A DEVELOPMENT OF REFERENCE MODELS FOR COMPUTER SYSTEMS

J E Holmes

Ministry of Defence, UK.*

ABSTRACT:

The use of Reference Models within the standards generation process as a tool for standards makers is discussed in the context of the overall product life cycle. Examples of the reference models being explored by the IEE for Computer System Architecture are described, along with some alternative approaches.

The importance of the reference model as an aid to human understanding and communication in the early stage of requirements elicitation is stressed. Also, the relatively unexplored dimension of management in the organisational hierarchy sense is illustrated in a layered reference model now being used to describe large complex real–time systems with 10^3 to 10^4 user–identifiable functions.

BACKGROUND:

The cycle of use for consumer and industrial products extends from the developing of the initial user's requirement into a specification against which potential products are assessed, through to the in–service use by the user which invariably amends and refines his/her ideas on what is truly required. Meanwhile the technology that industry can bring to bear is developed into new wondrous attractions that of themselves cause the user to update his expectations.

Into this merry–go–round of continuous change and upgrade comes the additional complication of economics; no financial director will permit new technology to be purchased for its own sake, yet neither does he want outdated and unfashionable computer equipment in his own department.

The individual who purchases a desk top home computer feels equally ham strung when the family budget falls short of upgrading to the latest HAL Turbo every few years. There is a general feeling that the pressure from the salesman and the resistance of the user are sufficiently mismatched that the short marketable life of a product is all to the benefit of the producer and not at all to the benefit of the consumer.

"Into this scene enters the standards maker who is believed to have promised to create calm from turmoil and turn the world into an environment where computer shall speak unto computer and all barriers to communication and understanding will vanish; Standards will rule; they multiply and flourish in abundance in this rich land. The haze that envelopes the user and the consumer was lifted, but now descends in a dense fog, and no direction (of progress) is clear."

There is therefore a crucial question to be settled; for whom is the benefit of this standard? If the reply is biased heavily towards either the producer or the consumer alone then it will fail; it must be seen to provide benefit for both these communities. Who then shall generate the standard; shall it be via a consumer–oriented group, an industry–oriented group, or via another, perhaps non–partisan, group? The professional institutions, such as the IEEE in the USA and the IEE in the UK are fora in which users and industry cooperate, which operate in a professional manner, and which support rationalised international standards making activities in the Computer and Software Engineering domains.

The IEE has a Computing and Control board which, via a sub–committee for computing standards, participates in the genesis of computer standards. During 1985 a new activity to aid in the standards creation process was begun via a Working Party on Computer Systems Architecture (CSA). The aim of the work is to create the mechanism for organising current and evolving standards in a way that enables their completeness and consistency as a set of standards to be evaluated. Of particular importance was the need to detect omissions in key areas needing coverage by standards. The organising mechanism now being developed is using reference models, vis à vis the graphical and associated syntactical descriptions of computer system, or elements thereof.

This paper describes several of the candidate reference models.

* This work is for the Institute of Electrical Engineers; the author is Chairman of the IEE Working Party of Computer System Architecture and may be reached through the IEE (C/O D. Marsh), Savoy Place, London, WC2R 0BL, U.K.

CH2549-4/88/0000/0065$01.00 © 1988 IEEE

Product life cycle: The over used term "cradle–to–grave" does describe the stages in the product life cycle at which standards can be applied and their benefits (and costs) accrued. Figure 1 shows the life cycle stages albeit at a macro level only. A conventional view of the use of standards is as part of the specification of the Users Requirement, with the aim of mandating a specific implementation feature.

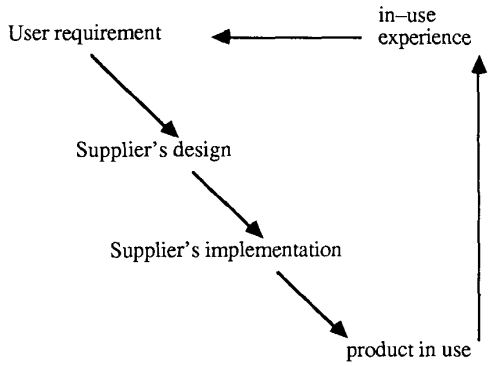


Figure 1 Product Life Cycle

There is another role for standards as part of the process of extracting, or eliciting, the requirement in the first instance. The problem that exists here stems from the different backgrounds of the erstwhile user and supplier. A common frame of understanding is required between them in order that communication of ideas is effective, for without it the completeness and self–consistency of the requirement statement are reduced, leading to a mismatch between the eventual product and the need of the user, and hence to a dissatisfied user.

Standards can assist in creating that common frame of understanding by providing a rigorous description of part of the requirement. The user must take the time and trouble to acquaint himself with the terminology and content of the standard, but once done it provides beneficial improvement in the communication with the supplier who is likewise articulate in that frame of understanding. The costs are outweighed by the benefits, or so we believe.

One mechanism used by standards makers which has proven itself useful in this communication between different communities is the reference model, specifically the layered reference model.

Reference models in the life cycle: The layered reference model is partly a graphical device and partly a rigorous description of the services to be provided and used by each component in the set of components comprising the modelled system. Protocols for the exchanges involved are also defined. The prime example of this form of model, the OSI standard, describes the cascade of layers that connect the applications functions to the physical device interconnection. While the user will be aware of the applications he would not normally be cognisant with the components contained in the other layers of OSI, bar physical identification of the interconnection cable perhaps. The uninformed user may therefore demand as part of his requirement, products conforming to OSI and thereby expect all interconnection problems to be solved. Whilst this is idealistic, it is no more than most mini– and micro–computer users expect of a backplane bus; to be able to interconnect at will the card–level products of different suppliers.

The naivety of this view stems from the problem of complexity, that is, from the problem of choice, as provided by the designer and thereby inherent in the product. Many users do not wish to have to select parameter values for a component before using it; the DIL switches on circuit boards are a parallel case, useful occasionally, perhaps, but most often a nuisance.

More useful is the delivery of the component with default settings compatible with those of other components, thereby enabling the system to be readily accessed and set to work without that vital a–priori knowledge. The problem of choice is eased by improved understanding on the part of the user; reasons for retaining design choice through into implementation must be made clear, eg. for improvement in the performance of a component via the by-passing of a high–load but optional feature in its operation.

The user's understanding of a complex design can be eased by the reference model constructs, and, importantly, the converse is also true. The designer's understanding of the user's requirement can be aided by the user's description of "Why" the problem for which a solution is required exists rather than "What" the user perceives as potential solutions to his (ill–defined) problem.

This elicitation of the requirement is likely to involve the topic of management in that services are demanded by/provided to superiors of/by inferiors in a management hierarchy.

An illustration of this use of a reference model is given in Figure 2 (after Morgan and Holmes, reference 2). The different levels of the organisation are shown to be operating in planning and execution loop timescales that vary monotonically between the limiting layers shown. Directives

from higher levels of command are a call on the layer 6 services, and so on down the hierarchy, until at layers 2 and 1 the direct control (layer 2) of the physical equipments (layer 1) for sensing and interacting with the environment during combat are shown. At each layer the users of the computer equipment involved have a need to understand and express their requirement in the Applications (OSI terminology) that exist at each layer, without needing to express the "infrastructure" requirement for OSI communication, for database support etc. It should therefore be understood that this 6 layer model (6 for economy reasons!) is actually orthogonal to any other model, such as OSI, which expresses the Applications/Infrastructure partitioning or the User /Computer partitioning (ie. man–machine demarcation and interfacing).

Computer System Reference Models: If the above partitioning were continued into even finer resolution of human–computer systems the detail on the computer side would lead to a problem. That problem is "What method does one use to describe a general computer system?" One can take the view of computers and their physical components, vis à vis specialist components for storage (RAM, ROM etc.), for processing (CPU), for communications (backplane, LAN etc.), or for User interfacing (VDU, keyboard etc.). Alternatively, and a current norm, is to consider some combination of the first three dimensions with a particular emphasis, eg. the Inmos Transputer with its specialisation for parallel processing. In fact it is now unusual at the board level not to have a significant mix of all three and, as the Transputer shows, this trend will continue downwards to components.

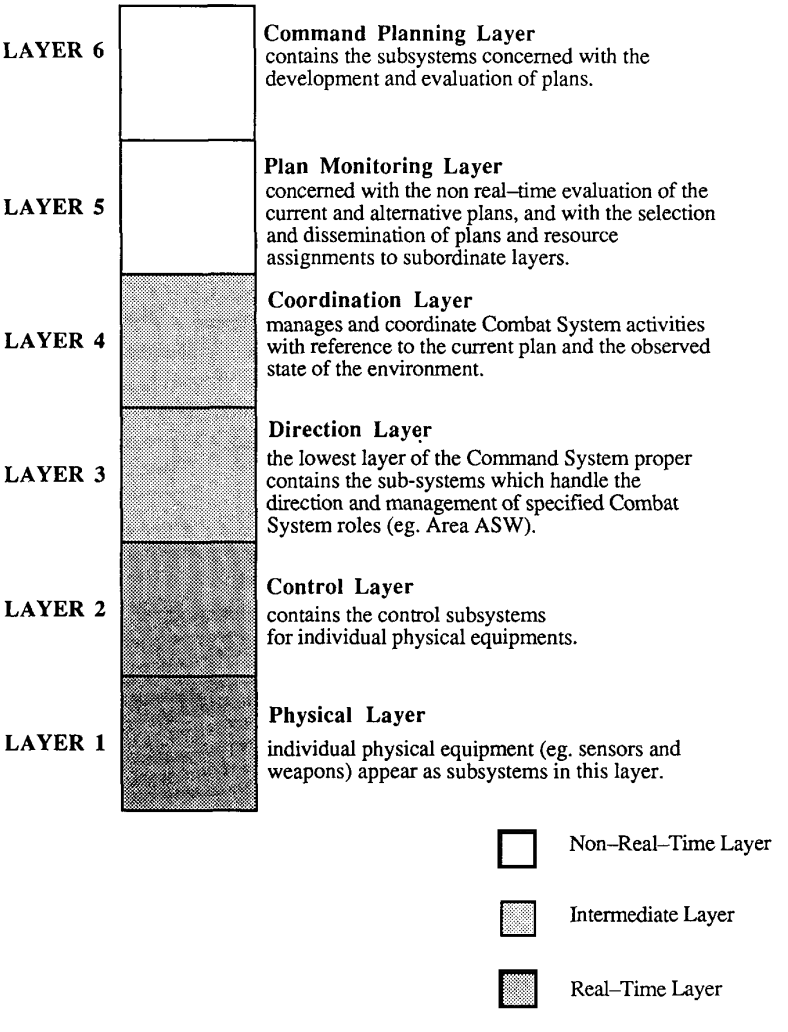


Figure 2 : A Reference Model For Command Systems

Alternative views can be taken using concepts of multi–dimensional form such as: space, time, location, where "space" refers to the algorithmic transformations, "time" refers to time, and "location" refers to physical location. For example, what one finds in this coordinate system is that some domains have no utility (Figure 3). Processors in the conventional sense provide the "space" transformations via their "impressed" functions that are embodied in the computer program. At the human–computer interface the goal is to provide a change in the form of the data between visual, audible or tactile and digital (or analogue) electrical without otherwise processing it in the "space" sense. The top left domain is where all practical processing systems reside since finite time is taken for "space" changes.

The above is not a layered preference model but is included to show that alternative approaches merit consideration.

What follows is a résumé of the CSA layered model approaches currently under development.

While the scope of the general architectural principles required for computer systems is very broad, this paper is primarily concerned with computer systems comprising terminals, computers and associated devices and the means for transferring information between such computer systems. The qualification of standards for computer systems in terms of computer system components, interfaces, services and protocols are discussed.

SCOPE OF THE CSA WORK :

The following aspects are specifically included within the scope of the Computer System Architecture work:

1) Single system views of the following interfaces, together with the services provided and the protocols employed at each
- the man-machine interface
- local peripheral interface
- remote communications interface (OSI) to support distributed operation

2) Services provided within a computer system such as:
- databases
- operating systems
- generic application services (eg. real-time clock)
- application generally
- distributed operation

3) Entities contained within a computer system such as:
- processors
- storage (main and backing)
- communications (eg. LAN, WAN, Terminal, Buses)
- transducers (ie. I/O peripherals)

4) The following aspects of Computer Systems are treated as external to the Reference Models
- software engineering
- distributed systems interconnection
- application specific concerns
- documentation of a computer system

The scope is illustrated diagramatically in Figure 4

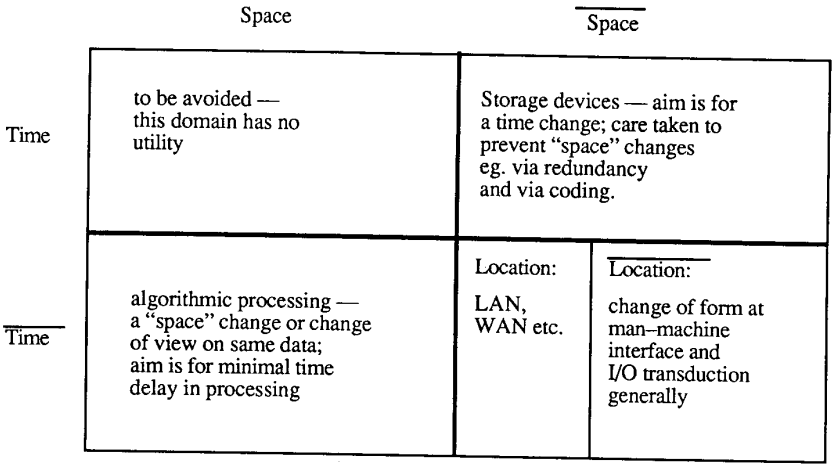


Figure 3 : Categories of Change, using 3 dimensions — space, time, location

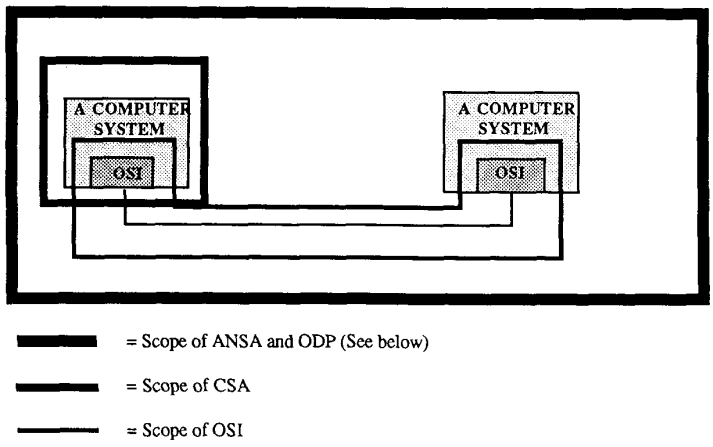


Figure 4 : Scope of the CSA work

The intent is that computer systems encompassed by this Reference Model are somehow restricted in geographical scope. The architecture does not include the distribution aspects of distributed systems, ie. the architecture does not explicitly contain OSI communications within itself, but regards OSI communications as an external interface to other such systems. However, multi-processor computer systems that form a single computer system from some viewpoint, local interfaces such as buses, and interfaces to local peripherals are included.

The related ANSA and ODP activities are illustrated for comparison.

ANSA : The purpose of the Advanced Networked Systems Architecture (ANSA) project (Reference 3), a collaborative venture involving the UK Alvey Directorate and seven major companies in the information technology field, is to develop and achieve wide take-up of an architecture that enables the construction of truly integrated multi-vendor distributed systems.

The specific goals of the project are:

1) To anticipate the architectural needs of future advanced networked systems while taking full account of current products and standards; to develop an architecture that enables the construction of truly integrated multi-vendor distributed systems;

2) To develop an architecture that spans the range of application areas for networked systems and which permits selective profiles of the architecture to be made to meet the needs of individual applications; to place the architecture in the public domain in the form of accepted International and European standards;

3) To demonstrate the architecture in use either within the project, within the community of ANSA collaborators, or by organisations outside the ANSA collaboration.

ODP: A New Work Item in ISO/TC97/SC21 on Open Distributed Processing (ODP) has recently been accepted. Its aim is to develop a reference model to encompass standards for the execution of tasks over two or more Open Systems. It is particularly concerned with the distributed nature of Open Systems, and the impact of such distribution on the standards for facilities such as graphics and databases, as well as communications (OSI).

METHOD 1 - Interface Oriented Functional Decomposition Modelling Technique

This method concentrates on interfaces. The first step is to identify the interfaces to a computer system. The computer system is regarded as a traditional "black box" and as being as fully specified as it needs to be for this purpose by the specification of its interfaces. The interfaces are distinguished by purpose rather than structure (ie. the structure of an identified interface may, or may not, bear some commonality with the structure of other identified interfaces).

On this basis one might identify three generic interfaces. These are the interface to other such (remote) systems, the interface to (local) peripherals and the interface the human user. These three interfaces are depicted as interfaces 1R, 1L and 1H respectively in Figure 5.

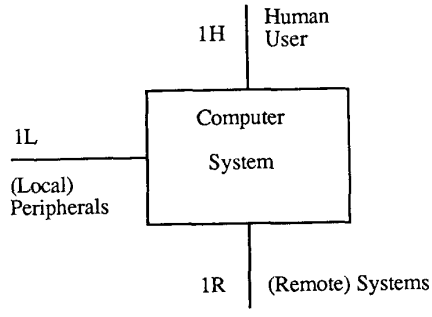


Figure 5 : Functional decomposition; Stage 1.

These are all level 1 (ie. external) interfaces. Interface 1R can be regarded as being specified by OSI. Interfaces 1L and 1H require definition and specification although there are probably existing standards that have a bearing and need to be accommodated (such as the SCSI–Small Computer System Interface or IPI–Intelligent Peripheral Interface for 1L and the Operating System Command and Response Language (OSCRL) work for 1H.

The next stage is to functionally decompose the system into a number of level 2 components and interfaces as depicted pictorially in Figure 6 which shows a hypothetical arrangement.

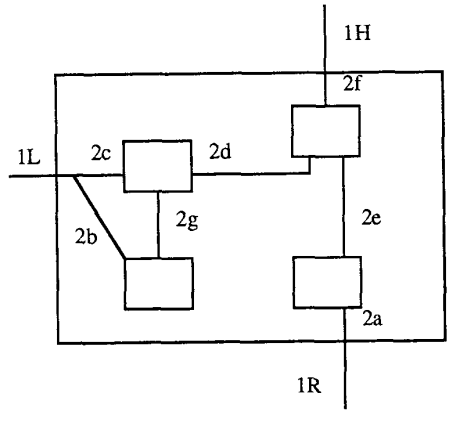


Figure 6 : Functional decomposition; Stage 2.

The number of distinguishable level 2 components and interfaces necessary would need to be determined. A level 2 interface may connect level 2 components as does 2d in Figure 6. (note: interface 1R performs this function for level 1 components). Alternatively they may wholly (as illustrated by 2f to 1H) or partially (as illustrated by 2c to 1L) provide a level 1 interface.

Subsequent stages of the process consist of subdividing each level 2 component into further level 2 components (in that they use level 2 interfaces) or into level 3 components. Again, the level 3 interfaces either connect level 3 components or provide, wholly or partially, the level 2 interfaces. This process of subdivision can be continued to appropriate depth. Different distinguishable components may be subdivided to different depths.

This method is the traditional block diagram decomposition method used in system specification and design.

METHOD 2 - Peer Entity Model

Method 2 is similar to Method 1, but emphasises the provision and use of services, and the peer relationship of entities.

The single lower surface of an entity makes use of the services provided by the entity below, or connects to one or more Physical Media for data transmission. The single upper surface provides services to the next entity up.

An entity may be pictured with as having an active face to model its peer to peer interactions.
(Note: passive surfaces may be used in drawing diagrams just to make the shape geometrically complete and the diagram easier to visualise).

A set of entities can be visualised as forming a "tower" that provides an overall service to an Application Process. An Application Process can use the services of more than one "tower", and can have a peer relationship with other such Application Processes.

These ideas are illustrated (for illustrative purposes only at this stage!) in Figures 7 and 8.

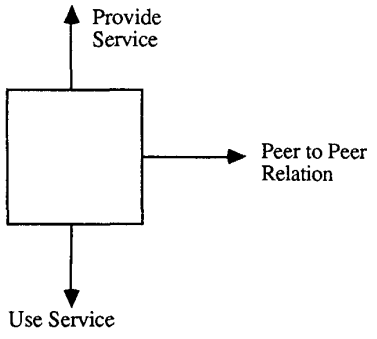


Figure 7 : Provision and Use of Service

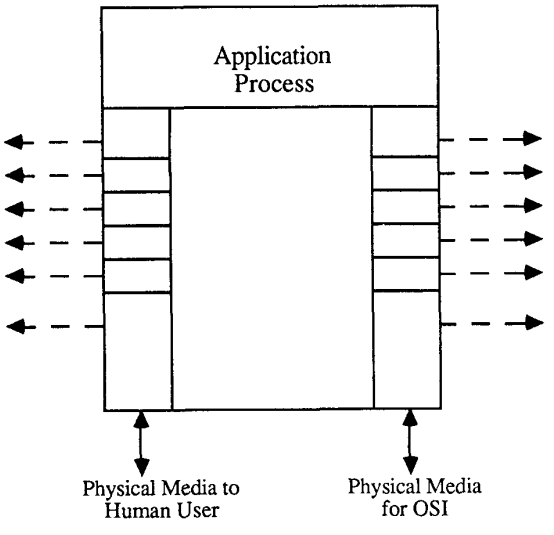
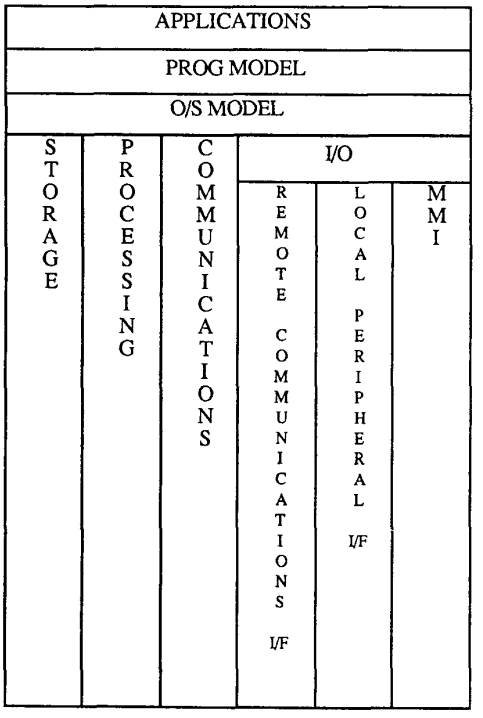


Figure 8 : Extension of Service Model to OSI and beyond

METHOD 3 - "Filing Cabinet" Model

Analysing existing and foreseen computers one could divide their functionality into the following areas:



O/S = Operating System I/O = Input and Output
MMI = Man Machine Interface I/F = Interface

Figure 9 : Functional or "Filing Cabinet" Model

Each area could be related in a defined manner to the other areas, as perhaps implied by Figure 9, or could be regarded simply as "filing boxes" of standards. Each area could be divided into Operating Principle and Hardware aspects, each area could be sub-divided. For example, the next few levels down for the Programming Model area (or "box of standards") might be:

GENERIC	PROCEDURAL	OBJECT ORIENTED
MODEL	ADA, Occam models	Smalltalk model
PARTICULAR LANGUAGE	ADA, Occam	Smalltalk

Any particular computer system could then "conform" to one (or more?) of the standards in each area (or a particular combination of standards taken from a path down the sub-levels of that area).

METHOD 4 - The Resource Based Model

Note: This method may be regarded as a particular application of method 2.

This model is based on four fundamental resource sectors and is shown diagramatically in Figure 10. The lower layers of each sector refer to the more "physical" aspects, and progression upwards gives increasing levels of abstraction.

Computer Systems Resource Sectors: In this model the resources of information processing systems are divided into 4 general categories according to their principal purpose in support of real application enterprises:

1) Terminals interface the information processing system to the real physical environment of people (eg. keyboard, display), machinery (eg. robot vision sensor, electromechanical effector), and the natural world (eg. security alarm sensor, voice synthesis annunciator).

2) Storage provides memory (eg. semiconductor RAM, optical disk storage).

3) Communications provides the means to exchange data with other systems based on data transmission hardware (eg. interconnect bus, communications modem).

4) Processors and management provide the means for representing, executing and managing AP's (Application Processes) based on computing machinery (eg. computer processor, operating system).

Typically every real computer system may contain resources from some or all of these general resource categories. A real computer system executing an AP can therefore be represented graphically as a domain containing the AP, above a layer of resources divided into 4. Each layer contains the elements of the computer system which provide the AP with its view of the information processing resources in one of the 4 categories.

This reference model is not intended to depict the way in which any particular real computer system is implemented. The sole purpose is to describe the relationship of computer systems standards to which the system conforms. Therefore the physical arrangement of hardware and software that comprises any particular real computer system, including the Local System Environment (LSE) which is the "glue" that ties together the manufacturer's implementations of the computer system standards, is not shown in the model. Every resource layer within every sector has a boundary with the LSE and receives its local operational information and services through the LSE.

METHOD 5 - "2+4" Resource Based Model

The "2+4 approach" suggests that:
a) All standardisation involves the implicit or explicit development of an application domain specific model, which when refined into detailed data structures and operations on those data structures frequently requires support by terminal hardware, storage devices, languages, and communications. It should be regarded as largely independent of the hardware sector, no matter who develops it (consider, for example, Office Document Architecture).

b) The range of (human) user services appropriate to operations in the areas covered by the model should be explicitly recognised.

c) Support for the two main pillars; the sector–independent model and the recognised user services, takes some or all of the following four forms:

1) Remote access (communications support), consisting of:
   i) Expression of the data structure and operations underlying the model in terms of communication–oriented service primitives and parameters which allow both remote access to the structure and remote operations.

   ii) Protocol support for i), based on progressive refinement using the 7–layer model.

2) Language bindings, consisting of:
   i) The data–structures and operations underlying the model in terms of a specific language oriented set of facilities; this may reflect more or less detail in the model, depending on the level of the language.

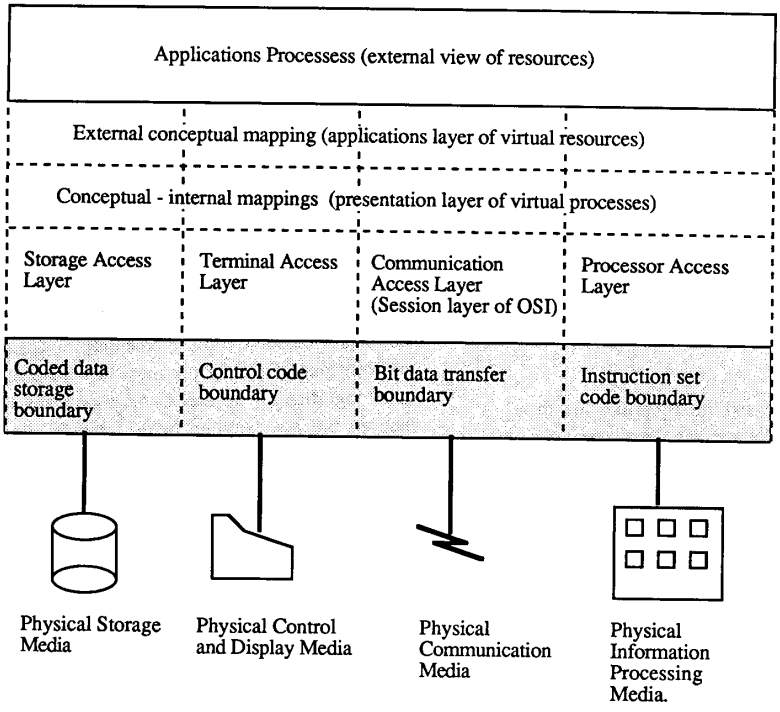


Figure 10 : Resource Based Model

For example, a programming language might reflect the full detail, an OSCRL (Operating System Command and Response Language) might reflect less.

ii) All language support for i), based on decisions on

semantics, syntax, defaults, macro facilities, scope rules, etc.

3) Terminal device considerations, consisting of:

i) The categories of user service and the level of support; that is, support by general purpose terminals or by model–specific terminals.

ii) Detailed mappings of model data structures onto display devices, and support for operations using keyboard features; this involves decisions on windowing or not, escape keys, keyboard functions and layout, screen or paper size, resolution, control codes, etc.

4)Storage considerations, consisting of:

i) Representation of parts or all of the features of the model in storage: primary memory, disc, tape, etc.

ii) Progressive refinement of this representation, involving decisions on structuring mechanisms (counts, pointers, escapes), transparency considerations, word length issues, character coding issues, integer and floating point representations, physical tape or disc size, encoding (electrical level) techniques, error detection/correction capability, etc.

Where to start? : It would be very wrong to develop a); (the model) and then b) (the (human) user-services) without any regard to c) (the supporting facilities). Many would argue that b) (the (human) user services) should be done first. Another viewpoint would regard it as not possible to specify services in detail without the terminology and concepts developed in the model, making the model a necessary first step. Still others would argue that, historically at least, existing facilities

(particular terminals, particular languages, particular protocols) are developed first, and abstract modelling and even user services come second, or sometimes not at all! This viewpoint regards all three aspects as important, and considers that a detailed interaction between these aspects should occur in the development of any particular application area.

Terminals are designed to meet what are seen as probable (human) user needs, user needs become clearer (eg. screen editing) when hardware to support them is available; models are frequently developed after the main standardisation is in place.

These, then, are the two main pillars of the 2+4 approach; models, and (human) user services and requirements. The four supporting pillars are communications, storage, terminals, and languages.

The central pillars are surrounded by detailed specifications of data structures, operations, allowed sequences, and ranges of values.
Each of the four supporting pillars has its own application-independent internal structure, together with application area-specific work. The 7-layer structure of the communications pillar should not be thrust upon the other pillars.
The diagram in Figure 11 illustrates the method.

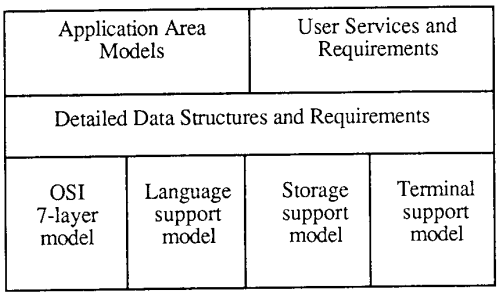


Figure 11 : The resource based model

METHOD 6 - Performance Based Model

This is a final contrary view of a "model" based on performance.

The problem with all models based on functionality is that they have no bounds: the same principles apply at every level, so it is not possible to distinguish one level from another, except by resorting to facts outside the model.

Information hiding is excellent for dealing with functionality, as it allows the externally visible behaviour or effects of a unit to be distinguished from the means by which these are achieved. But, it is not possible to hide performance, if the means of achieving a certain effect depends on defined behaviour of some more elementary constituent, then, the performance of the former must depend on the latter. This most significantly applies to the time taken to carry out an action, but also applies to the storage space needed for programs and data.

Models should not be solely logical to the exclusion of technological influences. Logic suffers from the same boundlessness as functionality. For practical purposes (which is why a reference model is required), we must take account of technological limitations and possibilities. This means that the details of the model might have to be changed in the light of critical technological shifts. We should not be surprised or alarmed at this; otherwise why do we try to make technological advances? The model can be sufficiently robust to contain the technology we can envisage.

Thus we also need a model based on performance, specifically the time taken to carry out typical actions. There will be groups of actions "at the same level" as with abstract data types, whose times should be comparable in magnitude. The model has several levels or layers, which differ in the times of their actions, on a logarithmic scale.

For the coarsest model, we consider timescale variations in steps of a thousand, centred round one second.

1 Gs = 10^9 sec = 30 yrs = human career, system service life
1 Ms = 10^6 sec = 12 days = major update installation
1 Ks = 10^3 sec = 16 mins = non-automatic repair
1 s = user interaction
1 ms = 10^{-3} sec = disk access, communication link
1 us = 10^{-6} sec = machine operation, member access
1 ns = 10^{-9} sec = gate operation

Note: how the technology shifts with the timescale: human actions down to one second, mechanical and electrical transmission from seconds down to milliseconds, electronic

from microseconds down to nanoseconds. Software can apply to all of these (in one sense) or to the microsecond to megasecond region (in a more limited sense).

It is important to distinguish two ways in which we may use the word "software" (and the related word "program"). In a general sense, a program is a formal description of an information handling process, without commitment as to the technology that may be used to carry out the process. In a more limited sense (which is often assumed without recognising that the sense is different), a program is a pattern of bits in the memory of a computer suitable for interpretation by other parts of the computer. These other parts may be hardware or software, but can be defined by software in the more general sense above. Thus the semantics of software is essentially recursive (as software defines its meaning), and a program does not normally mention the time or other resources it needs, since different interpreters could act differently: this is what we mean by portability.

High–level languages have given us software in the general sense, so that programs (in the general sense) are capable of automatically being transformed into programs (in the limited sense) for a variety of computers, with different architectures and timescales. Naturally, the rate of execution of a program, depends on the rate of the operations of its interpreter. If the interpreter is the conventional fetch–execute cycle of a computer, we have normal machine code execution.

The operations of such software cannot, with current technology, be faster than multiples of microseconds. Microcode is software that is executed faster using gates more directly, if the interpreter is a person (such as a user carrying out a standard operating procedure) then the timescale for the action will be multiples of seconds.

This model uses the "dependency" relation, namely A depends on B in the sense that B is used by A to achieve its externally visible effects and behaviour, and contributes to its total resource usage. What are the relations implicit in the other models?

CONCLUSIONS:

The product life cycle involves technological change. Manufacturers are oriented to exploiting that change, users of their products aspire to the latest yet are restrained by cost and other considerations. Standards provide a reuse of components and a compatibility dimension to the ever moving product life cycle.

Standards for computer systems and particularly for major components are under development in many countries, and so give rise to an international coordination problem. One aid to that coordination would be a common "currency" in which the various developed, developing, and proposed standards could be measured and assessed for scope and content.

The IEE's Computer System Architecture work is aiming at the generation of just such a "currency" via the use of reference models for computer systems. Some such models have been described here.

Acknowledgment: The efforts of the IEE's CSA Working Party, and in particular Tony Fletcher (British Telecom), Mike Fletcher (Ferranti) and John Miller (University of Birmingham) are hereby noted as crucial to this work.

Further participants are always welcome, distance is no object since electronic mail facilities are in use, please contact the author at the address given.

REFERENCES:

1.ISO 7498-1984
Basic reference model for open systems (BS 6568:1984)
Interconnection, ISO 1984.

2.Holmes J E and Morgan P E. (1987),
"On the Specification and Design of Implementable Systems",
Joint Directors of Laboratories Symposium on C^3,
Washington DC.

3. ANSA Reference Manual, Release 00.03, June 1987
ANSA, Cambridge, UK.