages may be ASCII text or any of the interchange formats for computer graphics, product definition or office documents. TOP systems supporting this building block can be used as message transfer agents within a multi-vendor private domain or as one of a set of private message handling domains.

The TOP AP for electronic mail requires only the support of the mandatory and optional essential service elements of the Message Transfer Service and the Interpersonal Messaging Service of the CCITT X.400, the interpersonal messages that can be sent/received using TOP electronic mail include simple lines of ASCII text; telex; computer graphics using CGMIF; and ODA office documents using ODIF.

Electronic mail may be used in conjunction with the following:

- Computer Graphics Metafile Interchange Format Building Block
- Product Definition Interchange Format[10] Building Block
- Office Document Interchange Format Building Block.

Electronic mail must be used in conjunction with at least one of the following:

- CSMA/CD Subnetwork Access Building Block.
- Token Passing Ring Subnetwork Access Building Block.
- X.25 Packet Switching Subnetwork Access Building Block.

Other building blocks include:

Remote Terminal Access Building Block

This provides for communications between different terminals and applications based on host computers. It supports four modes of terminal working – asynchronous, scroll mode with local echo control, paging/scrolling with local printer and CCITT X.3 packet assembly/disassembly.

Remote File Access Building Block

This allows TOP application systems to access or manage files held on unlike systems, remotely. It likewise provides other systems with the capability to access locally held TOP application files.

Computer Graphics Metafile Interchange Format Building Block

This allows graphic images to be generated, manipulated and communicated between TOP compatible systems.

Product Definition Interchange Format Building Block

This provides a common format for transfer of data required for analysis, design, manufacture and testing required over the life cycle of products.

Network Directory Building Block

Enabling TOP systems to access remote network directories, this block allows users to refer to applications by their names. It also allows the directory to retrieve current application addresses.

Network Management Building Block

Remote application and server systems can be managed using this block. It also enables configuration, fault and performance management.

X.25, Token Ring, MAP Token Bus and CSMA/CD Building Blocks

X.25 is familiar as the packet switching protocol used in many wide area networks employing intelligent switches to route communications. It offers en-route error correction, automatic re-routing to avoid congestion and line breaks, and a low cost of transmission.

Token Ring provides access to local area subnetworks of systems such as IBM PCSs. Token Bus provides access to MAP systems and CSMA/CD allows Ethernet systems to be connected.

TOP user applications

The number of office based tasks which can be aided by computers is increasing almost daily. In defining what is and what is not office automation, definitions necessarily become blurred. In fact the impossibility of precisely defining such segregations serves to underline the need for integration of information within organisations.

What is clear is that TOP does not address Manufacturing Automation, process control or robotics. Many office functions are common to most commercial and government organisation. TOP focuses on these, which include:

- Electronic mail
- Word processing
- Text/Graphics
- Database Access
- File transfer
- Distributed CAD/CAM
- Spreadsheet Exchange
- Banking Transactions
- Electronic Funds transfer
- Distributed Manufacturing Business Systems
- Peripherals and resource sharing

although not all of these are separately defined functions within TOP.

The area of distributed CAD/CAM is on the key area of overlap with the MAP initiative. MAP took information from CAD/CAM applications and applied it to numerical control systems, robotics and factory automation. For TOP users the data created by CAD/CAM applications can be compatible with all the associated office applications. The information from the drawing office can be taken through buying departments, research and development, senior management, sales and marketing. Once entered it is available to all office workers, unconstrained by technologies like wordprocessing, imaging or data processing.

Moreover, a specification can be taken from an engineering drawing and used as logical data for typed specification on ordering applications. With the links to the MAP standards, a vector curve from the same drawing could be used to drive production tools or potentially, to conduct sophisticated data searches.

The user group

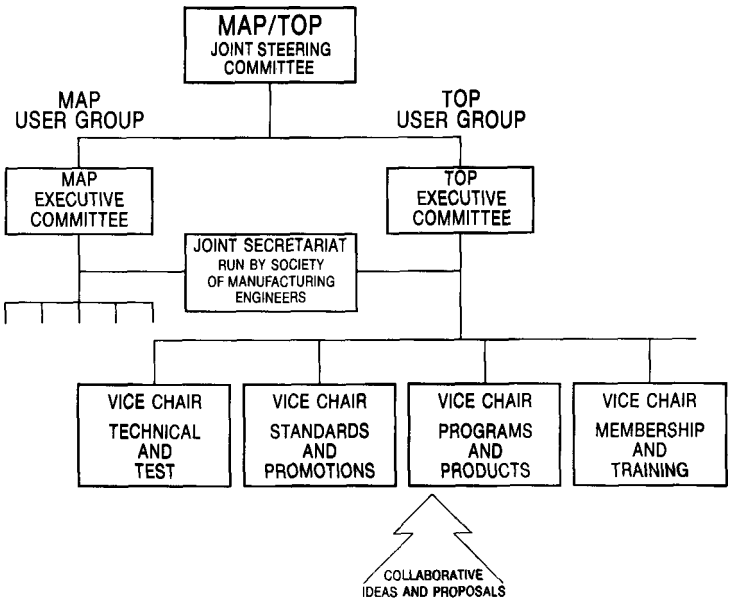


Fig. 2

Following the lead given by General Motors and the MAP initiative, TOP set up its Executive Committee to manage the technical planning, administration and promotion of the standards. Under the auspices of the Committee Chairman, four Vice Chairs each handle specific areas of the initiative.

The first is Membership and Training, which includes promotions and recruitment of corporate affiliates. The Vice Chair of Programs and Products monitors TOP compatible products, maintaining a current list of new and existing systems. This chair is also responsible for arranging agendas for TOP User Group meetings. The third Vice Chair is Standards and Associations, which monitors and liaises with other relevant standards bodies.

The Vice Chair of Technical and Test manages and structures the technical subcommittees. Responsible for monitoring vendor tests, this chair also identifies new projects and subcommittees.

The business of standards production, functional specifications, conformance

test specification and change control, is handled by the subcommittee structure. These are also the channels for collaborative ideas and proposals.

All Vice Chairs liaise directly with their opposite numbers in the MAP User Group. Both initiatives share a joint MAP/TOP steering committee, co-existing under an umbrella organisation which is able to ensure an equitable representation for all parties' interests.

The TOP User Group has established firm relationships with other standards bodies such as ISO, CCITT (International Telegraph and Telephone Consultative Committee), ANSI (American National Standards Institute), IEEE (Institute of Electrical and Electronics Engineers) and ECMA (European Computer Manufacturers Association). The SME (Society of Manufacturing Engineers) acts as the User Group's secretariat, administering meetings, agendas and disseminating information.

TOP today

There can be little doubt about the momentum behind TOP today. Members include well known companies like General Motors, Boeing, McDonnell Douglas, Proctor and Gamble, Eastman Kodak and in Europe, ICL and AEG. Their interest is based on solid commercial realities.

Non-proprietary international office system standards offer an opportunity to create better integrated information systems. The pay-off is in both better communication within and outside their organisations, and greater responsiveness to market opportunities. It frees users from dependence on any one supplier and creates an environment in which information system planning is simpler.

It is early days yet, but the first full functional specification called 'TOP V3.0' was released this year. In accordance with its policy of avoiding unnecessary differences with the MAP specification it is compatible with 'MAP V3.0'.

TOP is building on the work done by MAP and incorporates many OSI standards. Among these is the CCITT X.400 electronic mail and messaging standard and the Office Document Architecture (ODA) for the Office Document Interchange Format, already adopted by ECMA, ISO and CCITT.

Future developments

The TOP initiative plans to sponsor an Enterprise Network Event in 1988. It will demonstrate the interworking capabilities of TOP compatible systems. This event is still in the early stages of planning. However, the capabilities of TOP could be demonstrated by an aircraft component design, complete with documentation, being created between the design offices of Boeing and McDonnell Douglas. Designs could then be transmitted electronically to

BAC in the UK, where they could be manufactured. Finally, the part would be flown back to the US to be built into an aircraft.

An integral part of ICL's open systems

Clearly any company selling in today's office systems market must conform to OSI standards. Few users are prepared to tolerate a future locked into a single IT vendor, unable to communicate freely with their customers or suppliers.

As is the case with all IT suppliers, for ICL the question is selecting which standards to concentrate on. This is the real relevance of the TOP initiative. By creating a set of priority standards and conformance specifications agreed between vendors and users, the process of adoption can be greatly speeded and simplified. The faster and simpler the adoption process, the more vendors will be encouraged to conform and the lower the cost to users.

In 1986 ICL became the first European IT systems supplier to declare its support for TOP. ICL had of course played a leading role in the establishment and development of the OSI standards. Whilst this was primarily a European based initiative, the US has been quick to pick up the standards baton, giving it fresh impetus and relevance. ICL's role in TOP is a recognition of the importance of broadening the standards movement worldwide.

Another significant TOP characteristic is that like MAP it is user-led. Many of its key members are not computer system vendors, but users. They have sizeable IT environments and development programmes of their own. All recognise that the early adoption of innovative products into their system and the ability to communicate and interwork between systems is crucial in maintaining a competitive edge.

For ICL, TOP offers an opportunity to become a major participant in the newly emerging worldwide electronic communications structure. The ICL/STC relationship clearly has much to gain from a resolution of the computer and telecommunications industries into a single coherent force.

Perhaps the single most important achievement of the TOP initiative is to remove any doubt about the future of international standards under public change control. The amazing rapidity with which some of the world's largest computer users and vendors are developing and adopting these office systems standards has disabused any illusions about the future of proprietary standards.

Clearly it is becoming increasingly difficult for users to justify purchases of major systems that are out of step with the international standards movement.

References

1 CCITT Recommendation X.25 (1984), Interface between Data Terminal Equipment and Data Circuit-Terminating Equipment for Terminals Operating in the Packet Mode and Connected to Public Data Networks by Dedicated Circuit.

2 IEEE 802.3; Local Area Networks – Carrier Sensed Multiple Access With Collision Detection (CSMA/CD) Method and Physical Layer Specification [ISO 802/3].

3 IEEE 802.5; Local Area Networks – Token Passing Ring Access Method and Physical Layer Specification [ISO 802/5].

4 IEEE 802.4; Local Area Networks – Token Bus Access Method and Physical Layer Specification [ISO 802.4].

5 CCITT X.400 (Red Book, 1984), Message Handling Systems: System Model – Service Elements.

6 ISO DIS 8571 (August 1986), Information Processing – Open Systems Interconnection – File Transfer Access and Management.

7 ISO DIS 8613 (June 1987), Information Processing – Text and Office Systems – Office Document Architecture (ODA) and Interchange Format.

8 ISO DIS 8632 (1987), Information Processing – Computer Graphics – Metafile for the Storage and Transfer of Picture Description Information (CGM).

9 ISO 9041, Information Processing Systems – Open Systems Interconnection – Virtual Terminal Protocol – Basic Class.

10 IGES V3.0, ANS Y14.26 M-1987, American National Standards.

X.400 – International information distribution

Dorothy M. Elliott
Electronic Mail Marketing Manager, Office Information Business Centre, ICL, Bracknell, Berks.

Abstract

The evolution and acceptance of the X.400 international standard for electronic mail has progressed more rapidly than might have been expected. This paper examines some of the influences on the progress path and describes the ICL role and approach.

1 Introduction

The pace of commercial life in the 1980s has vividly illuminated the flaws in traditional communications. The most rapid, like the telephone, require both parties to be present, in possession of the facts and free to talk. The cheapest, like the post, are slow, unreliable and insecure. Even telex, so rapid between operators, somehow fails between mailroom and desk.

Now a new medium, electronic mail, offers a service compatible with today's communication needs. Able to transmit text and data of any length as documents, letters or messages almost instantly anywhere in the world: it is a remarkably powerful facility.

But to work, it is essential that all users and carriers conform to international standards. Only then can mail travel almost at the speed of light through public and private networks, beating time zone differences by store-and-forward techniques and verify its arrival to the sender. Only when otherwise incompatible computer systems conform, can electronic mail span the multitude of different terminals, mainframes and minis in use worldwide.

X.400 focuses this requirement into a single set of internationally agreed standards which enable this to become a reality.

2 History

X.400 comprises a series of recommendations, developed between 1980 and 1984 by the International Telegraph and Telephone Consultative Committee (CCITT). Ratified in 1984, they define a set of standards governing message handling systems (MHS).

The X.400 standard represents a natural progression of the work undertaken by CCITT earlier this century, developing international telephony standards. By specifying telephone network construction and how messages should be addressed, transmitted and switched, they laid the foundations for today's international service.

We now take it for granted that we can speak to anyone in the world by picking up a telephone and dialling a number. X.400 is designed to create an environment in which, using electronic text, communications will be just as natural, just as easy.

Work began on the X.400 series of recommendations to ensure Open Systems Interconnection for electronic mail. Until X.400 most electronic mail systems were incompatible with one another. With organisations building larger networks, often installing a range of manufacturers' equipment, integrating office systems was becoming increasingly difficult. Islands of electronic text messaging communities, each using different equipment or services, were isolated from one another. X.400 was designed to enable all the different mail and messaging systems already in place to be linked.

A number of issues had to be tackled to ensure that full international electronic mail could become a reality. It had to be possible to:

- link equipment which may have been bought at different times from different manufacturers
- connect independently purchased personal computers into central office automation systems
- link with other electronic mail systems of customers, suppliers and collaborators
- connect into public electronic mail networks in different countries
- provide store and forward facilities to cope with time differences, manage the different transmission speeds of various systems and ensure that when an addressee's equipment is temporarily unavailable, mail is not lost.
- establish simple and universal addressing techniques
- provide protocols to monitor the arrival and acceptance of mail.

The key to the compatibility of any private or public MHS defined by X.400 lies in the addressing and routing system which X.400 provides. Users of an X.400-compatible system each have a personal User Agent (UA) attached to Message Transfer Agent (MTA). One MTA can service several UAs on the same system. A mail item is submitted to the system via a UA which passes it, including the recipients' addresses, to the MTA.

The address of a recipient is coded hierarchically by country, carrier, organisation, department and personal name. The MTA analyses the address in order to route the mail item to the mailbox of the appropriate UA.

If the mail item is addressed to a recipient on a different system, the MTA

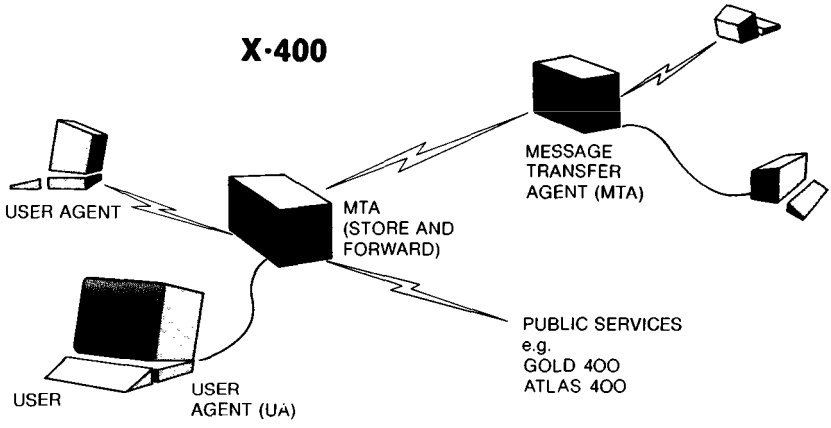


Fig. 1

time- and date-stamps it before passing it on to the next MTA on the route, as specified by the routing tables. That MTA will, in turn, route it on to another MTA on a store-and-forward basis, until the MTA serving the recipient UA is reached. At each stage of its journey, the mail item contains a record of its route through the system. If, at any stage, the mail item requires different routes to reach different recipients, then it is split, with each copy identifying just those recipients for which the next MTA is responsible for routing further.

3 The making of the standard

Any standard, if it is to be relevant and accepted by the majority of suppliers, requires lengthy consultation processes. Those affected must have a say in the formulation. Although a slow process, it is the only way to provide the stability and security required for users and the IT industry to make long-term plans and investments.

The recognised authority for X.400 is CCITT, which primarily represents the PTTs who will provide an integrated, world-wide, public electronic mail service. CCITT did take soundings from computer manufacturers during the process of developing the X.400 standard. ICL, in particular, contributes technical specialists to CCITT working parties.

Because the X.400 standards are complex, implementing them is both time-consuming and costly. And because X.400 allows for options within the standards, achieving cross-range compatibility between different equipment is not always easy. One manufacturer may interpret the standard in a slightly different way to another, so while the products of both are X.400-compatible, they are not necessarily compatible with each other.

Several bodies in different parts of the world have addressed the issue of X.400 complexity, to make life easier for the X.400 implementors. In Europe,

a group of twelve leading IT suppliers, including ICL, formed the Standards Promotion and Application Group (SPAG). SPAG realised that Functional Standards were needed in order to provide a common X.400 interpretation. They pioneered 'profiles' which define a consistent implementation for the message handling function. Other groups have since adopted the profile approach and more bodies have become involved in the process of ratifying the standard.

In January 1985, three companies, including ICL, defined the SICOB profile and this was developed by SPAG and issued as a SPAG Purple Profile in the Guide to the Use of Standards (GUS) in July 1985. Since SPAG's objective was to have these adopted as European norms, they were fed into the European Committee for Standardisation (CEN/CENELEC), a European Commission initiative.

Using the profile technique and incorporating the SPAG profile, CEN/CENELEC issued their profile in February 1986. This became a pre-standard in May 1986 and a mandatory requirement for public procurement.

The European Conference of Postal and Telecommunications Administrations (CEPT) have also issued their own profile. CEPT is a grouping of European Post, Telegraph and Telephone authorities (PTTs) – among them British Telecom – who ensure that national telephone and data transmission services interwork. CEPT harmonises the services offered by the European PTTs, building upon the CCITT recommendations. With electronic mail, their profile covers the interworking between PTTs and also between a PTT's public messaging system and a private messaging system.

In mid 1985, the North American computer companies set up their SPAG equivalent, the Corporation for Open Systems (COS). Significantly ICL was the first European computer manufacturer to join this group. In January 1986 COS announced that they would be setting up test laboratories and tools for CCITT conformance validation.

At Berne, conformance test services have also been set up under COMTEX-LAB, part of the SWISS PTT COMTEX project. ICL is one of the active testers, contributing to the specification of the tests, particularly to the P2 test suite. COMTEX-LAB are targetting for a Message Handling Test Service in 1988.

In the UK, British Telecom have played an important role in making X.400 available as a public message carrier. In January 1986, they announced a £5 million plan to set up a nationally managed network service to carry X.400 messages. Additionally BT plans to incorporate telex, teletex and facsimile facilities in their service, launched as GOLD400. So users of X.400 compatible systems should also gain access to these facilities worldwide. TELEPROVE, British Telecom's test house, is offering a new testing service to serve the growing X.400 market.

Throughout the 1984–88 work study period, CCITT have been reviewing extensions and enhancements to their X.400 recommendations of 1984. With co-operation for 'common text' between CCITT and the International Organization for Standardization (ISO), the 1988 recommendations have yet to be ratified by these two groups, but guaranteed interworking capability between implementations completed to these and to the '84 recommendations provide a broader and sensible platform upon which manufacturers can base their decision as to when to incorporate extensions such as

interconnection to the postal system
distribution list facilities
secure messaging
message store access (MSA) and management access of MSA

into their existing X.400 offerings.

Thus from 1980–88, we see a period which has introduced a standard for inter-personal messaging. Many groups have made significant contributions to its improvement and understanding, culminating in formal refinements and extensions in an agreed and cooperative manner. Let us now look at the reaction of the suppliers.

4 The acceptance of the standard

ICL was one of three companies to take part in the world's first public demonstration of multi-vendor X.400 interworking at the SICOB exhibition in Paris in 1985. Since then, X.400 has been moving fast. At Hanover Fair, in 1987, fourteen major international companies proved that the age of electronic mail for offices has arrived by demonstrating

- communication between public and private electronic mail systems
- direct communications between different private electronic mail systems
- international relaying via multiple public systems

The companies taking part at Hanover were:

British Telecom	Bull
Data General	Deutsche Bundespost
Digital	Hewlett Packard
ICL	NTT
Nixdorf	Olivetti
Philips	Siemens
Sydney	Xerox

Although notably absent from the Hanover demonstration itself, IBM have announced their intention to introduce an X.400 compatible message handling system late in 1988. This should allow direct connection between SNA-based DISOSS (Distributed Office Support System) and other X.400 Systems.

Also at Telecom '87 in Geneva, a large group of administrations and service providers from Europe, Japan and North America demonstrated their X.400 interconnection ability.

The overall commitment of major service providers and suppliers all over the world to X.400 significant not only in implicit agreement, but in terms of practical and commercial reality in such a short timespan indicates that this fundamental part of the electronic office was eagerly awaited and has been put quickly in place. Therefore the tasks in future relate not only to the evolution of the standard but to its exploitation in terms of increasing benefits to offices and staff.

5 ICLMAIL

X.400 is expensive. Because of its complexity as a standard, understanding and actually implementing it can be a major investment for any organisation. It can take up to three years, and just testing the combinations in one area of X.400 can take a considerable time. So, although the benefits it offers are invaluable, many vendors were understandably reluctant to invest in X.400 in 1984, even though committed to it in principle.

ICL, acknowledging the crucial importance of OSI, launched ICLMAIL in 1985. This employs central and distributed servers based on VME and DRX as Message Transfer Agents (MTAs). Electronic mail access points were made available incrementally from a range of workstations, including PCs, OPDs, DRS20s, DRS300s, word processors and "glass teletype" devices. Currently, it is available on ICL's UNIX-based office systems, with integrated mail access from the OFFICEPOWER and OFFICEPOWER-PARTNER software systems. This means that ICL VME mainframes and DRS and CLAN distributed systems can all interwork using X.400; and that all users, whether clerical, secretarial, professional or managerial, grouped in areas, departments or corporately, can intercommunicate effectively both nationally and internationally through electronic mail.

This approach to X.400 concentrates on visual ergonomics and simplicity of use. After all, there is no value in having the capability to send documents around the world if one cannot remember how to do this; it has been said that there are two types of users of office computers, the naive and the sophisticated: the naive can remember three commands, the sophisticated, seven.

6 Summary and conclusion

X.400 as an internationally recognised and accepted standard has achieved remarkable success in the last three years. Major administrations and service providers have launched their X.400 service – major suppliers have produced end-user systems – conformance test centres have been set up.

In addition, it must be remembered that X.400 is not restricted to electronic mail, that is to inter-personal messaging. It offers a common Message Transfer service, which may be used to carry any sort of data. One of the next candidates for standardisation is the carriage of Electronic Business Data, such as invoices and orders.

However, X.400 will only achieve total success if it offers dividends for the end-user organisation. If the dividends are observable or quantifiable by whatever means deemed appropriate then the supplier and the standard will have been effective.

Companies and organisations are often more impressed and learn better from others with effective working models than they do from text-based standards. Clearly this cannot be achieved quickly; if the communication barriers of inherent cultural differences can be overcome with X.400, then there is a genuine advantage in offering key additional features. If that foundation can be achieved and confidence be given to all to participate then the richer basis of comparison may provide a more effective and stronger critical mass or base.

This threshold of opportunity brings the communication benefits of the electronic office into commercial reality and enables suppliers of X.400 systems to generate a positive and profitable impact on the business and organisation of offices worldwide.

References

X.400	CCITT RED BOOK Volume VIII – FASCICLE VIII.7 Data Communication Networks Message Handling Systems (October 1984).
CEN/CENELEC	ENV 41201 – Private Message Handling System: User Agent and Message Transfer Agent: Private Management Domain to Private Management Domain (June 1986).
CEPT	ENV 41202 – Message Handling Systems: User Agent and Message Transfer Agent accessing to an Administration Management Domain (July 1987).
GUS	Guide To The Use Of Standards (S.P.A.G. – 1986).

INTERFACE AND PROGRAMMING LANGUAGES

A general purpose natural language interface: design and application as a database front end

B.G.T. Lowden, A.N. De Roeck, D.J. Phipps, R. Turner
Department of Computer Science, University of Essex, Wivenhoe Park, Colchester CO4 3SQ

Abstract

The aim of this paper is to give an overview of the work carried out by Essex University, under ICL Grant UE2, on the design and development of a Natural Language Front End System (NLFES) based on formal semantics. The prototype system, currently implemented, incorporates an improved version of the REMIT paraphraser module initially developed under an earlier ICL Grant and described fully in [Lowden and De Roeck 1986a, 1986b].

1 Introduction and overall system design

Typically for most existing natural language (NL) query systems, the user can ask a question in a human language and the system reports on the question's interpretation with little or no user intervention in respect of the interpretation process. This type of system is subject to a number of criticisms. First of all, there is an underlying suspicion that NL systems, with or without paraphraser feedback, may induce a false sense of security. Because front ends of this kind can deal with some human language input, the user's threshold of what to expect becomes higher. He may become careless in how he formulates his queries and may ask questions that either the system, or indeed the database management system itself, cannot handle. If the user's needs are not central to the development and design of interfaces, then the value of the system itself becomes at best academic. Secondly casual users, in particular, are not helped to any great extent by simple NLFES for query evaluation unless they are also offered a number of other facilities which are at least as crucial. For example some way is needed for finding out exactly what sort of information the database contains. Thirdly, although paraphrases of the system's interpretation of a query may be a significant improvement, it is only a single example of the various types of feedback a casual user will find useful.

This paper deals with the design and implementation of a more complex type of interface to relational databases. The ELITE system (English Language Interpreter, Translator and Evaluator) is characterised by autonomy of the

natural language module which is not only database – or querylanguage – independent, but also independent of the task the NL input is to fulfil. The output of the NL component is the starting point for the other parts of the system, including a query system, a meta-query handler, a paraphraser and an error recovery module which monitors feedback to the user. The fact that the NL component is a general one, however, extends its use beyond processing database questions and makes it possible to augment this system with other components accessed through the medium of human language.

Central to the system described here lies the assumption that human language is a suitable medium for man-machine communication. By no means does this assertion preclude other means of interaction; it merely illustrates our point of view that interfaces including natural language facilities are useful by virtue of the fact that language is a form of communication with which people are familiar.

The main benefit of this approach is that it separates the NL component from its traditional role of merely serving as a processing channel for database queries. Thus NL expressions are mapped into a meaning representation sufficiently flexible to cope with the different input needs of the interface as a whole, i.e. whether it is a straightforward database query, a meta-query about the organisation of the database or a modal question about whether certain states of affairs are allowable within the database.

Equally important is the assumption that it is impossible to construct a unified interface of this kind without insight into the semantics of human languages. Most work on semantics has been done in the Montague tradition. Montague grammar typically analyses the meaning of a fragment of language by mapping it compositionally onto expressions in intensional logic. These expressions are then interpreted using the full power of higher order functions and the onthology of possible worlds. However, there are serious theoretical drawbacks to the use of Intensional Logic as a vehicle for semantic interpretation – particularly in computer based interfaces. First of all, intensional logic is computationally intractable. Secondly, its notion of intension is secondary and derived from a notion of extension. This leads to counterintuitive equivalences between meanings. These considerations led to the development of T Theory [Turner 1987] which retains the notions of compositionality and the division between structural and lexical aspects of meaning, but which does not use types. It also has a valid notion of inference for certain classes of expressions and regards intensions as primary notions, a feature which makes the theory intuitively more attractive.

In the ELITE system, the natural language component consists of a compositional mapping between a fragment of English and a language called PC-DET (Predicate Calculus with Determiners) which is a weaker version of T Theory. The compositional mapping is given by a set of context free grammar rules generating a fragment of English, each associated with a semantic translation rule. The PC-DET expressions (which are reducible to

first order) capture the structural build up of the meaning of the original expression, and they are the starting point for the other components of the system.

Since the NL component does not interpret the input in terms of a particular task which must be performed, it is possible to use it in conjunction with any module performing a duty which can support NL communication with the user. For instance, it could be positioned to the front of a query language tutor. However, the immediate aim of the project was to build a front end for database query evaluation. The present system, therefore, comprises (Fig. 1) a database query system which maps PC-DET expressions first into the relational calculus (to enhance portability) and then into Querymaster formulae which can be handled by the ICL DBMS; a modal meta-query handler which provides the user with feedback on which possible states of affairs are allowable within a particular database; a paraphraser which reports to the user how his question has been understood, offering him a choice in case of ambiguous input and finally a recovery module which intercepts those questions which the system cannot handle for any reason.

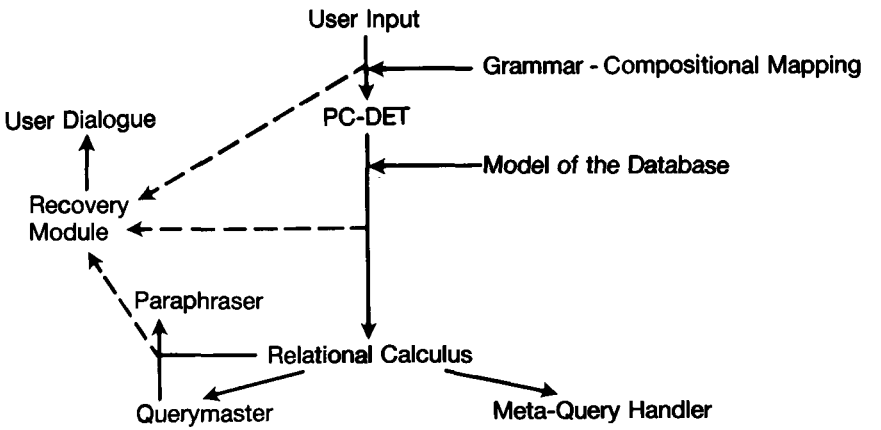


Fig. 1

2 NLFEs as a series of mappings

Writing a NL front end can conveniently be thought of as writing a program that will execute a mapping between a NL and a formal language suitable for querying a database. In this particular case, the NL has been identified as English, and the formal language as Querymaster [ICL 1985].

When defining the mapping one should remember that it must be able to extract from statements in English exactly that information which is appropriate for database interrogation. However, it is desirable that it also meets a number of other criteria. First of all, it would be appropriate if the mapping were not totally dependent upon a particular database. It is, we

believe, impossible to deliver front ends which are totally database independent. Still, it is perfectly feasible to restrict database dependency to isolated parts of the system which can be independently modified should the system be interfaced to a different datamodel. Furthermore, there are considerable advantages with systems that are portable across different querylanguages. Again, some aspects of the front end will have to be query language dependent, but it is possible to isolate that dependency within a specific component. Thirdly, it would be desirable if the system could be used not only to interrogate databases, but could also be adapted to interface with modules performing other sorts of tasks.

In trying to cater for the above portability criteria, the basic mapping between English and Querymaster devised for this system has been divided into four major stages. Each stage is characterised by specific aspects of dependency upon either of the input or output languages, or upon the database itself. In order to achieve this one must distinguish between those aspects of meaning which are universally valid and those which depend upon particular tasks to be performed.

In the present system that particular distinction is reflected in two consecutive mappings. First, English expressions are translated into First Order Predicate Calculus (FOPC) via an intermediary representation based on T Theory. The FOPC expressions reflect only those aspects of meaning of the input which are structurally determined on the basis of English syntax. They do not attempt to resolve word meaning as the precise way in which the content of words is to be interpreted must depend upon the context of use, i.e. the system one is interfacing to, on the one hand, and the domain it covers on the other (e.g. what "part" means will be different for an expert system concerned with aeroplane fault finding and for a supplier database). Lexical aspects of meaning are to be provided by a task and domain dependent model (in this case a datamodel over the ICL SCOPE database).

The resolution of word meaning will thus result in a representation which, in contrast to the FOPC, is no longer independent of these factors. In the present system this means that the FOPC representations will be translated, with the help of a model, into expressions of a typed (still first order) calculus sensitive to the use of the interface as a database front end and to the domain of customers and products as seen through SCOPE.

These two steps constitute the most difficult parts of the interface. They require insight into the semantics of English and necessitate the development of adequate models.

In the third stage, the typed expressions are mapped into the Relational Calculus. This part of the mapping is also sensitive to the sort of task the system will have to perform, in this case database interrogation. Furthermore, this intermediary step guarantees that the system can be easily adapted to work for different query languages. Only in the last stage, during which

Relational Calculus expressions are mapped into Querymaster statements, is the system linked to a particular query language. These last two mappings can be defined as pure syntactic transductions. As a consequence, their implementations have been relatively easy.

2.1 Overview of mappings and their implementation

Mapping English into FOPC Initially, statements in English (the input language) are mapped into expressions of the FOPC. There is a sense in which the FOPC is such an obvious candidate as an intermediary representation in systems of this kind that the choice needs no further motivation. The employment of FOPC as a representation language is widespread in Artificial Intelligence. It has become a basic tool in a wide variety of AI work and can be seen as the common denominator among an ever increasing range of computer applications. It has served as the basis for the design of programming languages and has the capacity of representing knowledge. Furthermore, it includes a valid notion of inference which means that, apart from providing a means of representing knowledge, it also defines formal and rigorous ways of arriving at new knowledge from known facts. It forms the base of a variety of representation mechanisms such as frames, scripts, semantic nets etc. Practically speaking, the choice of FOPC as an intermediary representation opens up a large number of possibilities in terms of the sort of system to which this part of the front end can be interfaced.

There are, however, basic difficulties in defining a mapping between natural languages and the FOPC. In particular there are problems in defining a simple compositional translation based on the syntax of both languages. Essentially the syntax of English (and indeed NL in general) differs considerably from that of the calculus, especially with respect to quantificational structure.

To overcome this it was necessary to devise a version of the Predicate Calculus, already referred to as PC-DET, that has a quantification structure resembling that of NL. We introduce into the FOPC two new syntactic classes. These classes are DETERMINERS and SET TERMS. They are designed to reflect the way in which NL quantification works. Nouns will be translated as set terms. NL quantifiers will be translated as determiners, which behave according to the principles of generalised quantifiers.

The translation between English and expressions of PC-DET is defined by means of a set of rules. Each rule has two parts: a context free grammar rule which will characterise a wellformed substring of English and a translation rule which allows for determining a subexpression in PC-DET which reflects the meaning of that substring. Since the translation rule only applies when the syntactic grammar rules does, the design guarantees that the mapping will be executed compositionally; this is in keeping with the traditions of both computational linguistics and computer science. The mechanism that executes the rules is a bi-directional chart parser [Steel and De Roeck 1987].

Explaining the differences between ordinary and bi-directional chart parsers and the advantages of the latter in terms of space and computing time would take us beyond the scope of this paper. Note, however, that the parser will produce all alternative solutions which can be detected at this stage. Syntactically ambiguous input strings will result in the production of more that one PC-DET expression.

The resulting PC-DET representation must then be translated into FOPC. The mapping at this stage is one to one (no additional ambiguity can be discovered) and is easily defined by a set of (compositional) translation rules. The implementation is fairly straightforward and takes the shape of a simple syntactic transduction.

The expressions now obtained do not include any reference to a particular domain as might be covered by a database, nor to any knowledge about the task on hand (i.e. interrogating databases). It would thus be possible to use this part of the mapping to "front end" other types of system as long as the task performed can be modelled by first order formalisms (e.g. inference engines, tutoring aids etc.).

Mapping FOPC into a typed first order calculus In mapping English into the FOPC, the resulting representation has been stripped of all structural syntactic information that characterises English. Nevertheless, the predicates in the expressions are just English words; this is in keeping with the distinction made earlier between structural and lexical aspects of meaning. In the second mapping to be described here, those predicates will be unpacked in terms of what they are taken to mean with respect to a particular application over a specific domain. This part of the process draws heavily on domain dependent information which has been cast as a "conceptual" model over the SCOPE database containing information about products, orders, stocks, warehouses and customers.

At this stage, the FOPC expressions are mapped into a typed first order predicate calculus, PC-TYPED. The typing was introduced because it offers a convenient (and traditional) way of distinguishing between different kinds of objects and relationships between objects in a domain of application. The translation process between FOPC and PC-TYPED expressions is sensitive to the fact that the necessary "conceptual" information must be cast into a relational datamodel. In this particular implementation, the domain knowledge is taken from a relational datamodel over SCOPE which was also used for the NEL system [West 1986]. Since the datamodel itself is not database or domain dependent (though its content is), it is easy to see how this part of the system may be adapted to work for any database for which a relational model can be defined. More importantly, relational datamodels can be devised for as many domains as may be addressed by different types of application. The fact that the system implemented here is intended to function as a front end to an actual database is coincidental. The NLFE itself can be interfaced to any system performing any task over any domain, for

which a relational model can be devised, by merely presenting it with such a model.

To provide all the information needed at this stage, the SCOPE datamodel must be supplemented by a dictionary which defines the relationship between English words (FOPC predicates) and constructs in the model. The current implementation relies on a dictionary which associates words with data-model objects and constructs of varying degrees of complexity. Some nouns (e.g. "client") may be associated with simple database relations or attributes whereas some verbs are taken to correspond to a variety of permitted access paths within the SCOPE model. Such a dictionary inevitably has to be constructed afresh for each application.

The FOPC expressions output by the first stage are explicitly structured to express the scope of quantifiers and logical connectives. They can be seen conveniently as right branching trees where the predicates, quantifiers, variables, connectives and constants occur as leaves. The translation into PC-TYPED, which must preserve scope of quantifiers and connectives, proceeds compositionally by recursive rule application which terminates at the leaf nodes of the given tree.

Taken in isolation, the translation of a predicate leaf node is simply the set of all objects associated with it in terms of a given database as defined by the dictionary. In the case of unary predicates which refer to simple database objects, the node is tagged with an interpretation type as assigned by the dictionary and the model. Hence the translation of **name(x)** may be:

{[[CUST–NAME(x)], attribute], [[PRODUCT–DESC(x)], attribute],
[[SUPPLIER–NAME(x)], attribute]}

In the case of n-ary predicates expressing relationships between items, leaf nodes translate to a set of primitive "meanings" compatible with the database as a whole. Thus the translation of the binary predicate **at(x, y)** may be:

{LOCATION(x, y) TIME(x, y)}

In both cases, therefore, translation of leaf nodes is simply a function of dictionary/datamodel look-up.

As the recursion unwinds, the rules for parent nodes are applied to their, now instantiated, leaf node arguments until, at the outer level, the rule for the root node (the whole expression) is finally applied. Having translated the leaf nodes, a given parent node in the parse tree now consists of the set of each possible interpretation of a given FOPC n-ary predicate over each member of the set comprised by the Cartesian product (A1 x A2 x ... x An) of the sets, A1 ... An, of the possible values of the n FOPC unary predicates to which the n variables in the n-ary predicate refer.

Translation of such nodes involves checks on the compatibility of the arguments with the predicate using the tagged database types together with checks, where appropriate, on whether a path generated by linking, in the case of attribute or value arguments, the owning relations or, in the case of relation arguments, the arguments themselves, would be permitted in the context of the given database. In the process of translation, therefore, all incompatible interpretations are eliminated. The resulting PC-TYPED formulae are expressions of database entities and paths between them cast in terms of equalities of key attributes.

As an example translation, consider the following query in English which relates to the SCOPE database.

Show all names at BRA01.

'BRA01' is a value of the attribute CODE in the WAREHOUSE relation. Note that this sentence is underspecified with respect to much of the information necessary to cast the question in formal terms. Translation to FOPC yields the tree:

$$(\exists \times 1(\times 1 = \text{you}) \wedge (\forall \times 2\ (\text{name}(\times 2) \wedge\ (\exists \times 3\ \text{bra01}(\times 3)\ \wedge\ \text{beat}(\times 2,\ \times 3))) \rightarrow \text{show}(\times 1,\ \times 2)))$$

where $\times n$ are variables. Mapped into PC-TYPED, the expression becomes:

$$\begin{aligned}
&\exists \times 12\ \text{PRODUCT}(\times 12) \wedge \forall \times 2\ \text{PRODUCT-DESC}(\times 2) \wedge \text{have}(\times 12,\ \times 2) \wedge \\
&\qquad \exists a1\ \#\text{product}(a1) \wedge \text{have}(\times 12,\ a1) \wedge \\
&\qquad \exists r1\ \text{STOCK}(r1) \wedge \exists b1\ \#\text{product}(b1) \wedge \text{have}(r1,\ b1) \wedge (a1 = b1) \wedge \\
&\qquad \exists c1\ \#\text{warehouse}(c1) \wedge \text{have}(r1,\ c1) \wedge \\
&\exists \times 13\ \text{WAREHOUSE}(\times 13) \wedge \exists d1\ \#\text{warehouse}(d1) \wedge \text{have}(\times 13,\ d1) \wedge (c1 = d1) \wedge \\
&\qquad \exists \times 3\ \text{CODE}(\times 3) \wedge \text{have}(\times 13,\ \times 3) \wedge (\times 3 = \text{BRA01}) \\
&\rightarrow \text{list}(\times 2)
\end{aligned}$$

Predicate names starting with # are key attributes in the SCOPE database. Scope of quantifiers is given by linear order and insignificant bracketing over "$\wedge$" has been left out to improve readability.

From the above translation it becomes clear that the TIME sense of 'at' (beat in the FOPC expression) was rejected in the above context and only the LOCATION sense succeeded. Similarly the only sense of 'name' that met the constraints of the system was PRODUCT-DESC (product description), i.e. only PRODUCTs can be located in WAREHOUSEs, not CUSTOMERs or SUPPLIERs. The path through the database linking the PRODUCT and WAREHOUSE relations via the STOCK relation is described by the equality of key attributes, #product, #stock and #warehouse.

Mapping PC-TYPED into the relational calculus Translation of the above PC-TYPED expression into Tuple Relational Calculus is now straightforward. The translation of 'show($\times 1$, $\times 2$)' in FOPC (where $\times 1$ is predicated

over as 'you' and $\times 2$ as 'name') into 'list($\times 2$)' in PC-TYPED indicates that $\times 2$ must be rendered as a free variable in the relational calculus (RC) expression. 'PRODUCT-DESC' will thus be placed in the target list of the RC query as follows:

$$\{\text{PRODUCT.PRODUCT-DESC} : (\exists r1 \in \text{STOCK})(\exists \times 13 \in \text{WAREHOUSE})$$
$$((\text{PRODUCT.\#product} = r1.\#\text{product}) \wedge$$
$$((r1.\#\text{warehouse} = \times 13.\#\text{warehouse}) \wedge$$
$$(\times 13.\text{CODE} = \text{BRA01}))))\}$$

The clear advantage of the above approach to supplying domain dependent interpretations is that, for any given expression, all possible meanings in terms of a given defined domain are produced. In many cases only one interpretation may emerge from the mapping process. Where an expression is ambiguous, with respect to a database, the user himself can decide which meaning he intended since a paraphrase of each alternative can now be presented. This is obviously an improvement on traditionally conceived systems which remove from the user all control over the disambiguation process, and which may, therefore, ultimately lead to the retrieval of misleading information.

3. Other system components

3.1 The paraphraser

The QUERYMASTER command, that is delivered by the last stage of the mappings executed by the natural language component, is passed on to a paraphraser. Its task is to give the user an English description of what the system has taken a question to mean. In cases where the front end has produced multiple interpretations, each of the QUERYMASTER commands is paraphrased in turn and the user is given a choice as to which particular (if any) reading is the one he intended.

The paraphraser embodied in this system is the same one as described in [Lowden and De Roeck 1986a, 1986b]. It maps the QUERYMASTER command first into the relational calculus, then into a predicate argument representation of the final text with the help of the data model, and then into a paragraph of English. We refer the interested reader to the above reference for more information.

The fact that the paraphraser works from relational calculus expressions, and the fact that a trivial mapping exists between each relational query language and the calculus means that neither the front end nor the paraphraser is dependent on the use of QUERYMASTER as the query language. In the present system, it would be possible for the paraphraser to work from the relational calculus expression delivered by the front end rather than the generated QUERYMASTER command. There are two reasons why that approach was not adopted.

Firstly, the system as a whole caters for users who wish to use QUERYMASTER directly, without using the NLFE. A casual user may feel confident enough to use the query language, but may still want the option of verifying whether his question has been adequately formulated and corresponds to what he intended. Secondly, although the relational calculus can accommodate the whole of QUERYMASTER, it is not the case that QUERYMASTER equates with the whole of the calculus. Users may ask queries that can be represented in the calculus but not in the query language. In that case, the user will be given an appropriate error message. Because we believe it is important that the system can report only on those questions which it can retrieve an answer for, it is imperative that the final translation into QUERYMASTER be effected before paraphrasing is initiated.

3.2 The meta-query handler

Little has been said so far regarding the handling of requests for meta information about the database. The reason is simply that the front end does not distinguish this type of question from other input it may receive. The approach is a consequence of our aim to build a general system which does not assume database interrogation as its sole purpose.

Meta queries typically are cast as modal questions. The present implementation only deals with the "can" modal but, with the necessary modification of the grammar dictionary and datamodel, other modals could also be included. A typical question for the present implementation might be:

"Can a credit limit exceed one million pounds?"

The question is passed through the various mappings by the natural language component and is ultimately rendered as a relational calculus expression with an empty left hand side (a closed predicate). That expression is then passed on to the meta query handler, a component which clearly is domain and task specific. An adequate reply to the question should not just be cast in terms of "yes" or "no", but must include an exhaustive list of conditions under which that answer holds, as well as exceptions when recorded. Providing an adequate reply to questions of this kind requires access to information on constraints regarding data integrity.

However, SCOPE, in common with many other databases, has constraints on data integrity built into the system in low level code. There are a number of practical disadvantages to this approach. Adding, changing and deleting of constraints typically involves manual (therefore costly and error prone) checking to see whether a modification will clash with other constraints already in place. No user access to rules is permissible and, more importantly, low level built in constraints cannot be used for any other purpose than to guarantee data consistency.

Recently, attempts have been made to use techniques developed in Artificial

Intelligence in order to build more flexible integrity checking mechanisms. The basic principle behind these new approaches is to state constraints as rules (i.e. declaratively) against which modifications to the database must be evaluated. The consequence is not only a lesser workload for systems professionals, but an increased potential in user facilities. Because the rules are separately stated they can be put to uses of their own. They can become part of a rule based inference mechanism, which can derive "new" constraints that are a consequence of existing ones. In effect, this means that a question concerned with a state of affairs for which no direct, immediately relevant rule is present can still be answered by evaluating the consequences of those rules which are in place. This task can be envisaged as part of the wider perspective of creating an intelligent knowledge base to act as a mediator between users and machines.

The meta query handler thus consists of a series of declaratively stated rules defining the constraints that the information stored in the database must conform with, and a mechanism for comparing information represented in the question with those rules. The kinds of questions which can be handled by the present implementation include:

- general questions of possibility
  e.g. Can a manager earn more than 20 000 pounds?
- existential modal questions of possibility
  e.g. Can Jones earn more than 20 000 pounds?
- modal commands
  e.g. Get all the customers whose credit limit can exceed 1 000 000 pounds.

Note that for the second query it is necessary to perform a database retrieval first in order to find out Jones's category.

The Modal Query interpreter operates on the assumption that to ask a query of a constraint base is logically equivalent to an attempt to assert an additional constraint into the base corresponding to the transformation of the query into an assertion. There are three possible outcomes that may arise from attempting to assert a new constraint w into an already consistent set of constraints W.

(a) $W \rightarrow w$. This would mean that $W \cup \{w\}$ characterises exactly the same valid database states as w and, thus, since nothing would be added or taken away from the constraint base by asserting w, the answer to the equivalent query to w is 'Yes'.
(b) $W \rightarrow \neg w$. This would mean that the assertion of w would lead to an inconsistent set of constraints and that the reply to the equivalent query to w should be 'No'.
(c) Neither (a) nor (b) is true, in which case the assertion of w would give rise to a new expanded consistent constraint base. The reply to the equivalent query to w would, in this case, be 'I don't know'.

These three outcomes follow the pattern established by Green (1969) for deductive first order question answering systems. There is a case, however, in a question answering system, for further dividing outcome (c) into two separate outcomes, (c1) and (c2):

(c1) There are rules in W which are logically related to w but neither (a) nor (b) follows from them. The reply to the equivalent query to w would then be 'Cannot be determined from existing rules'.
(c2) There are no rules in W which are logically related to w. In this case the reply to an equivalent question would be, 'There are no rules to this effect'.

This distinction is observed in the system described here as a means of providing more specific information to the user and as a means of identifying questions (possible constraint assertions, not identified by outcome (c)) which may not make sense in terms of the current definition of the database. The distinction arises naturally as a consequence of the implementation. Modal queries in extended relational calculus, output by the natural language component, are converted into hypothetical constraint assertions in clausal form and matched with existing constraints. If a match is obtained, a declarative rule packet is evoked which, using theorem proving techniques, establishes whether outcome (a), (b) or (c1) applies. If no match is obtained, then condition (c2) applies by default.

As an example consider a constraint base consisting solely of the rule:

'The maximum salary of managers of less than grade 15 is £40K'

which is represented in list form as follows:

[manager(x), grade(x, z), z < 15, salary(x, y), y < = 40] ... (1)

If the question,

'Can managers of grade 17 earn more than £45K?',

which can be mapped into a pattern like:

[manager(X), grade(X, Z), Z = 17, salary(X, Y), Y > 45] ... (2)

were asked of the system, the match between (1) and (2) would succeed, and thereby evoke a comparison rule which would assess whether adding (2) to a constraint base consisting of (1) would result in outcome (a), (b), or (c1). In this case the result would be (c1), since (2) is neither derivable nor falsifiable from (1).

If no applicable rules exist in the current constraint base for evaluating a question, the result would be outcome (c2).

The actual answers output by the system in response to outcomes (a) and (b) are more sophisticated than simple yes/no replies. In the case of outcome (a) extra information such as maxima and minima and other conditions are provided using canned English in conjunction with instantiated variables. In the case of a 'No' answer, exceptions to the rule are also given but, at present, this aspect of the system is trivial as exceptions are not built into the logic of the system but only into canned replies. Complex queries involving more than one constraint in the base are answered by itemising the replies generated by each rule comparison.

In the present system, existential questions of possibility are answered by retrieving the relevant details from a specially constructed prolog database in order to convert the question into a general one. For instance:

'Can Jones earn more than £30K?'

might be converted to

'Can a manager of Grade 14 earn more than £30K?'

by retrieving the fact that Jones is a manager of Grade 14. In a future integrated system, however, it should be possible to retrieve such information directly from the main database by generating a command in the appropriate query language.

In general, the Modal Query Interpreter described here is simply a prototype, implemented to explore the potential of natural languages for more sophisticated query processing. To this extent, the approach employed to date seems promising.

3.3 A personalised interface

The individual needs of every NLFE user are different. Each tends to express himself by selecting different words and can be expected to use different constructions. This applies to ordinary everyday conversation, and it is to be expected that the same will happen when communicating with machines.

It is practically impossible to foresee the particular needs of each and every front end user and to anticipate the words he will be likely to employ. Furthermore, if one tries to cater for the needs of all individuals, one may complicate the natural language component unnecessarily. For instance, if two people wish to use the same word consistently to refer to different things, then catering for both will result in a lexical ambiguity (although as far as the users are concerned there is no need to do so). This in turn will require the user to choose which interpretation he requires time and time again.

The most common solution to this problem involves restricting the vocabulary and syntax which all users are allowed to draw from. Although effective,

the result is definitely not user friendly. The system described here adopts a very simple but far more satisfactory solution. The main dictionary, grammar and datamodel are to be seen as the "core" system, the basic mechanism which is common to all users. Additionally, each user has a personal dictionary and grammar and also a personal datamodel/dictionary which allows him to choose which constructions and words (with associated word meanings) he prefers to use without unnecessarily complicating the overall system. Since these personal components are kept apart from those of other individuals, a change or addition to one of them cannot affect the environment defined by other users.

Personalised components are read in together with the core system whenever a user is recognised as a particular individual (upon logging into the front end).

3.4 Communication with knowledge engineers

In order to develop personal system components as described in the previous section, it is necessary that users have ways of communicating with the knowledge engineers responsible for introducing the changes the user requires. Furthermore, in order to improve overall system performance and record errors, there must be a way to trace anomalies when they occur.

In order to cater for the above tasks, a number of knowledge engineer communication files have been built into the present program. When a word is used that is not recognised by the front end, the user is asked whether he wants it included in his personalised dictionary. If so, he is asked to repeat the word and a context of use making clear which interpretation he wishes it to receive. The information is written to a file which the knowledge engineer can then consult.

A similar approach is adopted with input which the parser fails to recognise. If all the words in the input are known, but no satisfactory analysis can be found, the user is asked to record the sentence that caused the problem.

Various other sorts of errors can be recorded in the same way and the approach has the advantage that a user can report on an anomaly the very moment it occurs. The overall result is a more user-friendly system.

4. Conclusions

The system described is a first implementation of a prototype design. It was intended to verify the practical feasibility of constructing a generalised NLFE; however, the requirement to produce a working system in a relatively short timescale has necessarily meant that certain components are not fully developed.

For example the module catering for communication between user and

knowledge engineer could be made far more sophisticated. To take full advantage of such a facility, the core NL program (described in section 2) should be implemented in a more flexible way allowing for various checkpoints which could be reported on to the user.

The paraphraser incorporated in the system has been developed separately to report on QUERYMASTER and relational calculus commands. Its design reflects this aim to some extent and a more general text generator could be produced allowing modal queries to be reported on in a more flexible way. Furthermore, since QUERYMASTER cannot cope with universal quantification in the queries it allows, the paraphraser was not designed to cater for such expressions. This restricts the portability of the present system to query languages of the same type.

The implementation has also shown that the relational calculus is not necessarily the most appropriate means of expressing modal queries which could be better represented in the typed predicate calculus.

Despite these shortcomings no serious design problems have emerged and an efficient implementation of the current prototype would result in a flexible front end that compares favourably with more traditional systems in terms of portability and extensibility and in terms of the assistance it offers to casual users.

References

GREEN, C.: "Theorem Proving by Resolution as a Basis for Question Answering Systems", *Machine Intelligence*, 1969.

INTERNATIONAL COMPUTERS LTD.: "Using Querymaster (Q.M.250)", *Publication R00433/01*, 1985.

LOWDEN, B.G.T. and DE ROECK, A.N.: "REMIT: a natural language paraphraser for relational query expressions", *ICL Technical Journal*, Vol. 5, Issue 1, 1986a.

LOWDEN, B.G.T. and DE ROECK, A.N.: "The REMIT Systems for Paraphrasing Relational Query Expressions into Natural Language", in *Proceedings of the 12th International Conference on Very Large Data Bases*, Kyoto, Japan, 1986b.

STEEL, S. and DE ROECK, A.N.: "An Efficient Bidirectional Chart Parser with Heuristic Rule Application", *Proceedings of AISB*, 1987.

TURNER, R.: "Towards a New Foundation for Semantic Theory" in Turner, Chierchia and Partee (eds.), *Property Theory and Semantics*, MIT Press (forthcoming).

WEST, V.: "Natural Language Database Enquiry" in *ICL Technical Journal*, Vol. 5, Issue 1, 1986.

DAP-Ada: Ada Facilities for SIMD Architectures

L.M. Delves and M. McCrann
Centre for Mathematical Software Research, University of Liverpool, Liverpool, England

Abstract

We describe a package of facilities coded wholly within standard Ada, which provide language extensions to Ada aimed at expressing SIMD algorithms. The package is intended to complement the MIMD (tasking) facilities of Ada; the extensions are modelled after those provided by DAP-FORTRAN, an extended Fortran dialect developed for the ICL Distributed Array Processor. Examples of the use of the extensions are given.

1 Introduction

Parallel architectures are becoming increasingly common. The architectures being proposed, built, and in some cases even sold, are very diverse; but they come in three recognisable flavours:

Vector, or pipelined (Cray-1 etc, Cyber 205,)
SIMD (ICL DAP, Goodyear MPP)
MIMD (Intel Hypercube, FPS T-Series; other transputer-based machines

Expressing algorithms in vector or parallel form for these machines is not possible without suitable language facilities; to date, manufacturers have usually provided ad hoc extensions to Fortran, with the extensions being naturally both tailored to their own machines, and incompatible with others. The result has been a growing collection of Fortran dialects, and a growing portability problem. As parallel machines become the rule rather than the exception, it will be imperative to provide standard language facilities capable of expressing naturally the contructs handled efficiently by all three classes of machine. These constructs fall rather naturally into two classes:

a) MIMD machines: can run multiple, unrelated, concurrent processes.
b) Vector and SIMD machines: can perform parallel (or at least especially efficient) operations, of various types, on vectors and matrices.

ICL has spun out an independent company, Active Memory Technology Limited, to develop, manufacture and market DAP products. All enquiries should be directed to AMT Ltd. 65 Suttons Park Avenue, Reading RG6 1AZ.

A general purpose language should provide both types of construct. This is so only partly because of the need to cater for the newer architectures; just as importantly, there are many algorithms which are most easily expressed in matrix and vector form, independently of the hardware on which the algorithm is to run; while others are better expressed in the language of separate and concurrent tasks, even if they are to run on a serial machine. The language should also be widely available in a standard form. Currently, no language satisfies both of these criteria. The most widely used (for scientific purposes), Fortran, has a revised standard (Fortran8X) in preparation; the revision contains quite extensive SIMD (array and vector processing) facilities, based quite closely on those in DAP FORTRAN (which in turn were in part based on those in APL). But FORTRAN8X (at least as at January 1987) has no multi-tasking syntax – an omission which will be felt as soon as, or before, the standard is issued. The most likely alternative for a standard, widely available, scientific language, is Ada. Ada has well developed MIMD facilities (Ada tasking). It contains only rudimentary array-handling facilities.

Embedding SIMD facilities in an existing language can always be done by providing a long enough list of "system functions". Such extension are not very friendly in use. However, we have argued elsewhere [Delves and Mawdsley (1985)] that quite reasonable array handling facilities can be defined within an existing high level language, provided that the language is *extensible* in providing facilities for introducing new types, and for defining operations on those types. In [Delves and Mawdsley (1985)], a set of SIMD extensions were given for Algol68, the results exemplifying what can be achieved in providing a reasonably natural user image for the new facilities. Ada is in some respects more suitable than Algol68, and in others more limiting, for this kind of extension. Certainly, it is likely to be more widely used. In this paper, we look at the problem of defining DAP-Fortran like vector and array handling facilities wholly within standard Ada, and describe briefly an Ada package ("DAP-Ada") which implements these facilities. We give only a summary of the facilities here; for full details, see [Delves and McCrann (1985)]. A fuller motivation for the work is given in [Delves and Mawdsley (1985)], and the current paper follows the format of that reference closely.

Because Ada was designed to accept user-defined extensions, the extensions can be (and are) written wholly in Ada. Thus, SIMD programs can be developed using DAP-Ada even on serial machines. On parallel machines, the programs will run efficiently provided that the DAP-Ada package is implemented efficiently for the architecture involved; the compiler writer for the parallel machine may recognise the extensions explicitly and generate parallel inline code for them, or the procedures in the package may be handcoded. It is our belief that a set of "standard" SIMD extensions for Ada, should be developed; they could then reasonably be expected to be available on all machines supporting Ada. The facilities we describe show what can be achieved; but the package presented is *not* intended to be viewed as a draft

for a standard, since we have consciously omitted many details, and skimped on the facilities in a number of places, in a package developed originally for teaching purposes. We comment later on what we believe to be the most important omissions.

2 DAP-ADA

The extensibility of Ada lies in the ability of the user to define new data types; to extend the meaning of the built-in operators acting between variables of either new or existing types; and to overload user-defined function and procedure names. It is not possible to extend the syntax of Ada, and this has an obvious effect on the way in which user facilities can be provided.

In addition, Ada provides a "Generics" capability: an ability to write code in terms of abstract data types, with specific types provided as parameters when the code is "instantiated". This capability has been used in implementing DAP-Ada; however, it does not influence the design of the facilities in DAP-Ada, and we avoid explicit comment on its use.

In this section, then, we recall briefly the additional facilities introduced into FORTRAN by DAP-FORTRAN, and describe their equivalents in DAP-Ada.

2.1 Array operations

The most basic feature of the DAP is its ability to process whole arrays (of size up to 64×64) in parallel. DAP-FORTRAN reflects this ability by allowing the user to write whole-array operations, and operations on subsections of an array ("slices"). Table 1 gives examples, together with the equivalent code in DAP-Ada. We make the following comments on these examples:

1) apart from trivial representational differences (:= for assignment) the facilities for whole array operations look the same in the two languages.
2) Whole array assignment is already part of the Ada language. Arithmetic operations between arrays are not predefined, but the ability to define them is there. Thus, in the first five lines of Table 1, the operators $\times$, +, and * have been defined as extensions to the language. We note that the operation * is defined as pair-wise multiplication between matrix elements, rather than as an algebraic multiplication of the two matrices, because that is how it is defined in DAP-FORTRAN. Note also that the result of the matrix arithmetic operations is another matrix, space for which is generated automatically; this ability to generate storage as required, and to define functions returning matrix-valued results, is crucial to the DAP and to DAP-FORTRAN; it is already present in Ada. Similar facilities are available for Integer and Logical matrices, and for one-dimensional vectors.
3) Unlike DAP-FORTRAN (and Algol68), Ada does *not* provide syntax

for referring to a row or a column of a matrix (multi-dimensional slicing). All (Ada) scientific programmers regret this oversight even on serial machines; we have had to introduce the functions ROW, COL to provide a suitable facility.

4) It is evident from Table 1 that the equality comparison operator is treated non-uniformly: this non-uniformity stems from a recognised design defect in Ada, which provides default (and not always useful) definitions of equality between data types, and forbids re-definition (save in circumstances which are not useful here).

Table 1 Array arithmetic and comparison facilities in DAP-FORTRAN and DAP-Ada

Operation	DAP-FORTRAN	DAP-Ada
Array assignment	A = B	A: = B
Array addition	A = B + C	A: = B + C
Array multiplication	A = B*C	
Slicing a row	U = V + A(I,)	U: = V + ROW(A,I)
Slicing a column	U = V + A(,I)	U: = V + COL(A,I)
Array or Vector Comparisons	A.GT.B	A > B
	U.LE.V	U < = V
Array Equality	A .EQ. B	EQUAL(A,B)
Vector Equality	U .EO. V	EQUAL(U,V)
Logical operations	A.AND.B	A AND B
	U.AND.V	U AND V
	.NOT.A	NOT A
	.NOT.V	NOT V

A, B, C are assumed to be real or integer matrices. U, V are one-dimensional vectors.

2.2 System functions

Although DAP-FORTRAN significantly extends the standard FORTRAN syntax, it still provides a large number of operations via "system functions": pre-defined functions which can be called by the user. These are all very useful, but from the point of view of the present paper not very interesting: we merely have to provide routines which carry out the same operations, in Ada. The only feature of note is that DAP-FORTRAN allows the "overloading" of procedure names; that is, a given procedure name can refer to two or more procedures which expect different types of arguments and yield possibly different types of results. This facility allows, for example, the same name to be used for the procedure to sum the elements of a real matrix (SUM) as for the versions to sum the elements of an integer or logical matrix, or of a real, integer or logical vector. In Algol68, procedure names cannot be overloaded, but operators may; most of the system functions were therefore implemented as operators in DAP-Algol, leading to a notational difference (infix notation) for two-parameter generic functions, and to non-uniformity for functions with more than two parameters, for which alternative versions had to be given distinct names. Ada allows the overloading of function and procedure names; it also provides a "default value" capability for input arguments which is helpful. Hence, the system functions look neater in Ada than in Algol68.

A partial list of some of the more commonly used system functions, is given in Table 2; a full list is given in [Delves and McCrann (1985)].

Table 2 A partial list of system functions

DAP-FORTRAN	DAP-Ada	Type of result	Comments
ABS (ne)	ABS (ne)	same as argument	
EXP ATAN SIN COS SQRT LOG			also provided
FIX (re)	FIX (re)	ie	
FLOA (ie)	FLOAT (ie)	re	
ALL (le)	EVERY (le)	ls	logical AND of components (ALL is an Ada reserved word)
MERGE (ae, ae, le)	merge (ae, ae, le)	ae	merges first two depending on third argument

The allowable types of arguments are indicated as follows: s = scalar, v = vector, m = matrix, e = any of these, r = real, i = integer, l = logical, a = any of these, n = real or integer

2.3 Subscripting facilities

In DAP-FORTRAN, the FORTRAN concept of a subscript is generalised. In addition to the traditional use to specify a particular element of a vector or a matrix

A(i,j) ; V(i)

it is possible to specify a row or a column, and to provide integer and logical vectors and matrices as suffixes. The resulting facilities are extremely useful for specifying quite general DO loops without having to introduce an explicit loop. It is not possible within Ada to use the DAP-FORTRAN syntax as it stands; however it is straightforward to define new functions SUB, SUBR, SUBC which accept logical and integer vector and matrix arguments, and perform the same selecting actions as the "extended suffixing" provisions of DAP-FORTRAN. A list of the facilities is given in Table 3, together with their DAP-Ada equivalent.

Table 3 Subscript facilities available in DAP-FORTRAN and DAP-Ada

DAP FORTRAN	DAP Ada	MEANING
A(,i)	COL(A,i)	iTH column of A
A(i,)	ROW(A,i)	iTH row of A
A(LA,)	SUBR(A,LA)	{These forms each return a vector whose
A(,LA)	SUBC(A,LA)	components are a selection from the elements of
A(,IV)	SUBR(A,IV)	A or V. For details, see [ICL 1978]
A(IV,)	SUBC(A,IV)	
V(LV)	SUB(V,LV)	

(LV is a logical vector, LA a logical matrix, IV an integer vector).

2.4 Shift facilities

In DAP-FORTRAN there are facilities which enable users to perform datashifts on vector and matrix values. These shifts are performed by a set of pre-defined functions, each of which shifts either vector or array data between processors in the DAP processor array, in a horizontal ("EAST/WEST") or vertical ("NORTH/SOUTH") direction. The length of the shift is a parameter of the function; a simpler facility ("shift-indexing" – see below) is provided for shifts of length 1. Shifting a DAP-sized row of data one place to the right (say) introduces a blank in position 1, and shifts the right most data element out of the DAP processor array. How these edge effects are treated depends on the setting of what DAP-FORTRAN refers to as the GEOMETRY:

CYCLIC GEOMETRY: data shifted out are wrapped round and shifted back in at the other end.

PLANE GEOMETRY: data shifted out are lost; zeros or FALSEs are shifted in at the other end.

At any time, there is a standard geometry, set separately (and resettable) for the N-S and E-W direction. The shift operations themselves either impose an explicit temporary geometry, or use the default geometry. The facilities provided are listed in Table 4, with DAP-Ada equivalents; DAP-FORTRAN provides a separate function for each shift direction and shift geometry, while DAP-Ada provides a single SHIFT function, with the direction and geome-

Table 4 Shift operators

DAP-FORTRAN	DAP-Ada	Operation
SHNC(me,is)	SHIFT(mc,is,N,C)	Shift North Cyclic
SHNP(me,is)	SHIFT(mc,is)	Shift North Planar
SHSC(me,is)	SHIFT(mc,is,S,C)	Shift South Cyclic
SHSP(me,is)	SHIFT(mc,is,S)	Shift South Planar
SHWC(me,is)	SHIFT(mc,is,W,C)	Shift West Cyclic
SHWP(me,is)	SHIFT(mc,is,W)	Shift West Planar
SHEC(me,is)	SHIFT(mc,is,E,C)	Shift East Cyclic
SHEP(me,is)	SHIFT(mc,is,E)	Shift East Planar
SHLC(ve or me,is)	SHIFT(vc,is,N,C)	Shift Left Cyclic
SHLP(ve or me,is)	SHIFT(vc,is)	Shift Left Planar
SHRC(ve or me,is)	SHIFT(vc,is,S,C)	Shift Right Cyclic
SHRP(ve or me,is)	SHIFT(vc,is,S)	Shift Right Planar
A(+,)	NORTH(A) or NORTH(A,1)	
A((−,)	SOUTH(A)	Suffixed elements move in the stated
A(,+)	WEST(A)	direction with the default geometry.
A(,−)	EAST(A)	All versions may have 1 or 2 args
V(+)	WEST(V)	etc.
V(−)	EAST(V)	
A(−,+)	EAST(SOUTH(A))	

The allowable types of argument are as follows: s = scalar, v = vector, m = matrix, a = any of these. r = real, i = integer, b = boolean, e = any of these, c = r or b. An entry c for DAP-Ada implies we have not bothered to implement the integer equivalent.

try specified by additional arguments which default to NORTH, PLANE. The Ada facilities seem more uniform at least to us.

DAP-FORTRAN also provides Left and Right shifts for matrices; these treat the matrix as a long vector. This concept has not been mimicked in DAP-Ada, nor is it needed since DAP-Ada shift operators accept any length vector.

DAP-FORTRAN also permits the use of + and − as array subscripts to indicate a shift. This facility is provided in DAP-Ada by defining functions NORTH, SOUTH, EAST and WEST. The geometry of these shifts is given by the current geometry, as set in the global variables NSGEO and EWGEO. These can be altered at any time by either a straightforward assignment of the form:

NSGEO := PLANE (or CYCLIC);

or by calling a procedure GEOMETRY which takes a boolean vector argument of any size and sets either the NS, or both the NS and EW geometrys (see [ICL 1978]).

2.5 Masked assignments

The concept of a "Logical Mask" is a very important one in DAP-FORTRAN. A logical mask is a matrix of logical values which is used to determine which of the DAP processors shall be active during a given operation; examples of the use of such masks during subscripting operations, are given in the previous section. Equally important is their use during assignments: it is very common to find that time can be saved by computing a whole matrix of values, and then throwing away the unwanted ones during the assignment of the results to storage. In DAP-FORTRAN, the syntax for such a masked assignment is not distinguished from that for masked suffixing; the difference is detected by the compiler from the context.

In DAP-FORTRAN, a masked assignment takes the form:

A(MASK) = matrix-expression

which assigns the values of the components of the matrix expression to the elements of the matrix A, but only for those elements for which the corresponding element of the logical matrix MASK is TRUE.

The simplest way to provide masked assignments in Ada is via an ASSIGN procedure with argument list as for MERGE;

ASSIGN(A, matrix expression, MASK);

It is unfortunate that this lacks some of the mnemonic succinctness of the DAP-FORTRAN equivalent, and DAP-Algol was able to get closer to the original. However, Ada does allow us to provide a way of associating a matrix and a logical mask, and manipulating these together. We introduce the types.

MASKED_MATRIX, MASKED_VECTOR

(which are records containing the relevant two fields)

and can then provide (masked) arithmetic operations between variables of these types, and mixed operations between masked and unmasked vectors and matrices. For example, if MA, MB are masked matrices, the constructs

MA + B; MA*MB; A − MA; etc

are accepted, returning in each case a masked matrix. The assignment:

ASSIGN(A, masked_matrix_expression);
ASSIGN(V, masked_vector_expression);

are also accepted.

DAP-FORTRAN also accepts a number of other forms of masked assignments. Again, the syntax cannot be followed directly; but the constructs can all be expressed reasonably naturally within the DAP-Ada facilities already provided.

DAP-FORTRAN	DAP-Ada
A(LV,) : = matrix-expression	ASSIGN(A, mat_exp, COL(ELN(LV)))
A(,LV) : = matrix-expression	ASSIGN(A, mat_exp. ROW(ELN(LV)))
A(IV,) : = vector-expression	ASSIGN(A, mat_exp, COL(IV))
A(,IV) : = vector-expression	ASSIGN(A, mat_exp, ROW(IV))

and other variants.

2.6 Other features

DAP-FORTRAN contains other facilities (see ICL (1978)]). Most of these have close equivalents in DAP-Ada (see [Delves and McCrann (1985)], with the following exceptions:

1) Variable length reals and integers are not currently supported.
2) The debug facilities are not supported.
3) There is no equivalent of a FORTRAN COMMON block, and the distinction between DAP and HOST code is not maintained. Therefore, the conversion routines between DAP and HOST formats are not mimicked.

We hope to remedy 1) in due course; 2) and 3) are deliberate, since the intention is not primarily to mimick the DAP, but to provide general SIMD language facilities.

3 An Example

We illustrate the correspondence between DAP-Ada and DAP-FORTRAN, with an example: we compare two programs which perform a bubble sort on

N < = dapsize^2 elements, assumed provided in the first N locations of a DAPSIZE × DAPSIZE matrix and padded out (in the DAP-FORTRAN version) with dummy (large) values.

DAP-FORTRAN version, taken from [Gostick (1979)]

```
      REAL MATRIX FUNCTION BUBBLE (VALUE)
      REAL VALUE(,)
      LOGICAL MASK(,),CHANGE(,)
      MASK = ALTR(1)
    1 CHANGE = VALUE.LT.VALUE( + )
      IF(.NOT.ANY(CHANGE)) GOTO 10
      CHANGE = CHANGE.AND.MASK
      CHANGE = CHANGE.OR.CHANGE( - )
      VALUE(CHANGE) = MERGE(VALUE( + ),VALUE( - ),MASK)
      MASK = .NOT.MASK
      GOTO 1
   10 BUBBLE = VALUE
      RETURN
      END
```

DAP-Ada version

```
PROCEDURE Bubble_Sort (a: In OUT INT_VEC) IS

    a_prime: INT_VEC(a'RANGE);
    hiding_mask: BOOL_VEC(a'RANGE);
      change: BOOL_VEC(a'RANGE);

BEGIN

      hiding_mask : = NOT Alt(1);
      change : = a  <  West(a); change(change'LAST) : = FALSE;

      WHILE Any(change)
      LOOP
        a_prime : = a;
        change : = change AND hiding_mask;
        a : = Merge(West(a), a, change);
        a : = Merge(East(a_prime), a, East(change);
        hiding_mask : = NOT hiding_mask;
        change : = a  <  West(a); change(change'LAST) : = FALSE;
      END LOOP;

END BUBBLE_sort;
```

Although these two codes were written independently, rather than the Ada being a transliteration of the FORTRAN, the correspondence between them is very close, with the exception that we have used a WHILE loop in DAP-Ada to avoid the two GOTOs of DAP-FORTRAN. Note however that the DAP-FORTRAN code assumes that the data to be sorted is in a DAP matrix, which it then treats as a vector ("long-vector" in DAP notation). This

standard trick will be familiar to all FORTRAN users, but comes as a surprise to programmers from other languages; in DAP-Ada, arbitrary length vectors are accepted and the data required can therefore be provided in a vector.

4 Timings

DAP-Ada was originally developed as a teaching tool for an M.Sc class in parallel processing. The package therefore includes a pseudo timer; a global variable called TIME is updated whenever any of the DAP-Ada procedures or operators are entered, by the time taken for the equivalent facility on the DAP. Interrogating the timer then allows estimates of the speed of the corresponding DAP code. For the example given here, with DAPSIZE = 64, we obtain the timings given in Table 5. We see that the pseudo times are about 10% too slow. This is quite good enough accuracy to compare algorithms, since a 10% change in the speed of an algorithm is rarely significant in practice.

Table 5 Timing Results

Code		DAP-Fortran time	DAP-Ada pseudo timer
Bubble sort,	N = 256	79.2	86.0
	512	158.3	172.0
	1024	316.6	344.1
	2048	633.2	688.1
	4096	1266.4	1376.3

DAP-FORTRAN times obtained on the ICL DAP at QMC London. DAP-ADA run on a DEC microVAX II with DEC Ada. Times are in msec.

5 Comments

The facilities provided by DAP-Ada are, as the example shows, sufficiently close to those in DAP-FORTRAN that it is possible to develop DAP algorithms quite naturally in DAP-Ada and then translate line-by-line. The development is in practice aided considerably by the relatively good accuracy of the pseudo timer.

We achieve this timing accuracy because most DAP programs consist mainly of calls to array features, which are trapped by the pseudo-timer, and have relatively few sections of "serial" code in them which are replaced in DAP-Ada by standard Ada and hence are not timed. This in turn reflects the success of DAP-FORTRAN in expressing the operations which are needed for parallel processes on this type of machine; and suggests that facilities such as those in DAP-Ada would, if implemented efficiently, provide an appropriate Ada-based language for the DAP and other SIMD machines. However, we should point to two related limitations of the facilities described:

1) No allowance has been made for handling parallel three-dimensional

matrices. This limitation is perhaps natural on the DAP, which has only two-dimensional parallelism; but it is often necessary to declare multi-dimensional arrays, even if only a two-dimensional slice is handled at one time. The lack of a built-in slicing operation in Ada makes this a far from trivial type of extension to provide, whereas DAP-Algol was able to point to the Algol68 slicing mechanism as providing such an extension automatically.

2) The "pseudo-slicing" operations ROW, COL, which were introduced specifically to extract a row or column from a matrix, do so by copying. This provides a reasonably satisfactory facility for use on the right hand side of an assignment:

   V := ROW(A,i)

   but not on the left hand side:

   ROW(A,i) := v

   does not have the "obvious" effect.

3) As with DAP-Fortran, all of the facilities are fairly profligate with the use they make of memory: space for vectors and matrices is generated whenever it is convenient to do so. This is probably acceptable in a strictly two-dimensional set of extensions: copying or generating three-dimensional matrices would need rather more careful justification.

References

DELVES, L.M. and MAWDSLEY, S. (1985). DAP-Algol: a development system for parallel algorithms. Computer Journal 28, 148–154.

DELVES, L.M. and McCRANN, M. (1985). DAP-Ada Users Manual. Technical Report, Department of SCM, University of Liverpool.

ICL (1978) DAP-FORTRAN Language Manual, ICL Technical Publication 6918.

MAWDSLEY, S. (1983). DAP-Algol Users Manual, Technical Report, Department of SCM, University of Liverpool.

GOSTICK, R.W. (1979). Software and Algorithms for Distributed Array Processor, ICL Technical Journal I(2) . 116–135.

Quick Language Implementation

H. Gardiner, R.W. Lyttle, P. Milligan and R.H. Perrott
Department of Computer Science, The Queen's University of Belfast, Belfast BT7 1NN, N. Ireland

Abstract

This paper concerns an implementation of a parallel Pascal based language for the ICL DAP. This has been achieved by using the existing high-level language of the DAP, DAP FORTRAN, as the target language in a translator. The benefits of this approach have been to give early experience of the use of the new language before proceeding with a full implementation. The details of the translation are given.

1 Introduction

Advances in the development of parallel architectures have not been matched by corresponding advances in the design of programming languages which would enable programmers to express problem solutions in a straightforward manner. Invariably the only language available to the programmer is a variant of FORTRAN, such as CFT[1] or DAP FORTRAN.[3] These languages contain constructs reflecting the underlying architecture so that portability of programs is not readily possible.

Whilst it is a relatively simple process to design a new computer programming language, problems arise when attempting to implement such a language on machines whose underlying architecture and machine code are not sympathetic. The well known alternative implementation technique of generating pseudo-code which is then interpreted is not applicable to supercomputer language implementations due to the relative slowness of execution and the fact that the hardware of the machine is not being fully exploited.

In this paper we describe a further approach, translation of a parallel Pascal based language, Actus II[4], into an existing dialect of FORTRAN, DAP FORTRAN. The translated source program is then compiled and executed on the ICL DAP. The translation is achieved by scanning the syntax graph, produced by the Actus II compiler, and inserting calls to routines which

---

ICL has spun out an independent company, Active Memory Technology Limited, to develop, manufacture and market DAP products. All enquiries should be directed to AMT Ltd. 65 Suttons Park Avenue, Reading RG6 1AZ.

generate the required DAP FORTRAN. This approach[2] obviates the need for the implementor to have an in-depth knowledge of the machine architecture (with all of the complexities that this entails), rather all that is necessary is an understanding of the target language. It will of course be argued that a translation such as this reduces the portability aspect outlined above. However for an experimental language, such as Actus II, the experience gained in being able to exploit fully the parallel nature of the machine is a major factor.

2 Actus II and DAP FORTRAN

Actus II is a Pascal based language designed for implementation on array processors. It is a refinement of Actus[5,6] which was designed to be portable over vector and array processors and has all of the main advantages of Pascal, such as meaningful data structures, error detection and diagnostic facilities. The language constructs of DAP FORTRAN constrain the maximum 'extent of parallelism' (e-o-p) to a maximum of either 64 elements in the case of vector processing, or 4096 (64×64) elements in the case of array processing. Thus the e-o-p is the maximum number of elements which can be processed in parallel at any one time. An ideal implementation of Actus II will translate the user-defined parallelism of the problem solution into the physical parallelism provided by the number of processing elements in the hardware. As the strategy for mapping data structures whose size is greater than that of the physical hardware is both complex and time consuming, we will, for the purposes of this quick implementation, restrict the maximum size of an Actus II parallel data structure to conform to that of the target language, DAP FORTRAN.

2.1 Arrays

The array data declaration in Actus II is used to define those data items which may be manipulated in parallel. The sequential dots '..' in an array declaration indicate that the array is to be processed one element at a time. If, during the array definition, these sequential dots are replaced by parallel dots ':' then this indicates that the array index may be manipulated in parallel. For example

```
SEQ: array[1..64] of INTEGER;
```

defines a non-parallel array of 64 elements

```
PARA: array[1:64] of INTEGER;
```

defines a one-dimensional parallel array of 64 elements, all of which may be accessed simultaneously. The declaration

```
PARB: array[1:64, 1:64] of INTEGER;
```

defines a two-dimensional parallel array with a total of 4096 elements, all of which may be accessed simultaneously.

Sequential vectors and arrays are translated into similar DAP FORTRAN constructs, thus the DAP FORTRAN equivalent of the vector SEQ would be

```
INTEGER SEQ(64)
```

In keeping with the two-dimensional nature of DAP FORTRAN, and to facilitate a uniform translation scheme, all Actus II parallel vectors are translated into two-dimensional DAP FORTRAN arrays, the vector being stored in the first column of the array, e.g.

```
INTEGER PARA(,)
```

Two-dimensional parallel Actus II arrays are translated into their equivalent DAP FORTRAN constructs, thus PARB becomes

```
INTEGER PARB(,)
```

2.2 Index Sets

To allow simultaneous access to all or selective elements of a parallel variable, Actus II retains the index set concept of Actus but introduces a greater degree of flexibility. In Actus II there are two types of index set, viz.,

(a) explicit index sets which remain constant throughout their defining block, e.g.,

```
index
  ONE_TO_64: 1:64 ;
```

Note that it is also possible to define index sets with a regular increment, such as

```
EVENS: 2:[2]64 ;
```

which represents the indices 2, 4, 6 ... 64
'Broken' and 'random' ranges can also be formed using the set operators '+' and '−' as in

```
RANGE_A: 2:2 + 4:4 + 6:6 ;
RANGE_B: 1:10 − 5:5 ;
```

The intersection of two index sets can be achieved by use of the intersection operator '*'.

(b) redefinable index sets which enable the selection of different portions of a parallel variable, thus an index set defined as follows

```
index
  IS1: INTEGER ;
```

must have values assigned to it in a using statement (see below).

In keeping with the philosophy whereby Actus II parallel vectors are represented by DAP FORTRAN arrays, index sets are translated into DAP FORTRAN parallel arrays of type LOGICAL. The presence of an index set

value is denoted by the value TRUE in the DAP FORTRAN logical array. These values are determined by a run-time support routine COL_MATRIX (written in DAP FORTRAN). This routine creates the required index set by constructing a logical matrix with the values TRUE and FALSE in the appropriate places in the columns of the logical matrix. Thus for the above Actus II index sets the translator generates

```
ONE_TO_64   =  COL_MATRIX(1,64,1)
EVENS       =  COL_MATRIX(2,64,2)
RANGE_A
            =
               COL_MATRIX(2,2,1).AND.
               COL_MATRIX(4,4,1).AND.
               COL_MATRIX(6,6,1)
RANGE_B
            =
               COL_MATRIX(1,10,1).AND..NOT.
               COL_MATRIX(5,5,1)
```

2.3 Parallel Constants

In addition to scalar constants, Actus II allows the definition of parallel constants which define a sequence of values and which may be used to assign initial values to a parallel index of an array, e.g.,

```
parconst
    EVENS   =  2:[2]20 ;
    ODDS    =  1:[2]9 ;
    BOTH    =  1:[2]9 , 2:2[2]10 ; {where ',' acts as a
               catenation operator}
```

Parallel constants are stored as two-dimensional parallel arrays, but in this case the actual values of the parallel constant are stored in the appropriate positions in the columns of the array.

2.4 Parallel Statements

The assignment statement and the if, case, and while constructs of Actus are expanded in Actus II to cater for the expression of two-dimensional parallelism. In practice the e-o-p for these statements is first established by what is called a using statement and is subsequently manipulated by the particular statement type.

A using statement defines the e-o-p for the statements which it encloses. It has the following form:

```
using index-specification do statement;
```

The index-specification contains either explicit index identifiers which have already been given values at their point of declaration, or redefinable index

identifiers with an associated set of values, or a mixture of both. A maximum of two index identifiers may be present in the index-specification, in keeping with the array processing nature of Actus II. Thus a using statement constructs either a one-dimensional or two-dimensional mask for its following statements.

The e-o-p's formed whenever extent setting statements are encountered are represented as logical two dimensional parallel arrays whose elements are set to either TRUE or FALSE depending upon the presence or absence of an element in the e-o-p. For example an Actus II using statement of the form

using IS1 := 1:4, IS2 := 2:5 – 3:3 do

would be translated into the following DAP FORTRAN

```
RMATRIX    =  .FALSE.
CMATRIX    =  .FALSE.
CMATRIX    =  COL_MATRIX(1,4,1)
RMATRIX    =  ROW_MATRIX(2,5,1).AND.NOT.
              ROW_MATRIX(3,3,1)
EOP        =  CMATRIX.AND.RMATRIX
IF (.NOT.ANY(EOP)) GOTO {end of using statement}
```

The logical matrix function ROW_MATRIX (which is a complementary function to the earlier COL_MATRIX function) constructs a matrix with the value TRUE in the appropriate rows. These functions are used to form two grid masks, one for each index set in the using statement. These masks are superimposed to determine the correct e-o-p. Only if at least one element of the e-o-p is TRUE will the statement part of the using statement be executed. Thus the above example, assuming an 8 × 8 DAP configuration, can be represented as

CMATRIX								RMATRIX							
T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	T	T	T	T	T	T	T	T
T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	T	T	T	T	T	T	T	T
T	T	T	T	F	F	F	F	T	T	T	T	T	T	T	T
T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F
T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F

E-O-P

F F F F F F F F
T T T T F F F F
F F F F F F F F
T T T T F F F F
T T T T F F F F
F F F F F F F F
F F F F F F F F
F F F F F F F F

Assignment to a parallel structure is achieved by subscripting the structure with the appropriate index set identifiers. For example given the following declarations

```
var
  PARB: array[1:5, 1:10] of INTEGER ;

index
  ROWS: 1:5 ;
  COLS: 1:10 ;
```

then the assignment

```
using ROWS, COLS do
  PARB[ROWS,COLS] : = 0 ;
```

will initialise all 50 elements of PARB to zero simultaneously, and will be translated into the following DAP FORTRAN

```
PARB(EOP) = 0
```

where EOP is a two dimensional logical matrix representing the e-o-p.

Assuming the Actus II declarations

```
var
  PARA: array[1:64] of INTEGER ;
  PARB: array[1:64,1:10] of INTEGER ;

index
  FIRST: 1:64 ;
  SECOND: INTEGER ;
```

the following are examples of Actus II using statements and their translated DAP FORTRAN CODE

```
using FIRST do
  PARA[FIRST] := 1 ; { set all 64 elements of PARA to 1 }
CMATRIX = COL_MATRIX(1, 64, 1)
RMATRIX = ROW_MATRIX(1, 64, 1)
EOP = CMATRIX.AND.RMATRIX
IF (.NOT.ANY(EOP)) GOTO 1
PARA(EOP,1) = 1
1
using FIRST, SECOND := 1:10 do
  PARB[FIRST,SECOND] := 5 ;
CMATRIX = COL_MATRIX(1,64,1)
RMATRIX = ROW_MATRIX(1,10,1)
EOP = CMATRIX.AND.RMATRIX
IF (.NOT.ANY(EOP)) GOTO 2
PARB(EOP) = 5
2
```

It is permissible to nest using statements to any depth, but only those index identifiers associated with the nearest enclosing using statement are available for use at any stage, for example

```
using IS1 do                        — stack IS1
begin
{ statements involving IS1 }
using IS2 do                        — stack IS2
  begin
  { statements involving IS2 }
  end                               — unstack and discard
                                    — IS2
{ current e-o-p is IS1 }
end                                 — unstack and discard
                                    — IS1
```

Access to the diagonal components of an array is achieved by using the same index set for both indices, e.g.,

```
var
  PARC: array[1:64,1:64] of INTEGER ;
index
  IS1: 1:64 ;
begin
using IS1, IS1 do
  PARC[IS1,IS1] := 0 ; { zeroise the array }
using IS1 do
  PARC[IS1,IS1] := 1 ; { set the leading diagonal of PARC to 1 thus
                         creating the identity matrix }
end ;
```

This is translated into the following DAP FORTRAN code

```
      CMATRIX = COL_MATRIX(1,64,1)
      RMATRIX = ROW_MATRIX(1,64,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 1
      PARC(EOP) = 0
    1 CMATRIX = COL_MATRIX(1,64,1)
      RMATRIX = ROW_MATRIX(1,64,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 2
      PARC (DIAGONAL(1,64,1,EOP)) = 1
    2
```

where the run-time routine DIAGONAL returns a logical matrix with the necessary elements of the leading diagonal set to true. Thus in the above example the call to DIAGONAL (1, 64, 1, EOP) results in the following logical matrix being formed

```
1 0 0 0 0 . . . .
0 1 0 0 0 . . . .
0 0 1 0 0 . . . .
0 0 0 1 0 . . . .
0 0 0 0 1 . . . .
. . . . . . . . .
```

2.5 If Statement

If the boolean-expression of an if-statement yields a parallel result (i.e. a set of boolean values) then this set is used as the e-o-p for the then clause and the else clause is executed with the complementary set of boolean values, for example

```
var
  PARD, PARE: array[1:30,1:30] of INTEGER ;

index
  IS1, IS2: 1:30 ;
begin
using IS1, IS2 do
  if PARD[IS1,IS2] > 0
  then
    PARE[IS1,IS2] := 1
  else
    PARE[IS1,IS2] := PARE[IS1,IS2] + 1
end ;
```

Here the components of PARE for which the corresponding components of PARD are positive (if any) are assigned the value 1 and the remaining components (if any) are incremented by 1. The DAP FORTRAN generated for this is as follows:

```
Code for using statement
  CMATRIX = COL_MATRIX(1,30,1)
  RMATRIX = ROW_MATRIX(1,30,1)
  EOP = CMATRIX.AND.RMATRIX
  IF (.NOT.ANY(EOP)) GOTO 1
  CALL STACK(EOP)

Code for if test
  EOP = EOP.AND.(PARD(,).GT.0)
  IF (.NOT.ANY(EOP)) GOTO 2
  Code for then part
  PARE(EOP) = 1

Construct e-o-p for else limb
2 EOP = (.NOT.EOP).AND.EOPSTACK(„STACKTOP)
  IF (.NOT.ANY(EOP)) GOTO 3

Code for else limb
  PARE(EOP) = PARE(,) + 1

C end of using statement, remove e-o-p
3 CALL UNSTACK(EOP)
1
```

The point to note about the if statement is that two different e-o-p's are used, the first in the then clause and the second, which is the complement of the first, in the else clause. In general, whenever a using statement is encountered the e-o-p associated with it is evaluated and placed onto a stack. If, during the execution of a using statement, another extent setting statement is encountered which results in the modification of the current e-o-p (such as the if statement in the above example), then the e-o-p on top of the stack is combined with this new e-o-p for the next group of statements.

2.6 Case Statement

A case statement has the general form:

```
case selector of
  case-label-list1 : S1 ;
  ...
  case-label-listn : Sn
end ;
```

If the selector expression yields a parallel result, then the e-o-p of the selector expression is distributed among the case limbs by comparing the value of the selector expression for each element with the appropriate case label. Thus each case limb will have a different e-o-p associated with it.

The case statement can be considered as an extended form of the if statement. Whereas in the if statement the requisite e-o-p's are evaluated before each limb is executed, in the case statement all possible e-o-p's are evaluated and stored (in a queue) before each limb of the statement is executed. It would be possible to use a stack as opposed to a queue to store the e-o-p's but for ease of implementation and to facilitate the production of the DAP FORTRAN code in the same order as the Actus II code, this alternative data structure has been chosen. Prior to the execution of each limb the e-o-p at the head of the queue is removed, checked to ascertain whether it contains any elements and if so used as the e-o-p for the current limb.

For example

```
var
  PARH: array[1:4,1:4] of 0..4 ;

index
  IS1, IS2: 1:3 ;

begin
using IS1, IS2 do
  case PARH[IS1,IS2] of
    0..3: PARH[IS1,IS2] := 0 ;
    4   : PARH[IS1,IS2] := 1
  end
end.
```

is translated to

```
      CMATRIX = COL_MATRIX(1,3,1)
      RMATRIX = ROW_MATRIX(1,3,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 1
      CALL STACK(EOP)
      CALL QUEUE(EOP.AND.(PARH(,).GE.0).AND.PARH(,).LE.3))
      CALL QUEUE(EOP.AND.(PARH(,).EQ.4))
      CALL UNQUEUE(EOP)
      IF (.NOT.ANY(EOP)) GOTO 2
      PARH(EOP) = 0
    2 CALL UNQUEUE(EOP)
      IF (.NOT.ANY(EOP)) GOTO 3
      PARH(EOP) = 1
    3 CALL UNSTACK(EOP)
    1
```

2.7 While Statement

As with if and case statements, the boolean-expression associated with a while statement may yield a parallel result which determines the e-o-p for its enclosing statements, for example

```
var
  PARF, PARG: array[1:7,1:7] of INTEGER ;

index
  IS1, IS2: 1:7 ;

begin
using IS1, IS2 do
  begin
  PARG[IS1,IS2] := 0 ;
  while PARF[IS1,IS2] <> 0 do
    begin
    PARG[IS1,IS2] := PARG[IS1,IS2] + 1 ;
    PARF[IS1,IS2] := PARF[IS1,IS2] - 2 ;
    end
  end
end ;
```

Here the body of the while statement is repeatedly executed while at least one component of the array PARF is non-zero. At each iteration the loop body, the components of PARF which do not satisfy the condition are removed from the e-o-p and execution of the loop body terminates whenever the e-o-p becomes empty. This is translated into the following DAP FORTRAN code.

```
      CMATRIX = COL_MATRIX(1,7,1)
      RMATRIX = ROW_MATRIX(1,7,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 1
      PARG(EOP) = 0
      CALL STACK(EOP)
      EOP = EOP.AND.(PARF(,).NE.0)
      IF (.NOT.ANY(EOP)) GOTO 2
    3 PARG(EOP) = PARG(,) + 1
      PARF(EOP) = PARF(,) - 2
      EOP = EOP.AND.(PARF(,).NE.0)
      IF (ANY(EOP)) GOTO 3
    2 CALL UNSTACK(EOP)
    1
```

2.8 Subprograms

All subprograms can take parallel arrays as parameters and, in addition, functions can return parallel arrays as results. These parallel result arrays must be subscripted, as in

```
using IS1, IS2 do
  A[IS1,IS2] := F (parameters)[IS1,IS2]
```

The translation from Actus II procedures and functions to DAP FORTRAN subroutines and functions is quite straightforward. The main point to note is

that because of the parameter passing mechanism of FORTRAN, it is necessary to take a copy of all Actus II value parameters on entry to a subprogram block and to replace all references to these value parameters by references to a copy of the parameter.

Functions which return parallel arrays as results cannot be directly translated into DAP FORTRAN, instead a temporary variable is created to hold the result of the function call and this is used in subsequent expressions. For example the following Actus II function returns an array each of whose elements is the square of the corresponding element of the input parameter.

```
type
  PARARRAY = array[1:32,1:32] of INTEGER ;

function SQUARE (A: PARARRAY): PARARRAY ;
  index
    IS1, IS2: 1:32 ;
  begin
  using IS1, IS2 do
SQUARE[IS1,IS2] := A[IS1,IS2] * A[IS1,IS2]
  end ;
```

The DAP FORTRAN code generated for this subprogram is as follows:

```
      INTEGER MATRIX FUNCTION SQUARE (A)
      LOGICAL RMATRIX(,), CMATRIX(,), EOP(,)
      INTEGER A(,)
      INTEGER PARAM_A(,)

    Copy the input parameter
      PARAM_A = A

      CMATRIX = .FALSE.
      RMATRIX = .FALSE.
      CMATRIX = COL_MATRIX(1,32,1)
      RMATRIX = ROW_MATRIX(1,32,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.(ANY(EOP)) GOTO 1
      SQUARE(EOP) = PARAM_A(,) * PARAM_A(,)
    1 RETURN
      END
```

A call of this function such as

```
using IS1 := 1:32, IS2 := 1:32 do
  B[IS1,IS2] := SQUARE(A)[IS1,IS2]
```

Results in the following DAP FORTRAN being generated

```
{ construct the EOP }
TEMP = SQUARE(A)
B(EOP) = TEMP
```

2.9 Data Alignment

Two data alignment operators (shift and rotate) are present in the language and operate upon index sets. The shift operator causes movement of data within the declaration range of the e-o-p whilst the rotate operator causes the data to be shifted circularly (wrap-around) with respect to the current e-o-p.

Shifting and rotating of index-sets are achieved by invoking run-time functions, written in DAP FORTRAN, which take, as parameters, the structure which is to be manipulated, the current e-o-p and the direction and amount of the shift or rotate.

2.9.1 Shifting

This is the most straightforward of the two data alignment operations. The run-time routine merely invokes an appropriate combination of the standard DAP FORTRAN functions SHNP, SHSP, SHEP or SHWP, which causes the 64×64 array of values to be shifted in a N, S, E or W direction respectively. For example an Actus II statement of the form

```
using IS1 := 2:4, IS2 := 2:4 do
  AA[IS1,IS2] := BB[IS1 shift 1, IS2 shift -1]
```

results in the following assignments being performed

```
AA[2,2] := BB[3,1]   AA[2,3] := BB[3,2]   AA[2,4] := BB[3,3]
AA[3,2] := BB[4,1]   AA[3,3] := BB[4,2]   AA[3,4] := BB[4,3]
AA[4,2] := BB[5,1]   AA[4,3] := BB[5,2]   AA[4,4] := BB[5,3]
```

A run-time support routine, written in DAP FORTRAN, is called on each occasion that a shift operation is to be performed. The sign of the amount of the shift signifies the direction of the shift.

2.9.2 Rotating

When applying a shift operation to an e-o-p non-active elements of the e-o-p may become active, for example in a using statement of the form

```
using IS1 := 1:[2]5 do
```

the active elements are 1, 3 and 5. Shifting this e-o-p two places to the right results in the elements 3, 5 and 7 becoming the active elements. This is possible when a shift operator with a distance 2 is applied to the e-o-p, but when the rotate operation is applied no non-active elements may become active, the current elements are merely rearranged. Thus a more complex algorithm has to be implemented to cater for rotation.

For example assume that we have the following 5×5 array, AA

1	2	3	4	5
3	5	7	9	11
5	8	11	14	17
7	11	15	19	23
9	14	19	24	29

then the using statement

```
using IS1 := 2:4, IS2 := 2:4 do
  AA[IS1,IS2] := AA[IS1 rotate 1,IS2 rotate -1]
```

will result in the following assignments being made

```
AA[2,2] := AA[3,4]   AA[2,3] := AA[3,2]   AA[2,4] := AA[3,3]
AA[3,2] := AA[4,4]   AA[3,3] := AA[4,2]   AA[3,4] := AA[4,3]
AA[4,2] := AA[2,4]   AA[4,3] := AA[2,2]   AA[4,4] := AA[2,3]
```

This is achieved as follows:

(a) The "active" elements of AA are moved from their original position within the array to the top left hand corner of a new temporary array, viz

5	7	9	0	0
8	11	14	0	0
11	15	19	0	0
0	0	0	0	0
0	0	0	0	0

(b) A horizontal rotate (of 1) is then performed upon this temporary array by using planar shift operations (i.e. the values shifted in at the edge of the matrix will be zero) to give the result

9	5	7	0	0
14	8	11	0	0
19	11	15	0	0
0	0	0	0	0
0	0	0	0	0

(c) This temporary result is then subjected to a vertical rotate (of -1), again by using planar shifts to give the final result of the rotation, viz

14	8	11	0	0
19	11	15	0	0
9	5	7	0	0
0	0	0	0	0
0	0	0	0	0

(d) This final result is then slotted into its correct position in the original array, viz

1	2	3	4	5
3	14	8	11	11
5	19	11	15	17
7	9	5	7	23
9	14	19	24	29

This series of steps is dictated by the nearest neighbour connection scheme of the ICL DAP and the fact that not all the processing elements are active in the rotation process. A run-time support routine INTEGER_ROTATE, written in DAP FORTRAN, performs all of the above operations and returns the rotated matrix.

3 Input and Output

The execution of any program on the ICL DAP can be considered as a three stage process

Stage 1: input the program data
Stage 2: run the program
Stage 3: output the results

Where Stages 1 and 3 are performed on the host machine and Stage 2 is performed on the DAP itself. Data is passed between the host and the DAP via parameter lists or, more usually, via common blocks. As Actus II permits input and output to be performed anywhere within a user program and READ and WRITE statements are not present in DAP FORTRAN, a problem is apparent. Output is achieved by translating, as far as possible, Actus II write statements into DAP FORTRAN TRACE statements. READ statements do however cause problems and at present no satisfactory solution has been discovered other than a user being forced to set up all program data by assigning values to appropriate variables at the start of a program run. The disadvantages associated with this are apparent.

4 Conclusions

In this paper we have illustrated how it has been possible to implement a high level structured parallel processing language for an array processor by

employing a less sophisticated high level language as the target language. Consequently the "time before use" of the language has been dramatically reduced resulting in earlier "hands on" experience for potential users.

Coupled with the relative ease of implementation, an evaluation of the usefulness of the language can be performed at an earlier stage in its development process. Therefore any identified problems can be ameliorated before the rigours of implementation proper commence.

Acknowledgement

This work was supported by U.K. Science and Engineering Research Council Grant No. GR/C/19819.

References

1 Cray-1 FORTRAN (CFT) Reference Manual. Cray Research, Inc. 1979.
2 CROOKES, D.: Implementation of a vector processing language on the Cray 1. BCS Parallel Processing Specialist Group Meeting, Imperial College London, September 1986.
3 ICL DAP: FORTRAN Language. ICL Technical Publication TP 6918 1981.
4 PERROT, R.H., LYTTLE, R.W. and DHILLON, P.S.: 'The Design and Implementation of a Pascal based language for Array Processor Architectures'. Journal of Parallel and Distributed Computing (accepted for publication).
5 PERROTT, R.H.: 'A Language for Array and Vector Processors' ACM TOPLAS. 1979, 1(2), 177–195.
6 PERROTT, R.H., CROOKES, D. and MILLIGAN, P.: 'The Programming Language Actus' Software — Practice and Experience Vol. 13, 1983, 305–322.

Appendix

The following is an example of Cannon's parallel matrix multiplication algorithm written in Actus II, and the corresponding DAP FORTRAN generated by the translator. The resulting code enables the two languages to be compared.

Actus II Version

```
program CANNON(INPUT,OUTPUT) ;

const
  N = 3 ;

var
  B, C, RESULT: array[1:N,1:N] of INTEGER ;
  K: INTEGER ;

index
  IS1, IS2, IS3, IS4: INTEGER ;
```

```
begin
{ input the arrays B and C }
for K := 1 to (N – 1) do
begin
using IS1 := (K + 1):N , IS2 := 1:N do
  B[IS1,IS2] := B[IS1,IS2 rotate 1] ;
using IS3 := 1:N , IS4 := (K + 1):N do
  C[IS3,IS4] := C[IS3 rotate 1,IS4]
end ;

using IS1 := 1:N, IS2 := 1:N do
  begin
  RESULT[IS1,IS2] := B[IS1,IS2] * C[IS1,IS2] ;
  for K := 1 to (N – 1) do
    begin
    B[IS1,IS2] := B[IS1,IS2 rotate 1] ;
    C[IS1,IS2] := C[IS1 rotate 1,IS2] ;
    RESULTS[IS1,IS2] := RESULT[IS1,IS2] +
                        B[IS1,IS2] * C[IS1,IS2]
    end
  end ;

{ output the result }
end { Cannon } .
```

DAP FORTRAN Version

In this translated version of the algorithm, the system variables which are necessary for stacking and unstacking e-o-p's have been omitted as they are not used in program.

```
      ENTRY SUBROUTINE ACTUS_PROGRAM
      CALL CANNON
      STOP
      END

      SUBROUTINE CANNON
      LOGICAL EOP(,)
      LOGICAL RMATRIX(,), CMATRIX(,)
      INTEGER B(,), C(,), RESULT(,)
      INTEGER K
      INTEGER LOOPSTEP8L, LOOPSTEP10L
      INTEGER LOOPEND9L, LOOPEND11L

      K = 1
      LOOPEND9L = 3 – 1
      IF (.NOT.(K.LE.LOOPEND9L)) GOTO 5
      LOOPSTEP8L = 1
    6 RMATRIX = .FALSE.
      CMATRIX = .FALSE.
```

```
      CMATRIX = COL_MATRIX(K + 1,3,1)
      RMATRIX = ROW_MATRIX(1,3,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 7
      B(EOP) = INTEGER_ROTATE(B,EOP,0,1)
    7 CONTINUE
      RMATRIX = .FALSE.
      CMATRIX = .FALSE.
      CMATRIX = COL_MATRIX(1,3,1)
      RMATRIX = ROW_MATRIX(K + 1,3,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 8
      C(EOP) = INTEGER_ROTATE(C,EOP,1,0)
    8 CONTINUE
      IF (K.EQ.LOOPEND9L) GOTO 5
      K = K + LOOPSTEP8L
      IF (K.LE.LOOPEND9L) GOTO 6
    5 CONTINUE
      RMATRIX = .FALSE.
      CMATRIX = .FALSE.
      CMATRIX = COL_MATRIX(1,3,1)
      RMATRIX = ROW_MATRIX(1,3,1)
      EOP = CMATRIX.AND.RMATRIX
      IF (.NOT.ANY(EOP)) GOTO 9
      RESULT(EOP) = B(,) * C(,)
      K = 1
      LOOPEND11L = 3 - 1
      IF (.NOT.(K.LE.LOOPEND11L)) GOTO 10
      LOOPSTEP10L = 1
   11 B(EOP) = INTEGER_ROTATE(B,EOP,0,1)
      C(EOP) = INTEGER_ROTATE(C,EOP,1,0)
      RESULT(EOP) = RESULT(,) + B(,) * C(,)
      IF (.EQ.LOOPEND11L) GOTO 10
      K = K + LOOPSTEP10L
      IF (K.LE.LOOPEND11L) GOTO 11
   10 CONTINUE
    9 CONTINUE
    C Print results
      RETURN
      END
```

Notes on the authors

C.W. Blatchford

Clive Blatchford is Manager, Secure Systems in the ICL Marketing and Technical Strategy Division. In this capacity he has staff responsibility for definition, development and marketing of security in all ICL products and services. Particular emphasis is placed on multi-level security operating systems, secure networking (including encipherment) and physical and electronic hardening of work stations; the development of secure Open Systems Architecture, with supporting standards and procedures, has been given a high priority in which to map specific customer solutions.

He plays an active role in both National and International standards organisations and has lectured extensively in Europe, North America and Australia; he is currently Vice-Chairman of the USA Department of Commerce/National Bureau of Standards Special Interest Group on Security. Previous to joining ICL he had senior management experience in dataprocessing and/or security with Xerox Corporation and with Chase Manhattan Bank.

J.B. Brenner

John Brenner joined ICL in 1961 after graduating from Cambridge. He held a series of appointments in technical support, software development, consultancy, account management and project management within the UK sales organisation. In 1972 he moved into development and managed reliability improvement programmes for all ICL large systems. Between 1974 and 1977 he worked on an advanced development of fault-tolerant distributed systems. He has had an active role in OSI standardisation and managed ICL's network architecture activity from 1982 to 1984. Since then he has been concerned with strategy development for distributed processing.

I.R. Campbell-Grant

Ian Campbell-Grant works as manager of the Advanced Products Sector within ICL's Office Information Systems Business Centre, ODS. He was instrumental in the development of the application-level standards currently used for interconnecting ICL office systems product and has been active also in developing the ODA standard specified here. He was the editor and led the task group that produced the first version of ECMA-101, the Office

Document Architecture standard, which was ratified in June 1985, and now chairs the ECMA Technical Committee working in this area. He is also the editor of those parts of the ISO and CCITT standards that deal with the document architecture.

Professor L.M. Delves

Professor Delves graduated from Oxford with a D.Phil. in Theoretical Physics in 1960. After appointments in New South Wales and at Sussex University he was appointed to the Chair of Computing Science (now Computational Mathematics) at Liverpool in 1969.

Dorothy M. Elliott

Following a childhood spent in Illinois, Dorothy Elliott read Maths at University in Ireland. She has been involved in Office Automation and networking for seven years and with the development and marketing of electronic mail since 1983. She is particularly interested in multi-vendor interworking to international standards and is a Director of the European Electronic Mail Association.

A.R. Fuller

Alan Fuller graduated from the University of Hull with an honours degree in chemistry and gained his doctorate from research into high temperature explosions – an ideal grounding for industry! However, this was not to be: he changed career after doing this research and joined the University of Oxford Computing Laboratory as a systems programmer, writing parts of GEORGE 3 and software for a front-end processor on CTL hardware. In 1977 he joined ICL and has worked in several parts of the company, from pre-sales consultancy through to his current position where he is part of the group in Marketing and Technical Strategy responsible for determining the company's Networked Product Line strategy.

A.J. Herbert

Andrew Herbert is Chief Architect of the Alvey Advanced Networked Systems Architecture (ANSA) project. After graduating in Computational Science at Leeds University he went on to complete a Ph.D. in computer science at Cambridge, where he remained as a lecturer and researched in various aspects of distributed computing and software engineering. He became Chief Architect of ANSA in 1985.

David King

David King has recently joined the Secure Network Systems group in ICL Defence Systems. He graduated at the University of Birmingham in Com-

puter Science and Software Engineering, and received a Ph.D. in Computer Science for work in cryptography and data security.

H. Gardiner

Helen Gardiner received the B.Sc. degree in Physics and Computer Science from the Queen's University of Belfast in 1985. For the subsequent year she was a Research Assistant in the Computer Science department at QUB where she implemented the Actus II to DAP FORTRAN translator. She is now employed in the Missile Systems Division of Short Brothers PLC in Belfast.

B.G.T. Lowden

Barry Lowden graduated from King's College, London in 1964 with a first degree in physics. He then joined the Plessey Company where he worked in Computing for eight years, being finally responsible for all systems development at the Ilford site. In 1972 he studied for an M.Sc. in Computer Science at London, which he gained with distinction. Since then he has been first a Lecturer and then a Senior Lecturer at the University of Essex. His main research interests lie in the areas of information retrieval, relational databases and query languages. For the last two years he has been funded by the ICL University Research Council to carry out work on natural language front ends to relational systems. The research team at Essex is currently developing a front end to support queries both on and about the data held in the database.

R. Lyttle

Robert Lyttle is a research officer in the department of Computer Science at the Queen's University of Belfast, having received the B.Sc. and Ph.D. degrees in Computer Science from the same university in 1980 and 1985 respectively. His initial research work was in the area of abstract data types and in the design and implementation of "small core" general purpose programming languages extendible in terms of their own language constructs. Since then his research interests have been centred on the use of the multi-tasking features of Ada for the implementation of parallel algorithms.

M.P.O. McCrann

Mike McCrann graduated in Computer Science at Liverpool in 1985, then studied for an M.Sc. in Computational Mathematics while working as a Research Assistant in the Centre for Mathematical Software Research. He now works for Evans and Sutherland Limited, Cambridge, as a Systems Analyst. The work reported here was carried out as part of his M.Sc. studies.

P. Milligan

Peter Milligan received the B.Sc. and Ph.D. degrees from the department of Computer Science, the Queen's University of Belfast in 1977 and 1987 respectively. After a period as a Programmer and a Research Assistant, he is

now a Lecturer in the Computer Science department at QUB. His research interests centre on programming methodology, language design, language implementation with specific application to concurrent systems, and software support environments. He is a member of the British Computer Society.

Professor R.H. Perrott

Ronald H. Perrott is currently Professor of Software Engineering at the Department of Computer Science, the Queen's University of Belfast, where he has been a staff member since 1969. His current research interests include programming languages for multiprocessor, distributed, and array and vector processor configurations and the design and analysis of parallel algorithms. He has also been involved in the design and development of several operating systems and software engineering projects. He is author of *Parallel Programming* published by Addison-Wesley, Co author of *Pascal for FORTRAN Programmers* published by Computer Science Press, Editor of *Software Engineering* and Co-editor (with C.A.R. Hoare) of *Operating Systems Techniques* both of which have been published by Academic Press. He is a member of the IEEE Computer Society and the British Computer Society.

D.J. Phipps

David Phipps graduated from Sussex in 1979 with a degree in Psychology. After spending some time working in the psychiatric services he took an M.Sc. in Computer Studies at Essex in 1984. Since then he has remained at Essex, first as a Senior Research Officer, participating in an ICL-funded contract to provide a natural language front end to databases, and more recently as a Temporary Lecturer.

Mrs. G.M. Ringland

Gill Ringland has been General Manager of Office Information Systems Business Centre since it started operation in May 1984. In this role she has responsibility for the office automation strategy, and for the specification, development, marketing and introduction of office automation systems including word processing, language translation, electronic filing, electronic mail, viewdata and the integrated office.

After a childhood spent in England and Australia, and a degree in Physics from Bristol University (where she met her husband), she learnt to program in 1964 on the early Atlas computers. Research in theoretical physics at Oxford University and two years at the University of California, Berkeley, were followed by 10 years in the UK and California with systems house CAP, two years with semi conductor manufacturer Inmos during startup and three years managing the European operation of the Florida based minicomputer company Modcomp.

She joined ICL from Modcomp in 1982, initially with responsibilities for database systems.

P.J. Robinson

Peter Robinson has been in computing for 20 years, the last 10 with ICL. He has a background that includes compilers for real-time programming languages and the design of mainframe operating systems. In the last four years he has worked in the Office Information Systems Division where in addition to his involvement in TOP his duties include acting as the ICL representative on the ECMA committee which published the first ODA specification, ECMA-101. He is the convenor of the ECMA task group defining a standardised Page Description Language.

A.N. de Roek

Anne de Roek graduated from the University of Leuven, Belgium, in 1979 with a degree in Linguistics, English and Dutch. After a year wih the Belgian Telecommunications, working in the field of library automation, she joined ISSCO, a research institute of the University of Geneva, where she participated in the development of a number of machine translation systems. In 1984 she obtained a M.Sc. in Computer Studies at Essex and has remained there, first as a Research Officer and then as a Lecturer in Computer Science. She has participated in two ICL contracts on natural language front ends to databases and is currently working on database front ends that rely on formal semantic representations.

C.B. Taylor

Colin Taylor is a Technical Strategy Manager in ICL's Office Systems Division. He is a graduate of Manchester University, and a Fellow of the British Computer Society. He has held senior positions in the development of many of the major well known ICL product ranges from 1900 onwards in both systems design and development management roles. He was one of the originators of the VME/2900 architecture and was chairman of the companywide group that laid down the supervisor interfaces and architecture definitions. His team created and developed the DME and CME systems, and he was Technology Centre manager responsible for the architecture of the ME29. Over the last few years he has been particularly concerned with the world UNIX scene, and ICL's UNIX technical strategy. He is a founder of the X/OPEN Group and the ICL representative on its Technical Management committee.

Professor R. Turner

Raymond Turner has been Lecturer/Professor in Computer Science at the University of Essex since 1973. His appointments during this period include Sloan Fellow, University of Massachusetts (1982); Visiting Professor, Uni-

versity of Rochester, New York (1982); Visiting Fellow at the Stanford Centre for the Study of Language and Information, Stanford University, USA (1984). A graduate of Queen Mary College, London, with a B.Sc. in Mathematics and a Ph.D. in Theory of Computation, he also holds an M.A. in Philosophy and a Ph.D. in Formal Logic from Bedford College, London. He has broad research interests which include various aspects of formal logic and the theory of computation. He has been invited to give talks in his subjects in the USA and in the UK and is the author of over 25 papers and publications.

ICL TECHNICAL JOURNAL

Pages contained in each issue

(1) 1–166
(2) 167–356
(3) 357–608
(4) 609–829

Subject index
Volume 5

E

M *Management*

P *Parallel systems*

Author index
Volume 5

X