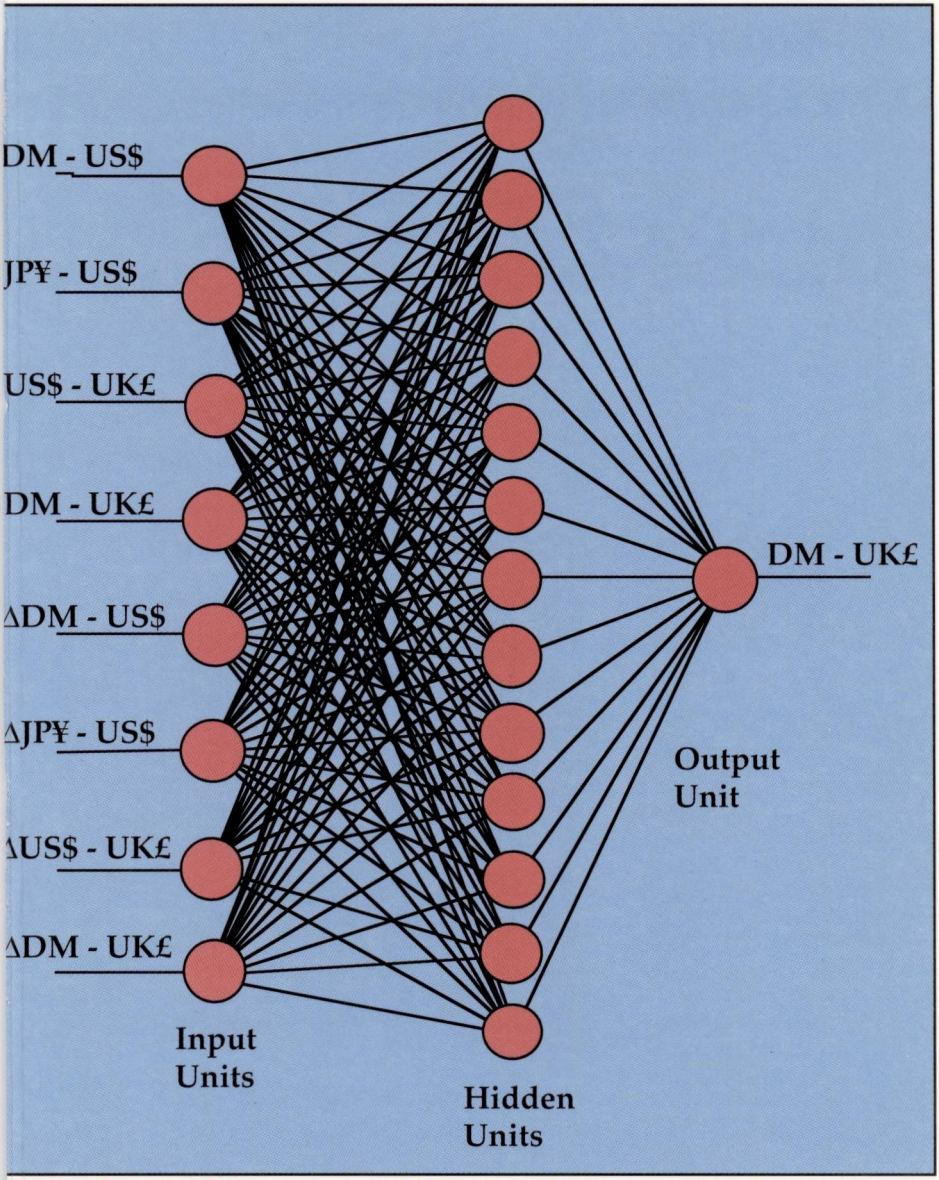
Volume 11 Issue 2. January 1997 Edition



ICL

The Systems Journal

ICL Systems Journal

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

The views expressed in the papers are those of the authors and do not necessarily represent ICL policy.

Published twice a year by Research and Advanced Technnology, ICL, Bracknell.

1997 subscription rates (including postage & packing):

	UK and Europe	Rest of World
Annual subscription	£72	$144
Single issues	£43	$86

ISSN 1364-310X

ICL Systems Journal

Volume 11 Issue 2

Contents

Front cover: A multi-layer Neural Network. See the paper, " Short-term currency forecasting using Neural Networks," in this issue.

Editorial

The editor wishes to apologise to the Journal's readers for the delayed publication of this issue. This issue (Volume 11, Issue 2) should have been published in November, 1996 but is now being dated, January, 1997. No change in policy should be inferred from this new date and publication of Volume 12 will proceed as planned for May and November, 1997.

Two invited papers written by academic research workers, who have been associated with ICL projects in the past, are included in this issue. The first by Professor Eason and his colleagues is concerned with the problem of generating requirements for the specification of computer systems. The paper is based on the results of a five year ESPRIT II research programme. The second paper is on Neural Networks, which have recently come of age and now show real promise in several application areas. In this issue Chris Hinde and his colleagues from Loughborough University give an overview of the subject, dwelling particularly on how trained nets can be used to derive complex rules that may be expressed in boolean logic. Owen Evans, of Research and Advanced Technology at ICL Bracknell, then shows how Neural Net technology may be used in currency forecasting.

In this issue, there is also an obituary of Gordon Scarrott, who was, for many years, one of ICL's senior research managers. Gordon's career spanned the first thirty years of the British computer industry to which he made many significant technical contributions.

V.A.J. Maller

The Year 2000 Problem

Janet Pavelin

Enterprise Consultancy, ICL, Lovelace Road, Bracknell, Berkshire

Abstract

Allowing only two digits to represent the year field suggests that many computers and their applications will fail when calculations and comparisons are performed on dates after 31 December, 1999. The widespread use of dates implies that every organisation using any Information Technology is a potential target for date-related problems. This paper discusses the imperative nature of the problem and the critical need for every organization to make rapid progress in identifying and resolving the issues.

1. Introduction

Chief executives across the globe face a stark choice: either to risk near certain business failure or to divert many man–years of effort to a task with an immutable deadline for no competitive advantage. The reason for this state of affairs is that hidden away in large numbers of computer systems are date routines which will fail fatally when they try to process the year 2000.

This paper discusses the issues involved and sets out ways of resolving them before it is too late.

It is widely held that the Year 2000 problem is a simple technical issue which, if it arises at all, will not do so until the end of 1999. This is a misconception. Firstly, the problem is anything but simple; one can make all one's own systems proof against it but still fall foul of problems in the systems of suppliers and customers. Secondly, many systems, for example, in business forecasting and contract management, are already encountering difficulties as they try to process dates beyond 1999.

A typical medium sized business provides an illustration. Having spent 8000 hours reviewing its IT systems, the company found that two thirds of them, consisting of 164,000 components, had problems with the year,

2000. The estimated effort to complete the necessary changes was 230 man–years.

Unfortunately, there are no short cuts and none are likely to emerge. Most computer applications handle dates but finding where dates occur can be difficult. Once found, amending the code may be straightforward but testing these changes can be a major problem. Getting to grips with this issue is now a matter of urgency. The longer it takes an organization to get started, the more it will need outside help and the more that help will cost as the end of the century approaches.

2. The Problem

The root of the problem is the format in which dates, particularly years, are stored. Most year numbers are held as two digits – such as '96' for 1996. The century number is then inferred to be 19. Credit card expiry dates, for instance, are held in the form MM/YY, so an expiry date of October 2000 appears as 10/00. Subtracting today's date, for example, 10/96, from that expiry date would yield a negative number, implying that the card is invalid.

Date stored with year = YY	**Date stored with year = CCYY**
Year of birth= 63	Year of birth= 1963
Age in 1996= 96-63 = 33	Age in 1996= 1996-1963 = 33
Age in 2000= 00-63 = -63	Age in 2000= 2000-1963 = 37

Figure 1: The root of the problem is the date storage format

One way round the problem is to infer the year 2000 from 00, rather than the year 1900. An alternative is to arrange for year numbers to be formatted as four digits rather than two. Both these solutions require full agreement amongst all the participants involved, such as card suppliers and network providers. In fact, the first solution is the one being implemented by the credit card industry. The difficulty is that the expiry date on a card can be checked anywhere in the world, so that the financial organizations which produce them cannot safely issue

cards with expiry dates in the next century until everyone (retailers, Automatic Teller Machine suppliers, etc.) has made the necessary changes.

2.1 Trouble ahead

Much the same problem is already emerging in sell-by dates on food and pharmaceutical products with a shelf life of four or more years. Where such dates are read and interpreted by humans, no confusion arises, since '00' in a date on a package label is unlikely to be interpreted as 1900. Machine reading, however, is a different story and, when faced with the year '00', many systems will crash. Worse still will be those that produce invalid results that then go undetected.

Consider, for instance, property insurance contracts: it may be impossible to specify dates beyond 1999 without error, or at least without placing records in the wrong sequence. Robotic warehouse systems will have to be reprogrammed to ensure that they behave as intended and PCs of all kinds will have problems. Setting the date to 31 December 1999 and the time to 23:58, followed by switching off and then on again after three minutes, will cause 70% of all PCs in use today to reset the date to 4 Jan 1980 (a base date for PC operating systems), rather than 1 January 2000. If left running past midnight, many PCs will roll over successfully to the year 2000, only to revert to 1980 after they have been switched off.

Another problem arises from the fact that the year 2000 will be a leap year. Century years are leap years when they are divisible not only by four but by 400 as well. The year 1900 was not a leap year, since 1900 is indivisible by 400, but 2000 is divisible by 400 and is, therefore, a leap year. Routines that calculate the day of the week from formulae based on two digit year numbers have, of course, to account accurately for leap years but they will fail to do so after 1999 unless they infer 20YY from the YY number.

Some of the problems in store in the year 2000 are highly esoteric. One example is the use of the last two digits of the current year in random number generators in lottery systems. Imagine the consequences if 00 is used as a multiplier!

3. The Cost of solving the problem

Estimating the cost of resolving problems with the year 2000 problem is difficult and imprecise. Gartner Group, a US research company, has

estimated the cost worldwide at an incredible $300 – $600 billion. This is roughly the same as the GDP of Spain; or put another way, it is rather more than the world's annual expenditure on Information Technology. Even if Gartner Group's estimate is overstated by an order of magnitude, which is unlikely, the numbers are still very large.

What might a given organization have to pay? One insurance company identified more than 400 date fields in 1400 large programs. Although only 5% of the code was affected, it was spread across 90% of the programs and the corrective effort was estimated at 53 man-years. Another organization spent four months doing a pilot investigation of three systems, from which it concluded that the whole job, for all 1500 programs, would cost 59 man-years of effort. These estimates, moreover, did not include any testing.

Making precise estimates, however, is difficult. A third organization, in a similar position to the two mentioned above, has predicted anywhere between 40 and 100 man-years for the job. So the answer to the question, how much will it cost me, is bound to vary, but it could be anything from 10% to as much as 50% of the annual IT budget between now and 2000.

There are many contributory factors relating to the imprecision of estimates. One is the wide scope of the occurrence of the problem. Data, screen displays, reports and executed code may be interchanged across departmental boundaries. Many organizations exchange information with their suppliers and customers, with banks and the Inland Revenue, and with partners elsewhere in the supply and distribution chains. Finding where trails of information interchanges lead is complex. See Figure 1.

It is particularly difficult to estimate the effort required to correct software packages that have been bought from outside vendors and customized in-house. Packages for which the source code is unavailable create their own problems and it might become necessary to obtain from an escrow agency the source code of a package whose owner has not survived to upgrade it to be proof against the year 2000. Indeed, one of the first things that an Information Services (IS) unit should do in cost estimating is to establish its suppliers' stances on upgrading their software. Vendors should state as soon as possible which versions will be proof against the year 2000 and when.

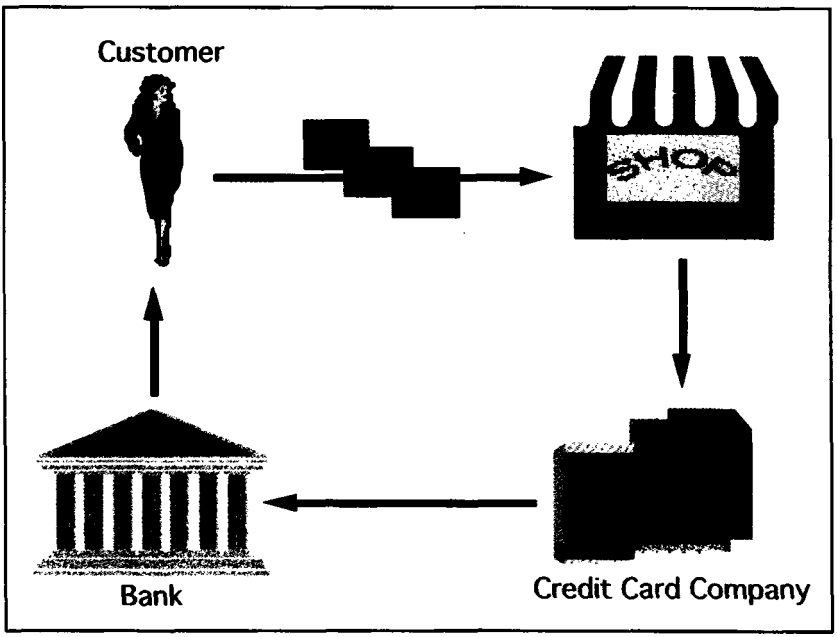


Figure 2: Moving Information around adds to the problem

4. Solving the Problem

There are four main steps to be taken in solving the problem. These are shown in Figure 3.

4.1 Identify the scope of the problem

It is best to determine the level of exposure to the problem before failures begin to occur. The first step is to convince those responsible of the need to spend money on an investigation in the first place. The argument for doing so is simple: failure to solve the problem will put the entire business at risk. Moreover, the longer it is left, the more expensive the solution will be. There really is no alternative to investigating every application on every platform. This is because all applications, including commercial off-the-shelf packages, are vulnerable. Moreover, their date functions may depend on the language in which they were written; so the language compiler will need to be upgraded first.

Having broadly decided the scope of the problem, the starting point is to create an inventory that includes everything relating to all those applications likely to be affected. The first activity, therefore, is to

find and count the elements that will make up the inventory. Everything relating to the applications will have to be included, so that, as well as the source code and data files, the screens, the reports and the job control scripts will need to be listed. Information about the hardware platforms themselves, as well as the software on them which supports the applications, will also be required.

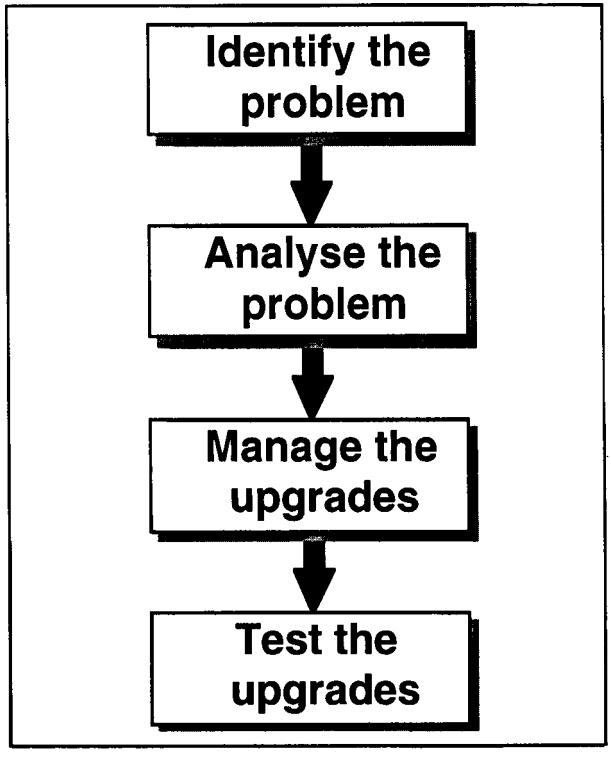


Figure 3: There are four main steps in fixing the problem

Unfortunately, few organizations keep accurate records of all their business applications, in particular those that are PC based. Yet many organizations are as critically dependent on local PC applications, as they are on centrally managed systems, and there are bound to be problems with the year 2000 in such applications. The need to investigate PC applications is, thus, just as great as for mainframe systems and, potentially, there may be an even greater need to help PC users check and, where necessary, upgrade their applications.

Even although some industry standard PC software already provides facilities to cope with the year 2000, it has been all too easy for users to develop PC applications, whether in a fourth generation language or in any spreadsheet package, which will cause problems. Many local applications also use enterprise data, so if date formats need to be changed at the corporate level, these will have to be taken into account at the PC level, as well.

The difficulty is that IS is often unaware of user-developed PC applications, so that it will be necessary to ensure that business managers take responsibility for upgrading applications developed by their own staff. Since those doing the remedial work will not be IT professionals, their need for help, even in understanding the scale of the problem, is likely to be greater than that required for those involved in 'fixing' professionally developed systems. Overall, the costs of modfifying the PC applications could be the largest part of the total costs.

4.2 Analyse the problem

Once the inventory has been compiled, work can commence on estimating the likely cost of making the organization proof against the year 2000. One approach is to conduct a pilot investigation of a representative subset of applications. With appropriate guidance on what to look for, a knowledgeable member of the support team can probably produce a reasonable estimate after inspecting the applications and their interfaces, although the outcome should be treated with some caution. Carrying out a complete upgrade on a selected application will help to refine the estimated cost. This is also a good way to rehearse the subsequent steps involved in solving the problem.

Whether or not a pilot study is conducted, a full audit of what is in the inventory will be necessary. This will enable the business impact to be assessed; e.g. which systems are key to the business, where are the interdependencies, which ones should be replaced instead and, most important of all, which ones will be the first to use dates which will fail?

The full investigation will be a laborious process involving line by line investigation of all the code and data, but it is the only certain way of making a confident assessment of the problem and obtaining a realistic estimate of the cost of the changes. This can be done manually, or one of the many parsing tools can be used that are able to find lines of code (usually written in COBOL) containing what appear to be dates. These

tools can help considerably in reducing the time taken to find lines with dates in them.

Unfortunately, these tools also detect lines of code which do not need to be changed, so every line identified must be investigated in detail. Expert advice and guidance about some of the more obscure date handling practices can be helpful at this stage.

Once found, the errant date handling code and data need to be corrected. Adopting a standard approach across the organization for handling dates would obviously be advisable and has a bearing on the cost estimates, since it affects the methods used for upgrading. For example, it might be decided that all date fields on screens, data bases and reports throughout the organization will be expanded to contain four digits for the year and, at the same time, that a common set of date routines will be consolidated in all date processing.

Although the obvious long term solution is to expand the data formats and add code as described above, the co-ordination of the changes and testing will mean higher costs. As this looks so daunting, many organizations are doing their best to avoid it where they can, by making code changes which use *century inference* instead.

Century inference means dividing the century into two parts, with say the years between 00 and 49 assumed to mean 20XX and 50 to 99 to mean 19YY. As long as all the dates that the application has to handle lie comfortably within a single century, it will be possible to choose a "pivot" year, such as 49, for which there is no chance of a date being computed incorrectly. Selecting an appropriate pivot year is critical. It would be unwise to choose 19 as a pivot year, for instance, if there is a chance that the program might, in the near future, have to look forward to the year 2020 and beyond.

Moreover, *century inference* cannot be used for applications which have to span one hundred years or more, such as certain life insurance and pensions systems.

4.3 Managing the upgrade programme

In a typical organization, fewer than 20% of programs are affected by regular business maintenance in the course of a year. By contrast, some 90% of all programs are likely to require upgrading for date problems. The emphasis is on the management throughout and, with cost estimates in the **£millions**, it is going to take the remaining 3–4 years

left to do the work. Since so many programs need to be changed, regular maintenance will have to be scheduled alongside the work on the year 2000. All in all, the year 2000 upgrades will become a high-profile project.

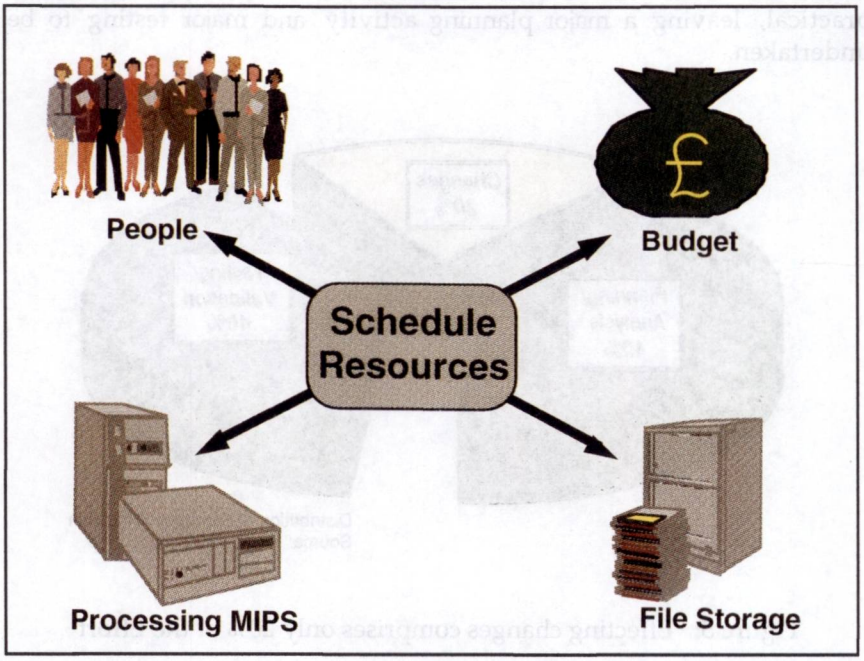


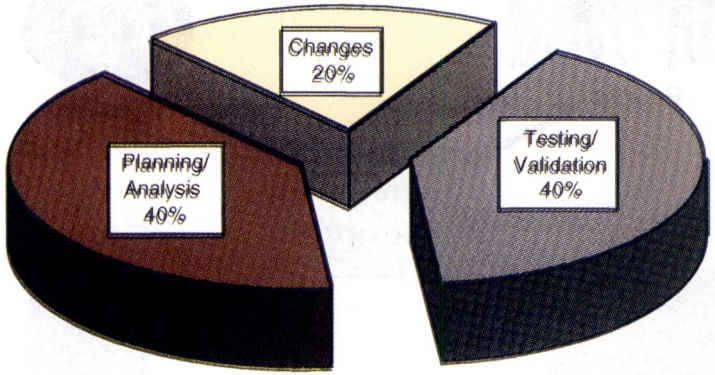
Figure 4: Resources should be scheduled with care.

A further challenge for management will be to co-ordinate all available resources in order to make the best use of in-house staff, systems, the additional data storage needed for conversion, as well as the outside contractors needed to make up for the inevitable shortfalls.

Experts agree that most effort will be spent on planning at the outset and on testing at the end. This leaves the important phase in between of upgrading the code and data. The way in which the programme is tackled will vary widely, but many organizations will choose to outsource as much of the work as is practical, in many cases to offshore specialists.

Grouping the changes so that interworking systems are dealt with simultaneously is essential, in order to minimize the problems. In particular, any changes to data formats need the most careful co-

ordinating. Either all the programs which use the data have to be changed, tested, and returned into production together, or "bridges" need to be built so that the data can be converted temporarily back and forth until all the programs have been changed to correspond. Sometimes the data will be too extensive for this approach to be practical, leaving a major planning activity and major testing to be undertaken.



Distribution of Resource Allocation
Source: Tactical Strategy Group

Figure 5: Effecting changes comprises only 20% of the Effort

4.4 Testing and Validation

There is no doubt that, of the four main steps, testing will be the major challenge. The problem is not merely one of scheduling, but that the system clock on a production mainframe cannot simply be reset to the year 2000 when some upgrades are ready to be tested. Apart from disrupting the rest of the workload, and corrupting catalogues and other system files, it may well cause a violation of some software licences and, once licences have expired, they cannot usually be retrieved.

As if there were not enough problems in getting the work done before the year 2000, all the changes will need to be completed in time to run through a complete annual cycle before the end of the century. So that organizations, whose financial year begins in April, would need to set a target of March 1998 for completion of all their upgrades and testing.

In order to prevent new date problems creeping in, after the work has been completed, it will be necessary to have, firstly, a process to certify

where upgrades for the year 2000 have been carried out and, secondly, some firm guidelines to prevent regression.

5. Where to go for help

An increasing number of vendors now offer services to assist with upgrades for the year 2000. Some offer tools of their own and others are armed with a selection of tools, from the many available in the market place, to fit in with their clients' particular needs. Most of the tools will help to analyse, and in many cases to automate, repetitive code changes in COBOL programs. This is hardly surprising, since an enormous amount of the problem code, in terms of sheer numbers of lines affected, is written in COBOL. Tools, which can look at other languages, are beginning to appear and more will be added as the vendors bring them to market.

6. Conclusions

Although the year 2000 is an issue that can still raise an indulgent smile when first discussed with the unwary, it is clearly a matter of major management concern, with direct implications for business survival.

IS has a duty to ensure that business managers understand the scope and scale of the problem and that the business provides IS with the resources to solve the problem.

References

GARTNER GROUP, "Year 2000 Crisis: Estimating the cost," KA-210-1262.

GARTNER GROUP, "Year 2000 Crisis: Building the Program Management Office," SPA-650-1351.

GARTNER GROUP, "Year 2000 Crisis: Why aren't CEOs Convinced?", SPA-650-1364.

Year 2000 – Additional Publications

External Publications:

- DRS User News, January, 1996
- CUA User Link, March, 1996
- ICL Strategy for Business, March, 1996

- "Dateline 2000," ITMP (industry subscription information service), April, 1996
- VAR magazine, June, 1996

Biography

Janet Pavelin is ICL's leading consultant on the issue of how the year 2000 date change problems will affect IT systems as the millenium approaches.

During her 25 years in the computer industry, she has held a number of technical and marketing roles, which have been undertaken in a variety of IT application areas including software development & support, CASE & application development, document management and office systems.

These activities have included providing application development solutions for information systems, where she was involved in consultancy to the sales force and customers, as well as liaising with third party suppliers. Prior to that she worked in office systems, primarily in the area of document management and was responsible for marketing ICL's products ICLFILE, POWERFILE, and software for Records Management.

Janet Pavelin has a degree in Mathematics from Birmingham University.

Working with Users to Generate Organisational Requirements: The ORDIT Methodology

Ken Eason, Susan Harker and Wendy Olphert

HUSAT Research Institute and
Department of Human Sciences
Loughborough, Leics. LE11 3TU, UK

Abstract

In an era of rapid change it is very difficult for user organisations to specify their requirements for future forms of information technology. This paper presents the ORDIT methodology (Organisational Requirements Definition of Information Technology Systems) which was devised in the European Union ESPRIT research and development programme to support stakeholders in user organisations in their efforts to define and agree future socio-technical systems for their business and therefore to define their information technology systems. The methodology includes an enterprise modelling approach which uses responsibility as a concept to integrate business functions, the roles of human agents and requirements for information systems. The paper includes a case study example of the application of the methodology and ends with a discussion of its potential value to supplier organisations.

1. Requirements Generation

It is a truism to say that information technology has the power to transform the way in which modern organisations conduct business, serve their customers, use their staff etc. In a rapidly changing world it is vital that user organisations learn how to harness the potential of the technology to keep them ahead of their competitors. And yet we frequently find that organisations are slow to appreciate the role the technology can play or engage in large scale and costly development plans which often founder. Evidence emerging over the last fifteen years indicates a poor success rate for major IT systems [Lyytinen and Hirschheim, 1987], [Kearney, 1990] and notable, costly failures continue to be regularly reported in the press, for example, the computer system introduced to support the London Ambulance Service [Page et al, 1993]. As a result many managers are nervous and uncertain about the

application of new technology and need support in the requirements definition phase of development to develop clear visions and the confidence that they can plan and implement effective systems. The purpose of this paper is threefold. First, to examine the reasons why organisations find it difficult to make radically new uses of information technology and secondly to describe the ORDIT methodology, an approach which helps organisations explore and define their future uses for information technology. Finally, we have found in practice that the methodology has a particular role to play for suppliers of information technology and we explore the issues of suppliers using the methodology to help customer organisations define their future needs.

We can identify two main reasons why the implementation of major IT systems is difficult. First, the systems have to meet the business needs of the organisation and second they have to be compatible with the structure and culture of the organisation [Walton, 1989]. Traditional systems analysis techniques have a number of weaknesses in relation to these issues. First, they focus on functional needs and do not pay direct attention to organisational or cultural needs. Second, they examine today's needs and the fast moving business climate often means the needs of tomorrow may be very different. System developers recognise these problems but when they ask user organisations to describe the important organisational and cultural needs or to define future business needs, they are usually met with bafflement. How can users specify the information needs of the future when they are still planning how to change their business? How can they be specific about such slippery subjects as the structure and culture of the organisation? Even more awkward - how might this new technology, which they only dimly comprehend, help them achieve future business goals? Requirements engineering is a notoriously difficult part of the design process and there is little systematic support in current methodologies for the definition and modelling of user requirements for information technology systems [Lubars et al, 1994]. It is our view that there needs to be a major shift in the perspective from which we address the requirements engineering problem. We need to move from talking about requirements capture (as though there were fixed needs waiting to be identified) and talk about *requirements generation* - helping the important stakeholders in user organisations work out the future they desire and then define the information systems requirements that flow from this. The ORDIT Methodology described in this paper is an example of an approach which seeks to meet this objective.

2. The ORDIT (Organisational Requirements Definition of Information Technology Systems) Approach

The ORDIT Methodology was developed by a five partner consortium in a five year project funded by the European Union ESPRIT Research and Development programme. ORDIT provides a technique to help stakeholders in user organisations to define alternative technical and organisational futures and, as a result, specify the future role for information technology. It aims to provide a systematic and usable process which supports organisational requirements generation, and to supply associated methods and tools to support the process. In order to achieve this aim, the methodology adopts the following principles.

2.1 A socio-technical systems approach

The design target of using the ORDIT Methodology is a socio-technical system which will serve organisational goals. A socio-technical systems approach seeks to achieve joint optimisation of human resources and technical systems [Weisbord, 1990]. If a technical system is treated separately from organisational issues, the result may be a system which functions well but does not serve its users. This may also mean the neglect of the organisational changes needed to allow staff to exploit the technology. Similarly, the independent development of organisational structures and policies to achieve organisational objectives would be counter-productive if it did not consider the possibilities and constraints of the technical system which will provide the tools to do the job [Eason, 1988]. The aim is systems integration, whereby the technical system is well aligned with organisational structures, processes and developments. The ORDIT Methodology has been designed to assist in achieving socio-technical systems integration, by identifying the requirements of the social system (organisational requirements) and by exploring the implications of possible technical systems.

2.2 A user-centred approach

Successful system developments are user-centred; the technology must be designed as a tool to serve the needs of users, not the other way round. This involves a high degree of user involvement in the systems development process, both to ensure that the relevant user knowledge about the organisational structures, processes etc. is captured and embodied in the IT system designs, and to ensure users understand and are committed to the systems which might then be implemented.

The ORDIT Methodology ensures that key users are identified and their concerns and requirements are explored. In order to support a user-centred design process, ORDIT attaches key importance to enhancing communication between problem owners and designers. In Section 4 below we outline the concept of an organisation as a network of work roles and responsibilities. This concept has been found to provide a useful communication medium between the parties involved in systems development. It helps to identify organisational requirements, and represent them in a form that both system designers and problem owners can understand and evaluate.

2.3 An iterative approach to the generation of requirements

Linked to ORDIT's user-centred philosophy is a system design process based on an iterative cycle of specification, prototyping, user testing and re-design. As users are unlikely to be fully aware of all the design possibilities and implications at the beginning of a design process, there is a need to be prepared to revisit earlier stages as their awareness increases and their perceptions change. In the traditional view of the software development life cycle requirements are captured before the design stage. This does not recognise the growing awareness of users as development progresses which means they are increasing able to generate requirements and be more realistic and confident about them. The ORDIT approach is explicitly evolutionary in recognition of the fact that this revision and growth occurs.

2.4 Providing choices and creating new opportunities

It is important that systems design does not confine itself to creating solutions to problems by fossilising existing practices which may have arisen historically for reasons which themselves no longer hold. New IT developments provide an opportunity to look at different ways of doing things. ORDIT therefore encourages scanning for new opportunities and consideration of a number of different socio-technical options to explore a range of possible futures. The aim is not to move immediately to a "solution" rather it seeks to expand the problem space and explore a wider territory from which a solution can be chosen.

2.5 Modularity

Experience in organisational consultancy indicates the need for a flexible methodology which takes account of the different needs and starting points of clients. For example, some organisations may have more IT experience than others or may already have well-established

user groups. Organisations vary enormously in the complexity of their structures and their links with the outside world. Organisations will also vary in available budgets and time scales for IT development. There may be different attitudes to participative approaches. The overall structure of the methodology is therefore based on a modular approach which allows choice from a variety of individual methods and techniques and which produces outputs which can be integrated with a wide range of technical development methods. "Route guidance" showing how and why the modules might be used in different contexts is therefore an important element of the ORDIT Process.

2.6 Enhancing Communication

A key objective of ORDIT is to enhance communications between 'problem owners' and 'problem solvers'. The ORDIT Methodology contains a set of modelling concepts which enable both problem owners and system designers to work with a common representation of the social and technical sub-systems that describe the organisation. The modelling concepts are supported by a number of processes through which problem owners and problem solvers can work towards a model which constitutes the agreed and desired future state of the organisation.

3. The ORDIT Process

ORDIT was developed by an iterative process with many tests against case study material of increasing complexity and realism. It has subsequently been used in a range of system developments one of which is described in Section 5. The methodology is summarised in this section (the full methodology is described in an ORDIT process manual, 1993). The primary constituents are a series of interrelated processes of engagement between problem owners and problem solvers and a modelling language by which socio-technical options can be represented, evaluated and debated. The activities in the process are described in Figure 1.

The four activities in the methodology are scoping, modelling, requirements definition and option generation/evaluation. The normal sequence begins with scoping but thereafter all activities interact to support the movement towards the definition of a agreed future system. The four activities are summarised in the following sections.

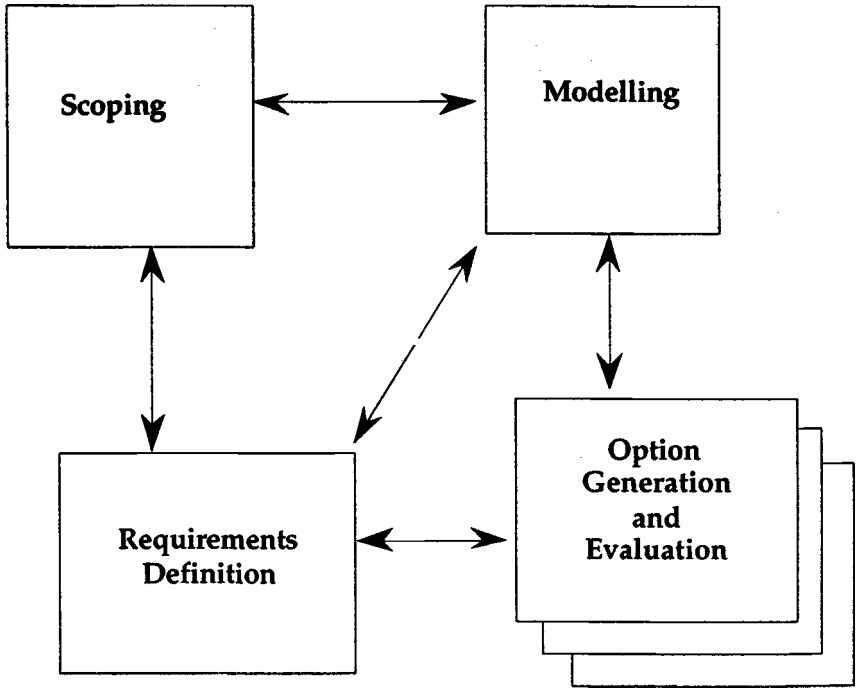


Figure 1: Activities in the ORDIT methodology

(Adapted from the ORDIT Process Manual, [ORDIT Consortium, 1993])

3.1 Scoping

The starting point for an ORDIT study is the determination of the scope of the future system in several respects; first the scale of the organisational unit which could be subject to change, second the principal problem owners or stakeholders whose requirements will influence the future system and thirdly the major drivers for change, i.e. the initial statements of requirements by the stakeholders. Another important component of this activity is the establishment of a contract between the problem owners and the ORDIT methodologists; for example, the access they will have to stakeholders, the nature of the dialogue, the expected output, the timescales etc.

3.2 Requirements Definition

One of the main purposes of the methodology is to help stakeholders make informed and explicit statements of their requirements and, where possible, agree requirements with other stakeholders. At the

beginning of the process it is likely that the requirements will be incomplete and vague and that there will be a lack of appreciation of the implications of the potential effects of the technology on organisational and cultural change. It is also unlikely that all the major stakeholders will have played a part in the debate about the potential change. The requirements definition activity involves those applying the ORDIT methodology in work with stakeholders to specify requirements which are both social and technical in character. The third and fourth activities are necessary to enable stakeholders to achieve these definitions. The aim is to use these two other activities (modelling and option evaluation) to help stakeholders generate more explicit and comprehensive sets of requirements.

3.3 Modelling

In order to help stakeholders appreciate the inter-linking of business, organisation and technical issues and therefore to refine their requirements, models are needed which can show these relationships. ORDIT includes a modelling language which has the concept of responsibility as its focus because all three aspects of the enterprise can be related through it. A business process is a set of responsibilities to be fulfilled both functionally (to meet a customer's order) and non-functionally (with politeness, opening the way for future orders etc.). It is the human agents who will be held responsible for achieving these objectives and they will have rights and obligations with respect to information if they are to fulfil these responsibilities. Within ORDIT the existing work organisation that is subject to change will be modelled as a set of interacting responsibilities and represented in a form that stakeholders can understand and verify. More details of the modelling language are given in Section 4.

3.4 Option Generation and Evaluation

An ORDIT model is produced for the existing work organisation which provides the basis for the generation of future socio-technical systems. The broad statements by stakeholders of their requirements for future systems are used to produce variants on the ORDIT model which demonstrate the direct implications for the business process, the organisation of human work roles and the information systems. Different stakeholders may, for example, be seeking a more virtual organisation, a more centralised structure, to 'out source' some non-core functions, to have a flatter structure with greater empowerment for members of staff etc. Each alternative has business, organisational and information technology implications. The aim of the ORDIT methodology at this stage is to model major alternatives, represent

them in a form stakeholders can understand and help them work through the implications from their perspective. This process involves the creation of future scenarios with which stakeholders can interact and apply a user cost-benefit assessment procedure to assess the implications from important perspectives. The effect is to help stakeholders see the wider effects, make more explicit their requirements, establish priorities and appreciate that there are many other variants of the scenarios. This process reveals the differences in stakeholder requirements but, because it also shows the alternative futures, it can also encourage a debate about mutually acceptable futures. Boehm et al [Boehm et al, 1993] believe it is possible to identify 'win-win' options in which the major needs of all stakeholders can be met. From our experience there are likely to be winners and losers but losses can be made explicit and minimised and early processes set up to manage the change.

The outcomes of this process are more explicit statements of the requirements of the stakeholders (business, organisation and technological) and a 'buy in' to a socio-technical system vision of the future from which can be specified future information technology needs.

4. Modelling Responsibilities

This section provides a more formal statement of the enterprise modelling approach used in ORDIT. Within the methodology the focus is on understanding and describing an organisation as a set of related work roles. A role is a descriptive concept that can be used to represent many different organisational realities from the formal and structured to the fluid and unstructured. Individuals within organisations usually hold multiple roles (e.g. one individual might be *manager* of a specific function, *leader* of a specific team of people, *member* of a particular committee, *deputy* to a more senior manager etc.)

A core concept in the ORDIT approach to understanding roles is the **agent** entity. We describe these as the primary manipulators of the state or structure of the system, but essentially they are the people in the socio-technical system. What an agent entity represents is an "office" in the sense of a role holder, and this can be any size of group from an individual to a whole organisation.

Following from the concept of the agent entity is the concept of the relationship between agent entities. We call these **structural relationships**, because we regard them as the skeleton of the socio-technical system. Every relationship between one agent entity and

another implies a conversation and the need for some sort of communication link between them permitting the exchange of information. Structural relationships between agents are thus central to ORDIT in two respects, in that they embody the organisational structure in terms of authorisation and power structures, and in that they impose **requirements** in terms of information and communication structures on any IT system installed.

Figure 2: A structural relationship between agent entities

The key to modelling structural relationships is the realisation that they are basically responsibility relationships between agent entities. This brings us to a second core concept in ORDIT, that of **responsibility**. ORDIT analysis and modelling is based on the fundamental premise that the function of an organisation is manifest in the responsibilities held by individual role holders, and that the structure of the organisation is manifest in the responsibility relationships between them. Therefore the identification of different role holders and their responsibilities leads to the identification of both functional and organisational requirements on the complete socio-technical system — and in turn the requirements on potential IT systems within that socio-technical context.

Responsibilities cannot be looked at on their own but must always be considered as a relationship between two agents. The states of affairs for which responsibilities are held may be at any level of granularity of the organisation. For example the responsibilities may be at a very high level such as for the adequacy of the service provided, for the continuity of a process, for safety, for security, for the accuracy of information and suchlike, or they may be at an individual level for a very specific state such as whether a door is closed, or whether a form is correctly filled in.

We can now provide an overview of the way in which ORDIT modelling can be used in the different stages of an ORDIT project. In order to do this it is necessary to include brief descriptions of the concepts and the modelling framework. We have already introduced three of the main ORDIT modelling concepts, i.e. agents, responsibilities and structural relationships. Other essential elements in modelling an organisation are actions and resources, where an **action**

entity is an operation that changes the state of the system, and a **resource entity** is what enables the agent to do the action. We say that the agent entity has a **functional relationship** to the action entity, since the agent does the action, and that it has an **access relationship** to the resource entity, while the action entity has a **requires relationship** to the resource entity; i.e. the agent must access the resources that are used by the action when it is performed by the agent. As already stated ORDIT is particularly interested in organisational structure so the **structural relationship** is shown. Relationships between resources (resource schema) and between actions (interactions) are of less interest to ORDIT, as these can be represented by data and process models respectively.

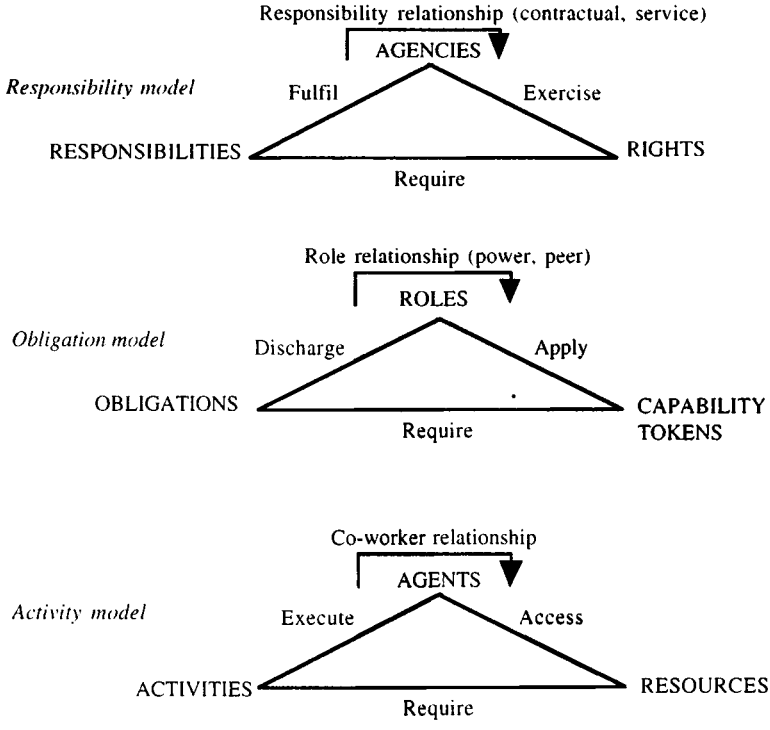


Figure 3: The specific entities and the relationships between them in the three levels of ORDIT models

In addition to being described as a set of responsibilities an organisation can also be described as a set of obligations and/or activities. These can be regarded as separate but inter-related models of the organisation

each including the same three basic entities: agents, activities and resources, and the relationships between them, as shown in Figure 3. The models are arranged vertically in the diagram to emphasize the fact that the vertical links join concepts of the same type, and that the intermediate level concepts are the links between the more abstract and more real concepts. As stated above, **obligations,** as we define them, are recognized to be the link between **responsibilities** and the execution of **activities** in the real world. Similarly **capabilities** to access resources are recognized as the link between the high level concept of **rights** to access resources and the low level concept of the **resources** themselves. These relationships form the infrastructure that binds together the separate models into a coherent framework.

The prime feature of the ORDIT approach to modelling is flexibility; the modeller is presented with a set of interrelated concepts and is free to employ them in whatever way will most effectively represent the problem in hand. Depending on the issues to be addressed and the kinds of changes to be considered it may be possible to model some parts of the system at a high level but to explore others in much greater detail. Working in a user-driven way means that the modeller creates models which are useful to the stakeholder deliberations rather than trying to achieve completeness for its own sake. It is also quite likely that the process moves quickly from a high level model of the current system to the modelling of possible future options before exploring any features regarded as problematic or interesting, in greater detail. The order in which different aspects are modelled is not prescribed but is driven by the emerging pattern of requirements.

5. A Case Study in the Use of ORDIT in a User Organisation

One user organisation in which this technique has been applied is the application of a mobile communications system in an electricity distribution company. The case study is more fully reported [Eason, 1996]. This case study involves a task common to many service organisations and is therefore of interest to the creators of generic products to support co-operative tasks. We will present it first as it was examined within an electricity company and, in Section 6, we will consider the implications for a generic product supplier.

In this study we first made a coarse grained model of the initial system, i.e. the current work roles, responsibility structure and structural relations and the principal activities necessary to complete what is a major co-operative task. What follows is a brief summary of the current

system. When customers telephone an electricity company, they may report loss of power or may request some other form of service, for example, repair to a central heating system. In the current organisation these requests were received by a clerk in a central customer services office. The clerk was responsible for designating the enquiry an emergency or non-emergency. If it was an emergency (no supply or a dangerous situation) the company had a statutory responsibility to respond as quickly as possible. The request was sent to a radio operator whose responsibility was to contact an electrician in his van in the vicinity who then had the responsibility to give the call priority and to deal with the emergency. If the clerk designated the call a non-emergency, the enquiry was sent to the foreman of the depot in the neighbourhood and the foreman's responsibility was to allocate this call and all other outstanding enquires, to his team of electricians who would take responsibility for making the necessary calls the following day.

This is a deceptively simple allocation and scheduling activity but, in the electricity company, there were significant problems in delivering an adequate service. The company was intent upon improving the speed and reliability of its service. The non-emergency route had a built-in 24 hour delay and it could be longer. The emergency route suffered from a problem with the radio system. Electricians spent most of their time on customers' premises and were not therefore in their vans to receive calls from the radio operator. As a result the radio operator often had to make many calls before being able to allocate the task to the electrician.

The next part of the analysis was to explore existing visions of the future. To overcome these difficulties the company planned to implement a mobile data transmission system. It had a vision of how this technical system would operate; it would equip the radio operator with a means of electronically encoding service requests and sending them directly to a data link installed in each of the vans. On returning to his van the electrician could then read the message and act upon it. Since electricians often had spare capacity, the aim was to transmit not only emergency jobs by this route, but also as many non-emergency jobs as possible in order to obtain a same-day response.

An ORDIT model of the new organisational scenario was developed which demonstrated the way in which the new technical system would affect the responsibilities of the staff concerned. It would provide customer service clerks, for example, with an opportunity to allocate

more of the non-emergency calls for completion that day by passing them to the radio operator for transmission to the van of an appropriate electrician. However, to do so they needed further information resources, i.e. access to the planned workload of each electrician and guidance on the mix of work that constituted a normal day's work for an electrician. This scenario was developed to show the responsibilities of each work role in relation to the new technical system and the resources each would need to fulfil their responsibilities. Once the scenario was developed it was presented to staff of the electricity company (mostly junior and middle managers rather than intended users) who considered it from the perspectives of the different users. They employed a *user cost-benefit assessment* method [Eason and Olphert, 1995]. This method helps people to assess the implications of change for a particular user at the task, job and organisational level. All of the staff were familiar with the normal way of dealing with service calls and the existing culture and practices in the electricity distribution industry and were therefore able to make judgements of the implications of the new system.

There are four major direct user groups in this organisational structure (enquiry clerks, radio operators, foremen and electricians) and a summary of the profile of probable outcomes for these staff is presented below. The profile of other stakeholders, for example, management, the customer, the depot storekeeper etc., have been omitted for the sake of brevity. The assessment shows clear winners and losers. With some caveats, the clerks and radio operators gain, but there are serious problems for the foreman and the electricians. The clerks are now able to promise customers a much faster response and allocate a proportion of non-emergency jobs on the day the enquiry is received. The radio operator can be much more efficient in transmitting messages and does not have to engage in debates with electricians. The electrician does not have to wait for contacts from the radio operator but experiences much loss of autonomy because he is now 'controlled' by the data link in his van. When the data link accepts a message it is, in effect, committing the electrician to undertaking the job. The foreman is responsible for all the electricians who work from his depot but, with the data link in operation, he does not know what work is being allocated to his staff.

In the view of the staff who acted as user representatives this system would be unacceptable within the culture of the organisation and ineffective in achieving the objectives of better customer service because it would not enable depot staff (foreman and electricians) to plan their

work effectively. It placed too much power in the hands of central office clerks who do not know local depot conditions or the nature of the

electrical work required and who may make inappropriate allocation decisions which would be difficult to change. As a result of this analysis the staff who made the scenario assessment now felt able to offer a wide range of proposals for a system which would be more acceptable and, by using the expert knowledge of foreman and electricians, be more effective for the company. These proposals were developed into four further ORDIT models. In summary, they were:-

1. The Clerks as Work Allocators. In this scenario the role of the clerks is further enhanced. The role of the radio operator is dispensed with and clerks make direct allocations to the electricians' vans. The allocation is regarded as provisional until the electrician acknowledges acceptance or refusal.

2. Allocation by the Foreman. In this case the clerk sends all calls for a specific area to the area foreman who uses the data link to allocate the calls, both routine and emergency, to his team of electricians. This reinforces the role of the foreman and enables him to use local knowledge to optimise the speed of response and the use of resources.

3. Allocation by the Electricians. Within the current organisation one of the semi-formal ways of coping was for electricians to swap jobs to suit local arrangements. One suggestion for a new system was that calls for a particular area should be announced electronically and electricians could decide between themselves how best to deal with them. This would be an example of a virtual autonomous work group.

4. Contracted Out Electricians. Within the spirit of the out-sourcing movement, another concept was that the electricians became approved sub-contractors of the electricity company. Jobs could be announced in the electronic system as they became available and bids accepted from competing electricians on criteria such as speed of response, price and quality.

Each of these scenarios represented a solution to a different set of requirements (or different priorities assigned to requirements) and they were debated by the major stakeholders within the company. They settled, in the short term, for a central allocation model but with more scope for the foreman and electricians to modify the allocation locally.

In the long term serious attention was being given to the possibilities of outsourcing.

The importance of ORDIT in this context is threefold. First, it enabled an early explication of the organisational implications of a planned system. The development and evaluation of the initial scenario for change made it possible, without purchasing or developing the system, to assess its advantages and disadvantages from the user perspective. Second, the stakeholders in the organisation became knowledgeable about the future opportunities. Thirdly, they were able to articulate their emergent requirements for the system and to develop concepts of other ways of implementing the technology to the advantages of their business.

6. ORDIT for the Supplier

The mobile communications case study was set in an user organisation and a prime role for ORDIT is to support the planning of technical change within a user organisation. However, the methodology also has important implications for the suppliers of systems to support tasks involving co-operation between individuals in organisations. One obvious application would be for suppliers to use the ORDIT methodology to work with potential user organisations to help them develop the requirements specification before a development contract is placed. However, we have also found that ORDIT offers suppliers of generic products a mechanism to help them specify their marketing and product strategy. We can explain this interest by reference again to the electricity company case study.

The service enquiry task is common to all service organisations and many are interested in electronic support for the logging, allocating and scheduling of enquires. It is easy to imagine that a supplier might attempt to develop a generic service enquiry management system for this market. It would be an example of a growing array of products to serve co-operative tasks in organisations. The specification of the functionality of such a system would be relative straightforward; there appears to be a universal need for mechanisms to log the details of the service enquiry and to progress the enquiry through allocation, providing the service, billing etc. But what assumptions should be made about the user roles in relation to the system? Should the designers assume that a clerk will receive the enquiry, log it *and* allocate it to a service provider? If they do, they will only be able to sell the system to user organisations who wish to adopt the centralised approach to handling service calls. It could not be used by organisations wanting to

adopt scenarios 2, 3 or 4 as described in Section 5. It would also be difficult for any user organisation that was contemplating moving to another structure in the future, i.e. working out how to move to a model of sub-contracted electricians.

The central issue is that it is the stuff of organisational life that companies continually review and revise their internal structures and processes even when the basic business process remains the same. What approach should suppliers take to this fluidity? Ideally they should be able to offer products which contain the core, stable functionality and support to help user organisations configure the system to match the internal work role structure the customer wishes to adopt. We have been exploring the use of ORDIT to determine the range of organisational alternatives a product might need to serve, to help suppliers specify what can be core functionality, what needs to be locally configured and the processes and tools necessary to provide customers with this kind of support.

7. Conclusions - the problem and the promise.

The problem for user organisations is that new technology has enormous potential, they cannot afford to let their competitors get the advantage but they have difficulty knowing how best to employ it for their own business benefit. They are increasingly nervous of being sold instant 'off-the-shelf' solutions that turn out not to meet their requirements. But they have difficulty knowing in any detail what their requirements are for an unfamiliar technology. We regard the central problem as one of how to help people in this situation to generate their requirements so that they can take charge of commissioning and implementing new systems with clarity and confidence. We believe that the processes and tools within the ORDIT methodology provide a route for user organisations to achieve these ends and that the methodology also provides opportunities for suppliers to work more closely with clients and to understand the implications of organisational variation for the creation of their own productions.

Acknowledgements

The ORDIT project was funded in part by the European Union ESPRIT programme (project no. 2301) and the authors wish to acknowledge the support of the European Commission, the advice provided by project reviewers and the contribution of our ORDIT partners, MARI and the University of Newcastle in the UK, the Work Research Centre in Dublin and Algotech in Rome.

References

BOEHM, B., BOSE, P., HOROWITZ, E. AND LEE, M.J., "Software requirements as negotiated win conditions," Proceedings of the International Conference on Requirements Engineering, IEEE Computer Society Press, Los Alamitos, CA, 74-83, 1994.

EASON, K. D, "Information Technology and Organisational Change," Taylor & Francis, London, 1988.

EASON, K. D, "Division of labour and the design of systems for computer support for cooperative work," Journal of Information Technology, 11, 39-50, 1996.

EASON, K. D and OLPHERT, C.W., "Early Evaluation of the Organisational Implications of CSCW Systems," CSCW Requirements and Evaluation, (Thomas, P., editor) Springer-Verlag, London, 75-89, 1996.

KEARNEY, A.T., "Barriers-2 : Barriers to the Successful Application of Information Technology," Department of Trade and Industry, London, 1990.

LUBARS, M., POTTS, C. & RICHTER, C., "A Review of the State of the Practice in Requirements Modelling," Proceedings of the IEEE International Symposium on Requirements Engineering (Fickas, S. & Finkelstein, A., Editors), Jan 4-6, San Diego, California, 1994.

LYYTINEN K. AND HIRSCHHEIM R., "Information Systems Failures: A Survey and Classification of the Empirical Literature," Oxford Surveys in Information Technology, 4, 257-309, 1987.

ORDIT Consortium: "ORDIT Process Manual," HUSAT Research Institute, Loughborough University, Loughborough, UK, 1993.

PAGE D., WILLIAMS P. AND BOYD D., "Report of the Inquiry into the London Ambulance Service," Report Commissioned by the South West Thames Regional Health Authority, 40 Eastbourne Terrace, London, W2 3QR, UK, 1993.

WALTON, R.E., "Up and Running: Integrating Information Technology and the Organisation," Harvard Business School, Boston, Mass., 1989.

WEISBORD, M.R., "Productive Workplaces; Organising and Managing for Dignity, Meaning and Community," Jossey-Bass, Oxford, 1990.

Biographies

Ken Eason

Ken Eason is Professor of Cognitive Ergonomics in the Department of Human Sciences at Loughborough University and an associate scientist in the HUSAT Research Institute. Ken Eason has worked for 25 years on the impact of computer systems on organisations and the development of design methods to take account of organisational issues. He was a co-applicant for the ESPRIT ORDIT project which created the ORDIT methodology. Amongst his publications are the books 'Information Technology and Organisational Change' and 'Managing Computer Impact.' He has engaged in a number of collaborative research projects with ICL and worked with Huddersfield University on the development of user-centred methods for ICL.

Susan Harker

Susan Harker is a Senior Lecturer in Ergonomics in the Department of Human Sciences at Loughborough University and an associate scientist in the HUSAT Research Institute. Susan Harker has made significant contributions to the development of user-centred design methods and is the convenor for the software parts of the international standard ISO 9241 on ergonomics of visual display units and for the BSI Working Party developing standards for user-centred design. Susan is co-author of the book 'User-Centred Design for Information Technology' which was based on the ESPRIT project HUFIT, of which ICL was a consortium partner. She was consortium project manager for ESPRIT project ORDIT and recently principal investigator in EPSRC project PROTEUS on the effects of changing requirements in systems development.

Wendy Olphert

Wendy Olphert is a Senior Scientist in the HUSAT Research Institute and a Senior Research Fellow of Loughborough University. Wendy has worked with many user and supplier organisations on the human factors issues in systems development. She was HUSAT project leader for ESPRIT project ORDIT and a leading team member in EPSRC project PROTEUS.

Network computing with remote Windows

J.B. Brenner

ICL, Lovelace Road, Bracknell

Abstract

This paper discusses the influence, that a small technology change (remote windowing), will have on a very large market (network computing, personal computing systems and the Internet). The author believes that this new technology will have a profound effect on the future course of evolution in IT systems and the IT industry.

The prospects for Network Computers, as they are currently conceived, are analysed and the conclusion emerges that they will have relatively narrow applicability. The main growth will be in powerful server systems and client computers will be "ultra-thin" personal computer terminals.

1. Introduction

Sometimes small changes in technology have big impact on very large markets.

This paper is about a small technology change (PC remote windowing), which will have big impact on a very large market (network computing, PC systems and the Internet). This new technology will change the future course of evolution in IT systems and the IT industry.

These events will unfold during 1997. The paper explains the technology, and will help interpret what happens to PC systems, Microsoft, Intel and the rest of the IT industry during the next year or so.

The starting point is network computing, based on the Internet, the Web and Java software. This is the main future direction for all business information systems. We explain what network computing is (section 2), and why it is so important (section 3).

Network computers (NCs) will be a key ingredient of network computing. We explain what they are and what they do (section 4). It is widely assumed that network computing requires network computers.

The natural relationship between the two terms "network computing" and "network computer" suggests that network computing is something that is done by network computers. But this assumption is wrong, and there are other more cost-effective ways of achieving network computing.

The strongest alternative is PC remote windowing technology (section 5). This splits PCs into two physically separate parts: virtual PCs in PC servers (section 6), which are used via remote windows at PC terminals (section 7). This enables PC-based network computing (section 8), and gives PC systems freedom to evolve in new ways (section 9) which are very different from previous expectations.

This leads to the conclusion (section 10) that the natural way forward for the PC installed base and many other systems is easy incremental evolution to PC-based network computing, not radical change to NC-based network computing. This conclusion is illustrated in the diagram in Figure 1, which also indicates the reasons.

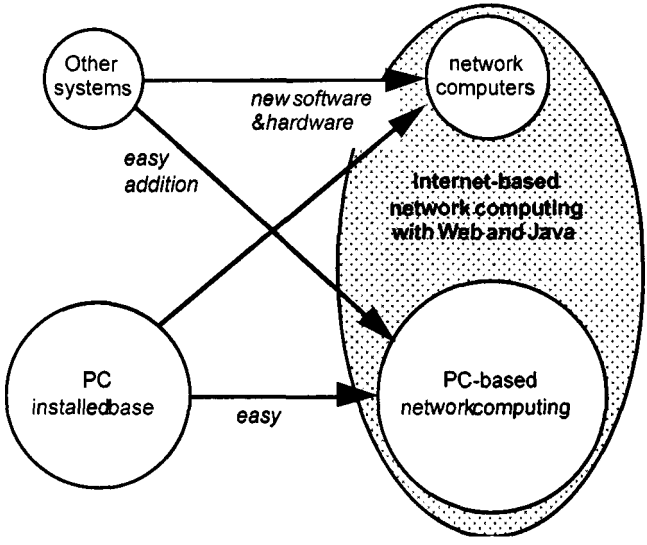


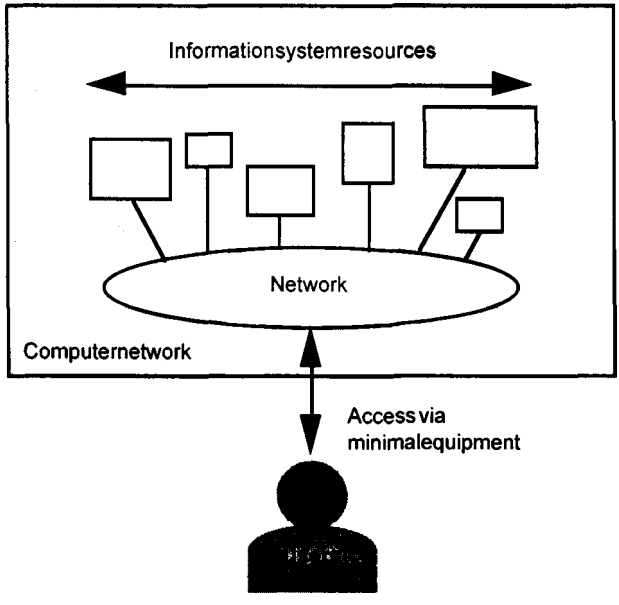
Figure 1 Evolution to network computing

It is, perhaps, self-evident that the PC installed base will evolve to PC-based network computing, instead of changing to completely different hardware and software (network computers). But this outcome is the exact opposite of current predictions that NCs will displace PCs from much of the business computing market.

As will be explained, these changes are all highly beneficial to PC users, but have some painful implications for the PC industry.

2. What is network computing?

Network computing is what happens when information resources are dispersed and connected together in computer networks, rather than being in one central place (mainframe computing), or on the desktop (traditional PC computing), or in a few places (traditional client-server computing). With network computing, everything is interconnected, and users with minimal equipment can have access to virtually limitless resources. This is illustrated in Figure 2. The Internet [Jeoffroy, 1996] is the definitive form of network computing.



Highly effective network computing has recently become universally possible by means of three standard technologies that are fundamental to the Internet:

- TCP/IP
- Web
- Java

TCP/IP [Jeoffroy, 1996] is a unified networking technology that provides flexible and powerful means of building computer networks. It includes Internet communications, Internet naming and addressing, Internet network management, Internet file transfer, Internet electronic mail and many other facilities.

Web technology [Jeoffroy, 1996] provides a distributed hypertext environment which is accessible over TCP/IP networking. Hypertext webs are a unified and comprehensive way of publishing information in computer networks, so that it can be universally accessible. Coupled with the introduction of graphical browsers, hypertext webs are largely responsible for the popularity of the Internet as it is today.

Java [JavaOne Conference, 1996] is a very recent addition to the Internet, but is already established as the Internet programming technology. It enables construction of highly modular software objects that can be stored in Webs, downloaded across TCP/IP networks, and executed independently of operating system or hardware differences. These modular software objects are called applets. Java also enables major improvements in the reliability, security, performance and flexibility of distributed processing software. Most new software for network computing will probably be Java software [Dolberg, 1996].

3. Why network computing?

The main reasons why network computing is advantageous are:

- Greater scale and reach
- Improved flexibility
- Simplified clients
- Centralised servers
- Simplified administration
- Reduced cost of ownership

The scale and reach of network computing is almost limitless. Use of Internet standards to implement network computing within a business enterprise enables computers of all kinds, throughout the enterprise, to co-operate with one another, and it enables them to reach systems in other enterprises anywhere on the Internet.

Flexibility is inherent in the Internet style of voluntary co-operation between autonomous systems, and is supported by the technical power and flexibility of TCP/IP, Web and Java technologies.

Simplified client computers become possible when the main resources are in the network. The simplified client platforms ("thin clients" and "ultra-thin clients") are cheaper than full-function PCs and workstations ("fat clients"). This reduces the equipment cost per user.

Centralised servers can be shared by many users, and concentration of workloads onto large multi-processor servers ("fat servers") now usually produces better price-performance than "thin servers" or "fat clients," or client-server systems constructed with these. This trend is expected to continue, and will further reduce the cost per user. The driving force is microprocessor hardware technology. For example, Intel microprocessor technology has recently moved upwards from client platforms and small uni-processor servers to medium sized symmetric multi-processing (SMP) servers, and will soon become prevalent in massively parallel processor (MPP) servers which will provide gigantic capacity.

Simplified administration is the biggest saving because administration is a large cost that is repeated year after year. Complex PC software and hardware that are dispersed on desktops throughout an enterprise are very expensive to manage, and this work costs much more than the equipment itself. Software and hardware that are concentrated in centralised servers are easier and cheaper to manage. This has always been an advantage of mainframes; but network computing achieves these advantages while delivering price-performance and flexibility far beyond what traditional mainframe configurations can achieve.

The total cost of ownership is reduced by the simplified clients, centralised servers and simplified administration. The appropriate measure is cost per seat per year, including client and server hardware and software, data-communications and administration. The cost reductions can be very large; for example [Bloor 96] quotes total cost reductions per seat per year from $14,500 to $6,900, within which the reductions for the client elements are from $13,400 to $2,900.

The market for network computing is much more than traditional business information systems. For example, it includes multimedia and telecommunications applications, and new applications in hand-held devices and in domestic appliances such as TV set-top boxes. Network

computing also embraces existing systems; for example, mainframes and "fat client" desktop and client-server systems can co-exist with network computing, and can participate fully in it.

Another factor is that the change to network computing will make markets more open. Major players in the IT industry, and in particular Sun and ORACLE, expect that it will break Microsoft dominance in high-volume software markets.

4. Network computers

Network computers are a key ingredient in the market battles led by Sun, ORACLE and Netscape against Microsoft dominance and the high cost of PC systems. Therefore, network computers are the subject of much hype, conflicting claims and counter claims.

Technically, network computers are a simplified kind of client platform, which is illustrated in Figure 3.

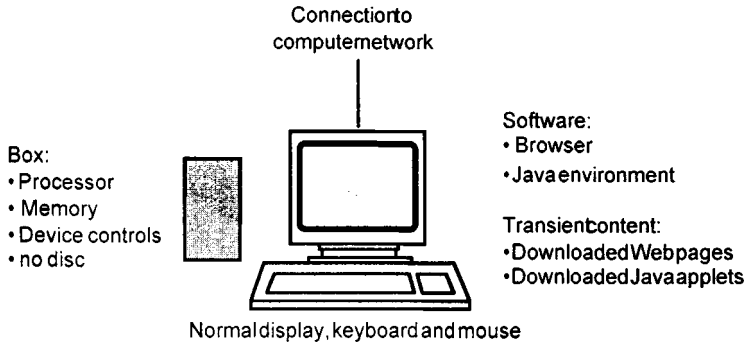


Figure 3 A network computer

These simplifications are based on the observation that with TCP/IP, Web and Java, it is possible for all participation in network computing by client platforms to be via a browser and a Java environment, working in concert. In a network computer the browser provides the user interface and all access to information, and the application software consists of Java applets which are downloaded to the Java environment whenever needed. Because all data and Java software applets are stored centrally in servers in the computer network, system administration becomes much easier, which results in big cost savings.

Network computers are marketed as "thin clients" that are much more cost-effective than "fat clients" (full-function PCs, with complex hardware, a complex operating system and comprehensive application software). This is true, but it misses the point that an equivalent PC with software reduced to a browser and Java environment is essentially the same as in Figure 3, except for about $100 of disc hardware to store the PC operating system. Furthermore, most users already have such a PC on their desktop (it just needs installation of a Java-enabled browser). Therefore, although a network computer is certainly a much more elegant design than an equivalent cut-down PC, the hardware is not actually much cheaper.

Network computers (and equivalent cut-down PCs) have the big advantage of reduced software administration costs when compared with "fat clients". But they also have the disadvantage that continual downloading of Web pages and Java applets depends on LAN network connectivity. Therefore, these configurations are not ideal for domestic or mobile use (two of the largest market areas) where this depends on dial-up modem connections. A more general difficulty is that there is, as yet, very little Java application software, and most of the relevant applications already exist as PC software packages which have dominant market share.

Although, as indicated above, network computers (NCs) are not universally the best buy, they certainly are an important new direction in network computing.

5. PC remote windowing technology

PC remote windowing technology is a another way for users to participate in network computing. It is highly cost-effective, and will be used from the desktop, hand-held devices and perhaps domestic appliances such as TV set-top boxes. We start by explaining the basic principles of this technology, and then go on to describe how it fits into network computing (see section 8).

5.1 Split PC

Technically, PC remote windowing is a way of dividing a PC into two separate parts which can be remote from one another. This is illustrated in Figure 4.

This split is based on the observation that inside PC systems there are powerful standard interfaces (e.g. the GDI graphics interface) at which

occur all the interactions between the user interface hardware and the rest of the PC (the processor, memory, disc hardware, and all the software and data). These interactions are highly economical, are not unduly time-critical (they are buffered), and are not directly visible to application software. Therefore, if a PC system is spit at this point, all application software can continue to work without change, and the whole system can run at full speed. To the user at the remote window it is as if the processing and storage were there, and not elsewhere.

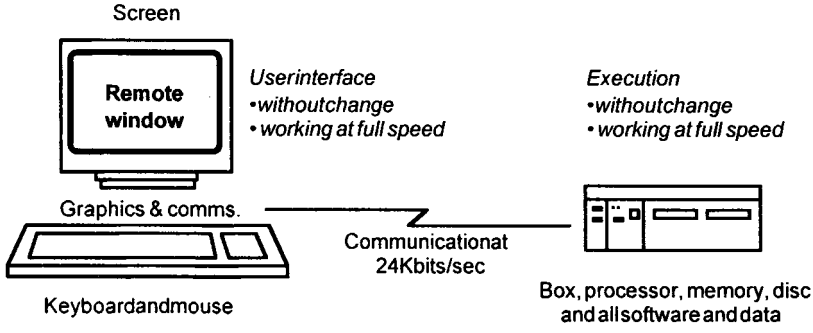


Figure 4 PC hardware split for remote windowing

The technology that does this became available during 1996. It is called ICA, and is developed by a company in Florida, called CITRIX, which is partly owned by Microsoft and Intel. The ICA protocol [CITRIX 96] is highly optimised. It uses TCP/IP to establish a session from the user equipment to the processing equipment, and then drops down to link-level frames for the rest of the interaction; it uses data compression and is encrypted. With this technique, 24Kbits/sec of bandwidth is sufficient for the whole interaction. This means that ICA can support remote windowing via normal dial-up telephone connections, and not just on high-speed LANs.

5.2 Multi-access server

The principles illustrated in Figure 4 are actually implemented in the way that is illustrated schematically in Figure 5. The terminology "PC terminal," "Virtual PC" and "PC server" is part of the OPEN*framework* Internet architecture [Brenner, 1996].

The PC software is executed in virtual PCs in PC servers (see section 6). The remote windows are at PC terminals (see section 7). A window at a PC terminal displays the whole desktop provided by the virtual PC accessed by the window. A PC terminal may connect to any virtual PCs

for which the user has access permission, and these virtual PCs may be in any PC server that is accessible from the PC terminal. A PC terminal may access several virtual PCs concurrently; in which case, each virtual PC is presented as a separate window.

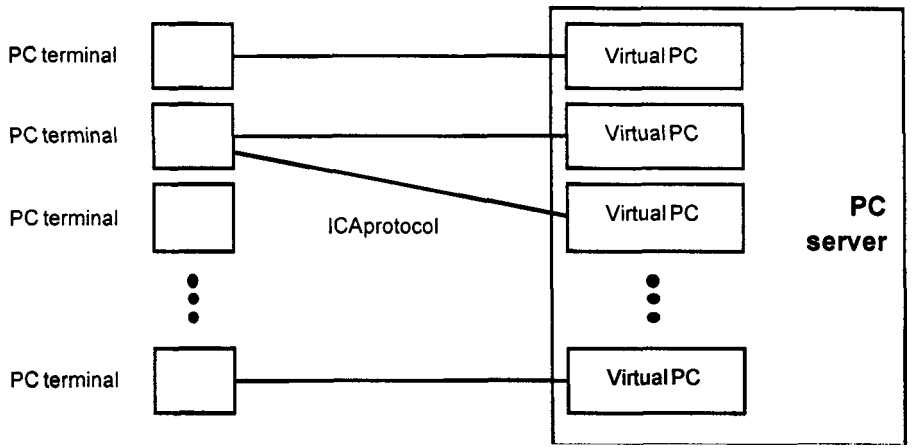


Figure 5 Remote windowing systems

By this means, the user equipment is reduced to "ultra-thin clients" (PC terminals), and all PC software and data can be centralised in PC servers, with consequent productivity improvements and cost savings. Furthermore, all the client and server functionality provided in a typical PC-LAN can now be provided in a PC server; in effect, the PC server is a "PC-LAN in a box". This has profound implications for PC networking, and for firms such as Novell who specialise in it.

5.3 X/Windows

Remote windowing is already a well-established technique in the UNIX world, where the corresponding technology is X/Windows. Referring to Figure 5, the corresponding X/Windows system structure is that the terminals are X/terminals, and they connect to UNIX processes in UNIX servers.

The main differences are that the field of application of PC remote windowing is immensely bigger (the whole PC installed base), the economics are much different from when X/Windows was introduced, and X/Windows does not perform well over dial-up modem connections. The market dynamics are quite different: PC remote windowing is a disruptive change hitting a huge mature market (PCs) at the exact time

and position where this is in collision with an even bigger emerging market (network computing).

PC remote windowing includes protocol gateways via which windows on X/terminals can use virtual Pcs.

6. PC servers

A PC server has a multi-access operating system that runs multiple virtual PCs at the same time. Each virtual PC is a single-user system, exactly like a desktop PC is; it runs the same application software that would run in the desktop PC.

This multi-access operating system is provided by Citrix **MultiWin** technology which extends the Microsoft Windows NT operating system into a true multi-user system (like UNIX already is, but executing PC software). Reciprocal agreements between Microsoft and Citrix enable this variant of NT to be produced and supported. Further extensions are provided by software from Insignia, which is a UK-based company with headquarters in the USA. The Insignia software [INSIGNIA 1996] is mainly concerned with systems management and with additional facilities needed to enable virtual PCs to be accessed from remote windows on UNIX workstations, X/terminals and Macintosh systems.

The underlying server hardware will normally be Intel-based, because this now provides the best price-performance, and is native to PCs. The server hardware can support concurrent execution (not just fast time-shared execution); this maximises the number of concurrent users. There can be multiple processors, each with internal concurrency (as in Pentium Pro), large processor caches, very large amounts of fast RAM memory, and multi-spindle disk systems with high-speed concurrent access. This also results in much faster execution than is usually achieved in desktop Pcs. Indeed, the only perceptible difference between remote windowing and local PC processing is that the remote windowing is usually much faster because the PC servers are so powerful.

Small PC servers can support small workgroups, and a large PC server (for example, a four processor Pentium Pro server) can support over 100 concurrent virtual PCs. There is no absolute capacity limit, because workloads consisting of virtual PCs are highly modular and incremental, and can be parcelled out to as many separate PC servers as are needed. In principle, the PC servers can be anywhere in computer

networks; but in practice they will typically be located somewhere near their users.

The PC software executed in a virtual PC can be DOS applications, 16 bit Windows applications, 32 bit applications, and the usual operating system facilities associated with them. NT allows all these different kinds of PC software to be executed in the same platform. The PC programs may be standalone PC applications, or they may use all the normal PC networking facilities. They may be the client ends of client-server applications, or terminal emulators that access legacy systems. The PC programs may also be Internet client software such as Internet browsers, Internet mail clients, Telnet clients, FTP clients, and Java applets executing on Java virtual machines. The implications of this for network computing are considered later (see section 8).

Although there are no inherent limitations to what can be executed in PC servers, the exact details of what-works-with-what and how well it performs are product-specific.

7. PC terminals

A PC terminal is either a specialised hardware unit (for example, a Fujitsu-ICL *ErgoWin* terminal), or terminal emulator software in a PC or other personal computer.

The current list of personal systems types that can behave as PC terminals includes:

- DOS and Windows PCs
- OS/2
- Macintosh
- UNIX workstations
- X terminals (via ICA gateways)
- JavaStation software environment
- JavaStation Network Computer (during 1997)
- Wyse Network Computer (also badged as Fujitsu-ICL *ErgoWin*)
- IBM Network Station
- Various other Network Computers currently being developed

- Personal digital assistants (PDAs) and other hand-held devices.

This enables all these diverse personal systems to have graphical windows into virtual PCs on PC servers. The terminal software displays the graphical images and transfers them and the relevant mouse and key movements between it and PC servers, via the ICA protocol.

Almost all PCs are powerful enough to operate successfully as PC terminals: the minimum specification is DOS 5 with a 286 processor, 4MB memory, VGA, 40 MB disc, and LAN or modem connectivity. As a PC terminal, this configuration can deliver to its user execution of PC software at NT server speed, without ever needing upgrades to the desktop hardware and software. Therefore, existing PCs easily evolve to handle all or part of their workload via PC terminal emulation.

8. PC-based network computing

PC remote windowing technology enables network computing in which software that would previously have been in client PCs is moved to PC servers, where it is used via remote windowing from PC terminals.

8.1 Ultra-thin clients

PC-based network computing enables an "ultra-thin client" approach to computer networking; it can move everything out of the client platforms and into server platforms in the computer network. This is network computing to an extreme degree. The conventional "thin client" approach is network computing to a much lesser degree. The contrast between these two degrees of network computing is illustrated in Figure 6.

This difference of degree indicates that key benefits attributed to network computing with network computers (namely, simplified clients, centralised servers and simplified administration) can be achieved to far greater degree with this alternative approach. And migration to this PC-based network computing is so easy.

Another way of viewing this distinction is that PC "fat clients" and network computer "thin clients" are both client-centric system designs, whereas PC remote windowing with "ultra-thin clients" is a server-centric system design. These are all valid ways of doing network

computing; market competition will decide how big a market share each achieves.

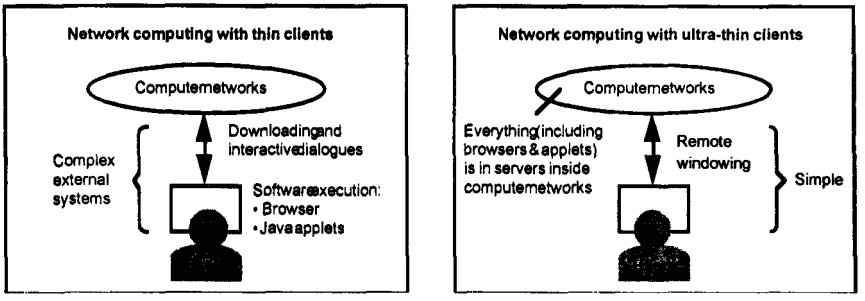


Figure 6 Two degrees of network computing

8.2 Combination of PC and network computing

In practice, most personal systems that operate as PC terminals (via terminal emulation) will continue to be used for other kinds of processing; a PC terminal window may be just one among several windows that are open at the same time.

This co-existence allows great flexibility and gradual transition to network computing, with selective trade-offs between local and centralised provisions. Also Microsoft promise software improvements leading to "zero administration," which will reduce the administrative cost differences between central and desktop systems. Combination of local and centralised computing is likely to be the most widely prevalent kind of PC configuration; it is illustrated in Figure 7.

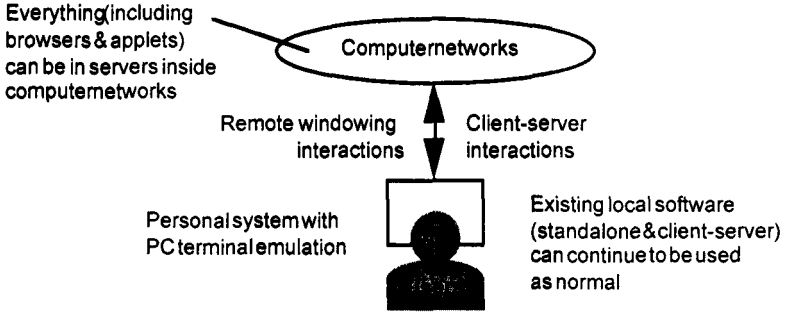


Figure 7 Combination of local and centralised computing

This approach is also applicable to non-PC desktop systems (e.g. UNIX workstations and Macintosh). They can continue to function as previously, but simply by adding PC terminal emulation software they can have windows onto virtual PCs anywhere in the computer network. For enterprises with diverse desktop systems, this is the easy way to evolve to unified PC desktop systems.

This approach is also ideal for mobile computing (using notebook computers or hand-held devices such as "personal communicators"). The terminal emulator is a small piece of software that can be implemented in any environment; it provides access to virtually unlimited functionality (everything that is done by PC software), and it performs well over dial-up modem connections.

8.3 System qualities

When PC software and data are in PC servers there can be much stronger guarantees of availability, security and performance than are generally possible with personal systems which are on desktops or carried about by people.

There are standard ways of configuring PC-based network computing so that virtual PCs can be automatically replicated onto separate PC servers, with strong guarantees of data integrity and continuity of service when PC servers fail, or are unavailable, or become unreachable [Brenner 1996].

PC server hardware enables higher performance, and can be used to provide professionally managed IT services with stringent service level agreements (SLAs). Most aspects of security can be better assured in professionally managed servers than in much larger numbers of client platforms managed by their users.

8.4 Universal clients and servers

With this technology, the user interface of every virtual PC automatically becomes a network-accessible service which can be accessed from any PC terminal anywhere (subject to access permissions). Virtual PCs are a universal kind of single-user server, and PC terminals are their universal clients. This is illustrated in Figure 8.

There are no software changes; it is just a matter of configuring the PC server and loading PC application software into it. In some ways this is unremarkable; for example, most UNIX programs are already network accessible. What is remarkable is the complete uniformity (one

terminal type fits all), the universal applicability (this applies to every PC user interface), the simplicity (no software changes), and the fact that this unexpected innovation applies retrospectively and without prior planning.

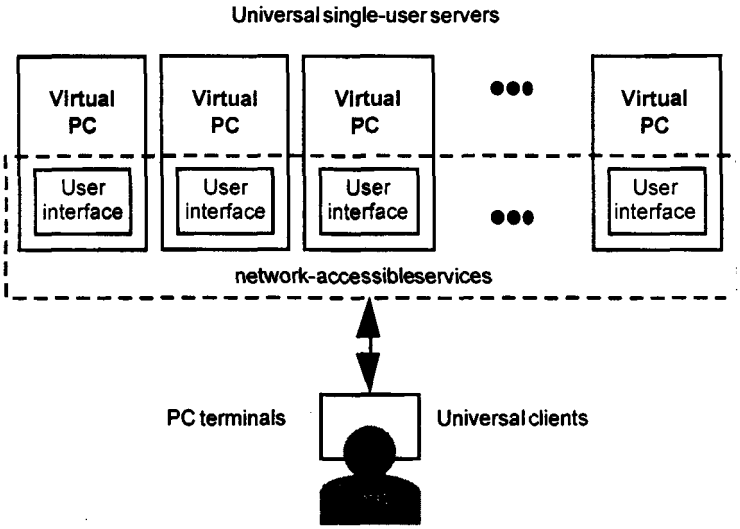


Figure 8 Universal servers and universal clients

8.5 Client-server application systems

PC client-server applications benefit greatly from this retrospective change.

The server parts are not directly affected; they remain where they were. But the client parts execute in virtual PCs in PC servers; their user interfaces are physically on remote PC terminals (although logically in the virtual PCs). This structure is illustrated in Figure 9.

With this approach, client-server applications are entirely executed in server platforms; all distributed interactions are then server-to-server or within-server. This simplifies the implementation, testing, roll-out and administration of PC client-server systems, and enables them to achieve better qualities, and in particular, better scaleability. This is mainly because the client parts of the applications are centralised in professionally managed servers, instead of being dispersed to desktops throughout the enterprise.

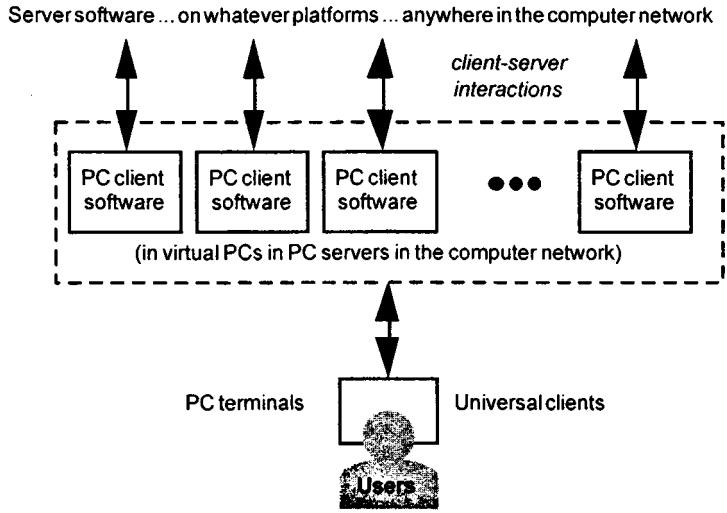


Figure 9 PC client-server applications

8.6 Browsers and Java

A distinctive characteristic of this PC-based network computing is that network browsers and Java software are in PC servers, instead of being executed in desktop Pcs.

In effect, this relocation creates "virtual network computers" inside virtual PCs inside PC servers. This is illustrated in figure 10. All network communication that would previously have terminated at the client platforms now terminates at the PC servers. All execution and caching of Java applets and Web pages is now in the PC servers, where there can be abundant resources. This enables every network-connected PC to participate fully in network computing without expensive upgrades.

This systematic repositioning of browsers and Java environments creates the opportunity to cage all downloaded files and all Java execution securely in the PC servers. Network viruses and other such security attacks do not propagate via the ICA protocol and its remote windowing; the PC terminals are outside their reach. Therefore, this innovative approach is inherently more secure than use of network computers and traditional "fat client" PC configurations.

Other Internet facilities such as Internet mail are now supplied as part of browser software products. When this mail client software is in

virtual PCs in PC servers, users can access their mail from PC terminals anywhere, and data integrity of their mail files stored in PC servers is strongly assured (see 8.7). This enables major simplifications and improvements in the operation and administration of electronic mail services.

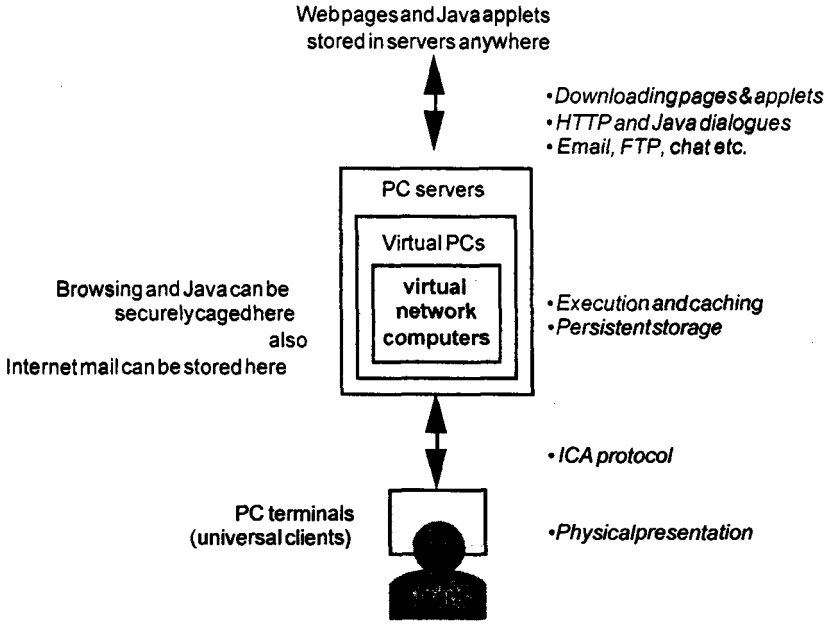


Figure 10 New approach to mainstream network computing

8.7 Interactive Web applications

PC remote windowing provides a simple way of constructing interactive Web pages by including PC application software in them. This is a very effective way of including business logic and other functionality in Webs. It is done by linking Web pages to PC remote windows. Web authoring tools can do this by a simple point and click operations that insert a special type of hyperlink.

Figure 11 illustrates what happens when such Web pages are accessed. The user is using a browser (the diagram is neutral to whether the user has a PC or NC with the browser in it, or a PC terminal using a browser in a virtual PC, as illustrated previously in Figure 10). The user accesses (1) a Web page via the browser, and selects (2) a hyperlink which happens to be linked to a PC application. The browser then uses its remote windowing (3) to access the user interface of the PC

application. The PC application may be of any kind, and might itself be a client application that has client-server interactions (4) with server applications. When the user exits from using the remote window, interaction with the original Web page may resume (5). The interaction (3) with the PC application can be of any kind, ranging from a simple request and response through to a prolonged session with many GUI interactions (this all depends on the design of the PC application). Important engineering details are not shown in this diagram, and will vary in different implementations, but the sequence (1) to (5) is the overall logical flow perceived by the user.

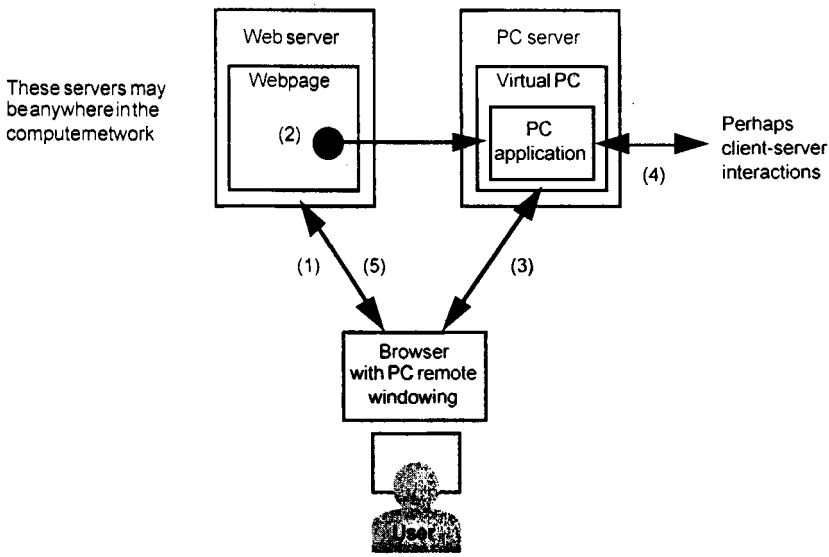


Figure 11 Interactive Web applications

This way of constructing interactive Web applications can exploit the full power and flexibility of PC software technology, and is much simpler than using the CGI interface and PERL scripting language, which is currently the main technique. As already explained above, the PC application may support extended interactions, in which state is created and maintained in the virtual PC (extended stateful interactions are much more difficult to implement via CGI). Therefore, this is a major advance in Web techniques. An advantage over using Java is that this technique simply reuses existing PC application software.

9. How will PC systems evolve?

Inevitably, most PC systems will evolve in directions set by Microsoft. In late 1995 Microsoft refocused on a decisive move into the network computing using Internet standards. The Windows user interface is changing to a browser interface, and all Microsoft office applications are becoming "Internet-enabled". The key product is the Microsoft browser, Internet Explorer, which will become the front-end for all the other products. Microsoft server products, packaged as Microsoft BackOffice are another major part of this strategy (Windows NT Server, SQL server, Systems Management Server, etc.). Windows NT and Internet Explorer are gaining market share relative to UNIX and Netscape Navigator, respectively. The whole range of new and upgraded products will be launched during 1997. Network computers are not on this Microsoft agenda (but Java is, although somewhat grudgingly).

Partly as a consequence of what Microsoft are doing, the PC installed base is already moving rapidly and successfully into Internet-based network computing.

PC remote windowing technology is the Joker in the pack, because it has emerged suddenly and somewhat unexpectedly. Microsoft have now signed deals that include it in Internet Explorer and BackOffice in various ways. But this is all very low-key. The implication is that Microsoft is not in a hurry to make this revolution happen. A present, Microsoft is totally focused on its 1997 product launches, and there is plenty more on their agenda that is more immediately profitable for them than PC remote windowing.

If the technical analysis in this paper is correct, PC remote windowing technology is going to cause revolutionary change in 1997, whether Microsoft wants it or not.

10. Conclusions

PC remote windowing is a vital new ingredient of network computing, and it puts a very different spin on the whole network computing and network computer story:

- **PC software** moves to **PC servers** and is accessed by PC terminals
- **Collapse of entire PC-LANs** into PC servers: "PC-LAN in a box"

- **User interfaces** from all PC programs can become remotely accessible **network services**
- **All PC software** is therefore easily accessible to network computing
- Web pages and Java applets are downloaded to **browsers in PC servers**
- **Web pages and Java** can then be securely **caged in PC servers**
- **Web and Java user interfaces** will be windowed on PC terminal screens
- **Interactive Web applications** are made easy by embedding virtual PC windows into Web pages
- **PC terminals** are much simpler than **network computers**
- PC terminals are **ultra-thin clients**
- Implemented as simple terminal emulator software in PCs, or as low-cost terminal hardware
- This is network computing **without network computers.**

This analysis indicates that PC remote windowing technology will become a vital ingredient of the Internet and network computing, and part of the mainstream of PC technology. Network computers will have relatively narrow applicability, and PC terminals will have near-universal applicability; "ultra-thin clients" will be more prevalent than "thin clients".

Despite this advance in PC technology, Java will be the mainstream for software in network computing; but Java execution will mostly be securely caged in PC servers, and accessed by remote windowing from PC terminals. This is a win for Sun's software businesses, a win for Microsoft, and a win for Intel; but with strong downwards pressure on their margins and prices, which will be a very big win for customers.

The general conclusion is that most PC software and data will inevitably move away from desktops and into servers; this will happen because the migration path is so easy and the results are so beneficial. Therefore, many enterprises are likely to stop upgrading PCs and buy more servers instead. This is a major threat to PC industry revenue, and it will emerge strongly during 1997.

Some parts of the industry may want to block these changes because of adverse effects on their profitability and market position; but this will only delay the outcome and cannot stop it. PC remote windowing technology is an invention which, like the atomic bomb, cannot be un-invented once it has become known.

These changes create exciting new opportunities for IT users and for systems integrators such as ICL. A future paper will describe how these opportunities are exploited in the OPEN*framework* Intranet architecture for designing network computing solutions.

References

BLOOR, R., "The Enterprise by other means", Bloor Research Group, ISBN 1-874160-21-X, 1996.

BRENNER, J.B., "OPEN*framework* Intranet architecture", http://intra.bra01.icl.co.uk/Architecture

CITRIX Inc., "Windows without walls", http://www.citrix.com

DOLBERG, S, DePALMA, D.A, JOHNSON, J and MAVRETIC, M "Java's FATE", The Forrester Report: Software Strategy Service, Vol 7, No. 2, May 1996.

INSIGNIA, "Insignia solutions", http://www.insignia.co.uk

JEOFFROY, M., "The Internet and how it is used," ICL Systems Journal, Vol 11, Issue 1, 1996

JAVAONE CONFERENCE, 1996; see http://www.javasoft.com/.

Biography

John Brenner is a managing consultant in OPENframework Consulting, which is part of Enterprise Technology, in ICL Enterprises. He joined ICL (then called ICT) in 1961, immediately after graduating from Cambridge University. He has worked in many parts of ICL, mostly in advanced development, and mainly in the field of networking and distributed computing. He has published a number of papers in the ICL Systems Journal, and is the holder of the first software patent awarded to ICL.

Neural Networks

C.J. Hinde,[1] G.P. Fletcher,[2] A.A. West[3] and D.J. Williams[3]

[1]Computer Studies, Loughborough University
[2]Computer Studies, Glamorgan University
[3]Manufacturing Engineering, Loughborough University

Abstract

The paper gives a description of the background to, and mechanisms used, in neural networks. Particular attention is given to the interpretation of neural networks and accelerated training systems. Some applications of neural networks are described and the interpretation of the rules derived from them discussed. Hopfield networks are introduced to show the mechanisms of a recurrent network.

1. Background

There has always been a fascination with the creation or emulation of human like intelligence. Boole's work [Boole, 1854] about the process of thought was clearly stimulated by this fascination. Gottlieb Frege's [Frege, 1879] refinement of Boole's work later that century also focused on the manner of thought. The symbolic school, characterised by this kind of work, dealt with the performance of thought, the way thought and intelligence were, or should be, conducted but had little to say about how thought and intelligence came into being, or about the mechanisms by which the substance of the brain could manage to perform the thought processes attributed to it. Research in artificial intelligence may be roughly divided into two areas: the first attempted to develop and explain intelligence per se and perhaps to incorporate it into intelligent artefacts and the second which tried to explain human intelligence. Clearly the two areas drew much from one another [Pylyshyn, 1978].

The substance of the brain is made up of around 10^{11} interconnected cells called neurons, or neurones, of which the essential details are shown in Figure 1. Griffith [1966] provides a simple explanation of the biochemical basis of neurons:

> "The cell is bounded by a surface membrane which is semi-permeable. This means that the membrane will allow certain

ions and molecules to pass through it, but not others. The semi-permeability has certain consequences which can be understood using the theory of thermodynamics. One of these is that the interior of the cell normally has a negative electrostatic potential of about -70mV with respect to its exterior. However, if this potential difference is artificially altered above a threshold value of about -60mV, the membrane suddenly becomes permeable... ...This change of permeability is self propagating, it starts at one point and spreads to adjacent points and so on. Shortly afterwards the membrane recovers its semi-permeability. Thus a transient burst of electrochemical activity is generated."

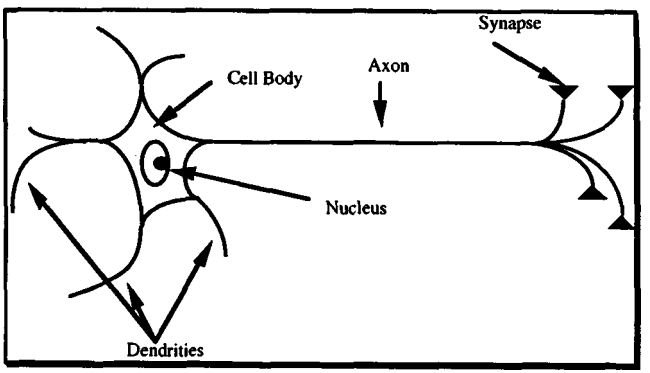


Figure 1: The neuron consists of a nucleus, inputs which are known as dendrites and outputs known as axons. The connection between the outputs of a neuron and the inputs of another neuron are via the synapses.

Even this model is grossly simplified, but the mathematically idealised neurons are even simpler. It is tempting to believe that the large number of neurons in the brain can be trained or educated to emulate almost any function, this may be the case, however Wasserman [1972] shows convincing evidence that the connectivity of the brain is largely predetermined and learning only activates, inhibits or changes what is already there.

The study of how these interconnected neurons compute and collectively behave is the study of neural networks. One of the early landmark papers written by McCulloch and Pitts [1943] caused great excitement. Although McCulloch and Pitts did not claim that their mathematical idealisation of a neuron was correct it laid the foundation for the work

that followed. Ashby [1952] used the ideas of McCulloch and Pitts to put forward his "Design for a Brain." Ashby's realisation that it wasn't enough just to connect together enough neurons and it would de facto be intelligent resulted in his foundation work in cybernetics [Ashby, 1958].

Rosenblatt [1962], in proposing the perceptron, with both single layer and multi layer configurations, provided the second substantial milestone in the development of neural networks, after McCulloch and Pitts. Others seeking to explain the way the brain works proposed other models. Hebb [1949] proposed the idea of reverberating neurons as a mechanism for memory, although he mainly focused on trying to explain human behaviour and intelligence. Hopfield's recurrent net [Hopfield, 1986] was based somewhat on Hebb's ideas but extended beyond brain emulation to applying the concept to optimisation problems, such as that of the travelling salesman.

Rosenblatt showed that a suitably large and multi-layered set of idealised neurons could emulate any Boolean function, but did not suggest any way in which these sets of neurons could be trained from examples, unless the configuration was restricted to a single layer. This single layer perceptron became popular and a comprehensive treatment of them is given by Nilsson [Nilsson, 1965]. The limitations of perceptrons were largely ignored at the time until Minsky and Papert [1969] wrote their seminal book on perceptrons, pointing out the limitations and also proving many theorems about them. Research on neural networks then faded somewhat.

2. Feed forward nets

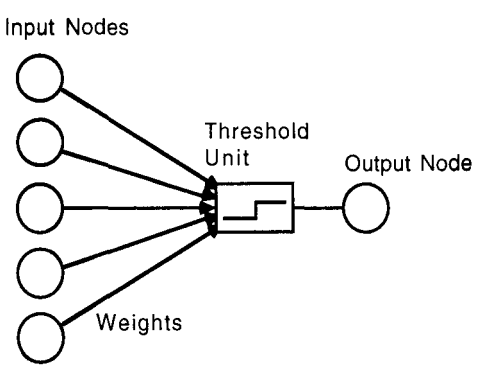


Figure 2: The block diagram of an idealised neuron.

The basis of the perceptron is a combination of linear decision surfaces and threshold logic units. These linear decision surfaces determine what the "hurdles" are that should be crossed before the neuron switches on or "fires." There is a set of "input" nodes, corresponding to the features of a pattern, which feed into output nodes that correspond to the pattern classes. These output nodes take their values from a weighted sum of the features and, if this sum is above a certain value, the output node "fires" and the pattern is placed in one class, otherwise it is placed in the other. Figure 2 shows a block diagram of the essential components of an idealised neuron.

This system implements a straight line in two dimensions, a flat surface in three dimensions or a hypersurface in spaces of more than three dimensions.

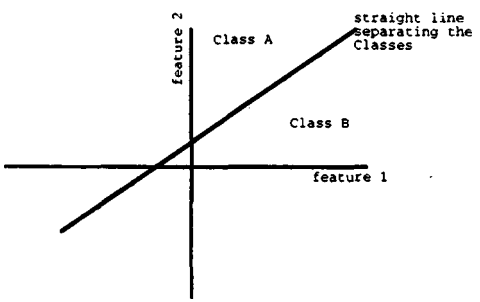


Figure 3: A threshold neuron essentially implements a straight line which separates the two classes A and B

The form for separating two patterns P_1 and P_2 is:

$$P_1 \quad if \quad \sum_{i=0}^{i=n} W_i.F_i > 0$$

$$P_2 \quad if \quad \sum_{i=0}^{i=n} W_i.F_i < 0$$

for features F_i from 1 to n and where W_i are the associated weights and $F_0 \equiv 1.0$.

This may be generalised to separate any number of classes with each class having its own weight set, threshold unit and output node. This is illustrated in Figure 4.

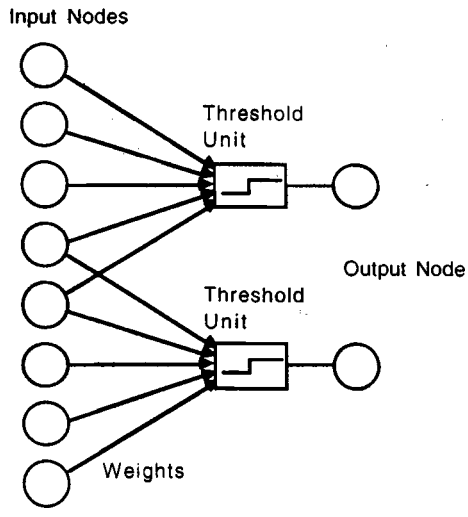


Figure 4: Two neurons taking from similar input sets but implementing different decision surfaces.

By associating a set of weights with each pattern class, the strength of the match may be obtained from the system and the greatest value used to indicate recognition. If the system has been trained correctly then the presence of other patterns is taken into account in each set of weights and so it is possible that in a perfectly trained system only one node will fire. Thus there is a set of weights, W_{ji}, where the suffix, j, denotes the pattern class and the suffix, i, denotes the feature. Recognition then proceeds by multiplying the weights by the features and choosing the largest combined value. The value of $\sum W_{ji}.F_i$ is the tensor product of W_{ji} and F_i giving a vector of scalar discriminants.

2.1 Some Limitations

Such machines or programs can recognise a large class of patterns but there are several classes of pattern that cannot be recognised and these are those with "global" properties. In psychological terms, these are the Gestalt patterns where the relationships within the pattern are relevant and the whole is greater than the sum of the parts.

Given a set of features, or inputs, F_1 and F_2 then, if the output is classified as shown in Figure 5, it clearly requires two straight lines to separate the points in the set from the points not in the set.

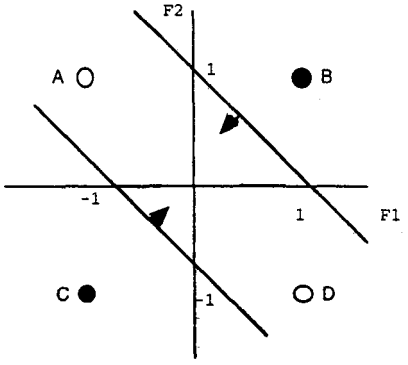


Figure 5: The open points A and D cannot be separated from the shaded points B and C with only one straight line.

The Boolean operator corresponding to the pattern shown is the "exclusive or" or "XOR." In higher dimensions this is generally represented as parity.

Patterns are linearly separable over a given set of features if there exists a set of weights which will distinguish the pattern classes. "XOR" is not linearly separable.

Various functions can be implemented, however, using a single layer perceptron. For example, a threshold logic unit can implement the Boolean functions AND and OR.

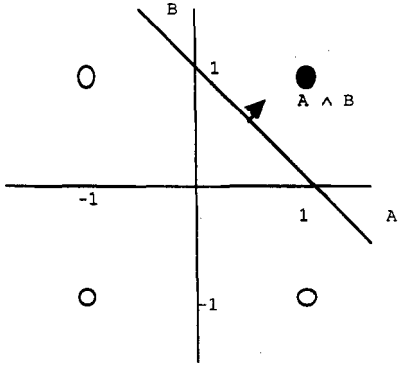


Figure 6: The surface representing an AND gate.

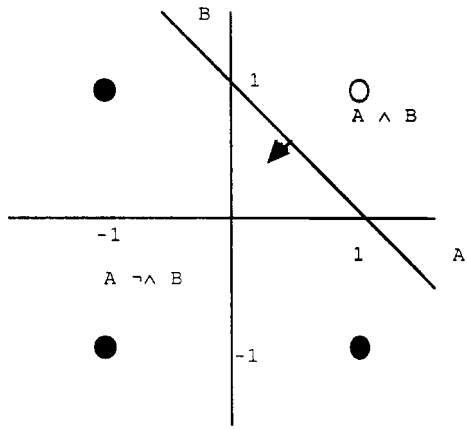


Figure 7: The surface representing a NAND gate.

The AND, OR, NAND and NOR gate surfaces are all easily implemented in a single node.

2.2 Non linearly separable patterns.

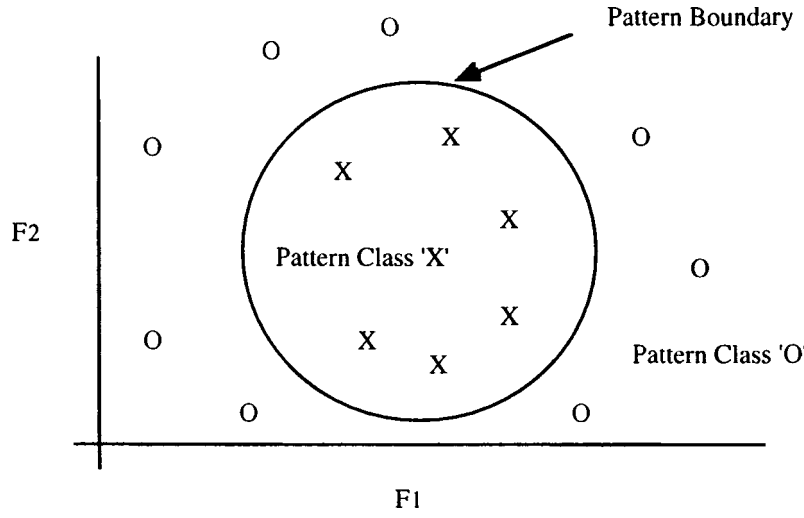


Figure 8: A non-linearly separable pattern

If the pattern class is not linearly separable over a given set of features it may be separable over some algebraic function of the set; for example a quadratic. The Polynomial ADaptive LInear NEuron, referred to as a

PADALINE, can give considerably more power to a single layer net. If the pattern class is separable over a quadratic surface then if some of the old features can be combine into quadratic functions then the patterns may be made linearly separable over the extended set of features. For example, let F_n be $F_m * F_m$. A non-linearly separable space is illustrated below which is separable over an extended set of features.

Clearly it is not possible, in Figure 8, to separate the '*X*' patterns form the '*O*' patterns using a single straight line in the feature space defined by F_0, F_1 and F_2. It is possible, however, to separate them over the space defined by $F_0, F_1, F_2, F_1 * F_1$ and $F_2 * F_2$ as the circle is a conic section. If the circle has centre (C_1, C_2) and radius R in the F_1, F_2 plane, then the equation of the circle is:

$$(F_1 - C_1)^2 + (F_2 - C_2)^2 = R^2$$

The patterns may now be grouped ('*X*' class positive) using the conditions:

$$X \text{ if } R^2 - (F_1 - C_1)^2 - (F_2 - C_2)^2 > 0$$

$$O \text{ if } R^2 - (F_1 - C_1)^2 - (F_2 - C_2)^2 < 0$$

These expressions may be rearranged as:

$$X \text{ if } R^2 + 2C_1F_1 + 2C_2F_2 - F_1^2 - F_2^2 - C_1^2 - C_2^2 > 0$$

$$O \text{ if } R^2 + 2C_1F_1 + 2C_2F_2 - F_1^2 - F_2^2 - C_1^2 - C_2^2 < 0$$

If F_3 is substituted for F_1^2 and F_4 is substituted for F_2^2 then weights may be assigned to the variables, as follows:

$$W_0 = R^2 - C_1^2 - C_2^2$$

$$W_1 = 2C_1$$

$$W_2 = 2C_2$$

$$W_3 = -1$$

$$W_4 = -1$$

Although the separating circle has not been replicated over the original space, it has been possible to separate the points in an extended space. In general, giving the correct inputs to a system, or using the correct representation, results in a simpler analysis and implementation [Amarel, 1968].

2.3 Control

Homeostasis is the property that a particular parameter, e.g. body temperature, must be kept constant and any deviation from this value must be avoided. Homeostasis is the heart of all control systems, based on keeping parameters within limits. The curve for homeostasis is a vertical straight line and pattern vectors must lie ON that line in order to be classified as correct; it may also be regarded as an interval within which the values must lie. It is much more productive if we regard the control problem as one of keeping parameters within an interval. A simple switching controller such as a thermostat cannot operate effectively without such an interval. The classical view of control is outlined in Figure 9 where the inputs to the system are mixed in with an error signal and then fed to the plant through a compensator.

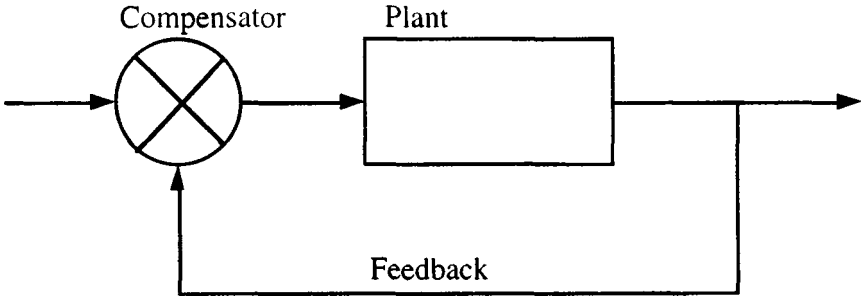


Figure 9. The classical feedback control loop.

The problems and advantages of using a neural network as a compensator will now be explored.

It is easy to emulate homeostasis, as shown in Figure 10, using hidden layers with two intermediate nodes and an interval to represent the desired values.

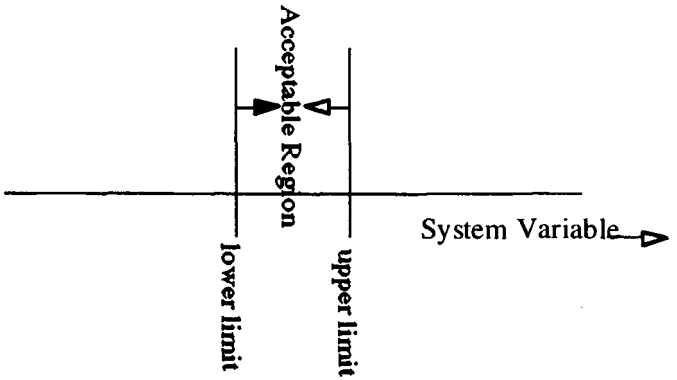


Figure 10. A system variable has to be kept within some limits, each region corresponds to a particular control action.

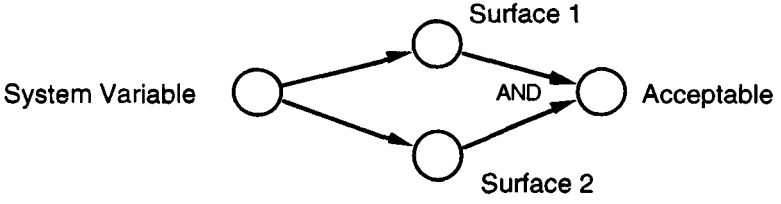


Figure 11. The network implementing a simple "bang bang" controller.

The weight vectors for the two intermediate nodes and the final output nodes for a desired value of 'H' and a tolerance of 'ε' are:

	W0	W1	W2
Surface 1	$-H+\varepsilon$	+1	
Surface 2	$+H+\varepsilon$	−1	
Acceptable	$-2+\delta$	+1	+1

δ is a small quantity, typically 0.1. Note that there is only one input to each of the intermediate nodes, node 1 implements a decision surface that fires if the input value is below $H+\varepsilon$ whereas the second surface fires if the input value is above $H-\varepsilon$, the output node implements the conjunction of both nodes. Effectively the system recognises that the input value is within the interval $(H-\varepsilon, H+\varepsilon)$.

This simple system can be turned into a controller by connecting the outputs of the various nodes to control switches.

Getting the overall arrangement of the network correct is important, otherwise it will be untrainable. The simple net above, designed to keep a system variable within limits, can be implemented easily and a reasonable estimate made of the weights required to implement such control system.

This simple control system can be extended to a second order system which, instead of just switching on and off, can control the amount of power involved. A second order system requires knowledge of its history, so it is necessary to feed back the previous error value as an input. Higher order systems can be constructed similarly.

2.4 Transforming feed forward neural networks.

If a network has been trained so that it has become stable and the outputs are interpreted as either on or off then the network itself must be equivalent to a set of rules. Nodes, which represent Boolean rules, may be transformed quite simply by inspecting the corresponding weights.

The model neuron, that has been used, shows a threshold unit but does not specify it in detail. The logic may be chosen appropriately depending upon what is required to be achieved and using the correct representation allows features of problems to be seen that other representation may hide. In particular, a choice may be made between a 0/1 logic and a –1/+1 logic. So far a –1/+1 logic has been used; thus a positive result (+1 when passed through a threshold) indicates one class whereas a negative result indicates another class. Sometimes it is more convenient to use 0/1 logic, particularly since transforming networks to sets of rules is easier using 0/1 logic. The activation function takes care of this difference and so the basic equations remain the same. The major difference is in the values of the inputs, an input of -1 meaning "off" affects the balance between the weights, whereas an input of 0 meaning "off" has no effect.

To explore the derivation of Boolean functions from a neural network a simple two dimensional network will be examined:

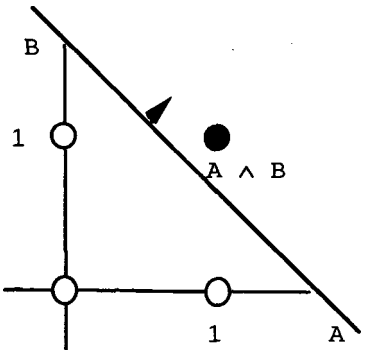


Figure 12: The line implements a 0/1 Boolean AND gate.

Suitable weights for an AND gate in 0/1 logic might be:

$$W_0 = -1.5,\ W_A = 1.0,\ W_B = 1.0$$

Given a simple threshold system and that A and B can only take on the values 0 and 1, i.e. they are Boolean, then the expression:

$$W_0 + W_A * A + W_B * B$$

only becomes greater than 0 if both A and B are equal to 1.

A simple OR gate may be implemented in a similar way.

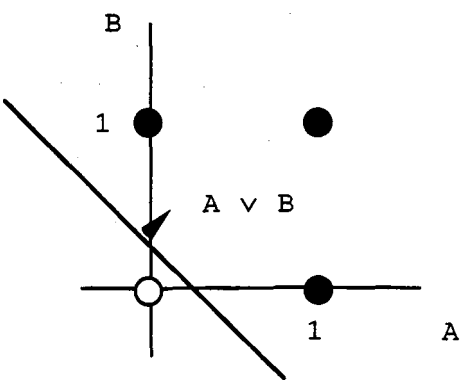


Figure 13: This line implements a simple 0/1 Boolean OR gate.

Suitable weights for an 0/1 OR gate might be:

$$W_0 = -0.5,\ W_A = 1.0,\ W_B = 1.0$$

Given a simple threshold system and that A and B can only take on the values 0 and 1, i.e. they are Boolean, then the expression:

$$W_0 + W_A * A + W_B * B > 0.0$$

is true, if either A or B are equal to 1.

Three or more input neurons may be analysed similarly. The method is like packing a knapsack and may be developed in stages.

First of all a set of weights is taken such that W_0 is negative and all the other weights are positive. Sets of inputs are then selected such that the sum of their weights is just greater than $-W_0$. If no such set of weights exist then W_0 is so negatively large that the neuron will be constantly false and offers no contribution to the network. The sets of weights so chosen are ORed together as sets of ANDed inputs; in other words the result will be in disjunctive normal form.

Taking the above AND gate with the weights:

$$W_0 = -1.5,\ W_A = 1.0,\ W_B = 1.0$$

the total of W_A and W_B is greater than $-W_0$, thus there is only one set of inputs, $\{W_A, W_B\}$, meaning that the neuron implements A AND B. The interpretation of the OR node is equally simple with the weights as:

$$W_0 = -0.5,\ W_A = 1.0,\ W_B = 1.0$$

where both W_A and W_B cover the value of $-W_0$ quite comfortably and so the two sets are $\{W_A\}$ and $\{W_B\}$ giving W_A OR W_B.

Taking a three input system with the weights as:

$$W_0 = -1.5,\ W_A = 1.0,\ W_B = 1.0,\ W_C = 1.0$$

then there are three subsets of inputs that will just cover $-W_0$, they are $\{W_A, W_B\}$, $\{W_B, W_C\}$ and $\{W_A, W_C\}$ and so the Boolean interpretation of this node is $(A \wedge B) \vee (B \wedge C) \vee (A \wedge C)$.

This shows that the representational power of a neuron is greater than that of a single Boolean function,

The above has shown how to transform simple neurons where all the weights are positive except W_0 which is negative. This will not always be the case and so a general case needs to be examined.
What is needed is a transformation of the equation such that it gives the same output with a positive weight and the input associated with the weight inverted. Consider a simple three input system with weights, W_0, W_1, W_2 and W_3 and where W_1 is negative. Let the new weights be from the set $\{W_i'\}$ and then the following sets of conditions must be true:

$$\textit{If } F_1 = 0 \textit{ then } W_0' = W_0 + W_1$$

$$\textit{If } F_1 = 1 \textit{ then } W_0' + W_1' = W_0$$

given $W_0 = -0.4$, $W_1 = -1.0$, $W_2 = 1.0$ and $W_3 = 1.0$.

Then $W_0' = -0.5 - 1.0 = -1.5$ and $-1.5 + W_1' = -0.5$. Thus $W_1' = 1.0$ and therefore:

$$\{W_i\} = \{-0.5, -1.0, 1.0, 1.0\}$$
$$\{W_i'\} = \{-1.5, 1.0, 1.0, 1.0\}$$

All that has to be done is to convert the new all positive set of weights to a Boolean expression and substitute $\neg F_1$ for F_1 to get the correct interpretation.

W' is therefore interpreted as

$$(\neg F_1 \wedge F_2) \vee (\neg F_1 \wedge F_3) \vee (F_2 \wedge F_3).$$

If the bias, W_0, is greater than 0.0 then it has to be modified. A positive bias cannot be part of a valid neuron, if there are no negative weights, as the neuron would always be on. As above the negative weights may be eliminated and a negative bias obtained as a result.

i.e. $\{W\} = \{0.5, 1.0, -1.0, -1.0\}$.

Subtracting F_2 and F_3 from the bias:

$$\{W\} = \{-1.5, 1.0, 1.0, 1.0\}$$

i.e. $E = (F_1 \wedge \neg F_2) \vee (F_1 \wedge \neg F_3) \vee (\neg F_2 \wedge \neg F_3)$

For layered networks the required interpretation for the whole net can be obtained by back substitution; for example, if F_1 was the output of a previous layer and the interpretation of F_1 was $(I_1 \wedge I_2)$ then this would be substituted wherever F_1 occurred.

Thus the expression:

$$E = (F_1 \wedge \neg F_2) \vee (F_1 \wedge \neg F_3) \vee (\neg F_2 \wedge \neg F_3)$$

would become:

$$E = ((I_1 \wedge I_2) \wedge \neg F_2) \vee ((I_1 \wedge I_2) \wedge \neg F_3) \vee (\neg F_2 \wedge \neg F_3)$$

Fletcher & Hinde [1995] extend the idea of transforming neural networks into rules by showing how this can be done more efficiently. Essentially the knapsack algorithm which was used to extract the rules in the above examples grows exponentially with the number of inputs. In the paper cited methods are described for reducing the size of the exponent and also for "cleaning up" the rules. Examples are given that show how to determine the prime implicants of the neural system, where these are not clear from examination of the rules extracted. Using this reduction mechanism it can be made clear where the network has been trained incorrectly and therefore indicates where the training set should perhaps be extended.

3. Vision

There are situations where it is inappropriate to express knowledge about a transformation as a set of rules. Neural networks can be used as pattern classifiers in a vision system, transforming quantised pixels into Boolean classes. Extracting the rules thus induced from training the network on large sets of pixels or patterns can be less than useful as patterns are rarely expressed as rules involving individual pixels. Such

a set of rules would be of little value to a human being taught to recognise patterns.

A much more useful output would be if the network could produce those patterns, or a summary of those patterns, which it recognises as belonging to a particular class. In this sense the outputs of a neural network are being set and the network asked to indicate the corresponding inputs; that is, a feed forward network is being transformed into a feed backward network. Fletcher & Hinde [1996] show how this can be achieved by assessing the significance of the evidence required by a neuron to fire and feeding this back to the inputs. The resulting pattern can then be examined to see if it corresponds to the "general idea" of the training patterns being presented.

4. Training Neural Nets

It has been well known for over twenty years that a multi-layered perceptron, or neural network, can replicate a much larger class of functions than a single layered system. There was no widely known training system until Werbos [1974], and Rumelhart et al. [1986] publicised the back propagation algorithm. It is interesting to note that Bryson [1969] shows a version of the back propagation algorithm but this was not well publicised.

It has been shown above that a neurons can model AND/OR gates and any Boolean function that can be expressed in disjunctive normal form and, therefore, only a two layer system is required to model any Boolean function of propositions. It may be possible to develop a simpler model with more layers but theorems about the minimal sizes of networks are very rare. Messom [1992] proves that a fully connected two layer network with n inputs and 1 output can model any Boolean function with only n intermediate nodes. Earlier work had suggested that the minimum number of intermediate nodes was 2^n.

As all Boolean functions can be reduced to a set of AND and OR nodes in not more than two layers, so this includes ALL rule based systems or expert systems that only use simple statements. The difference is that neural networks are difficult to understand and therefore cannot generally offer the explanation facilities associated with expert systems. The advantage is that neural networks can be trained easily.

The threshold function, as used in the early perceptrons, was a step function, as shown in Figure 14.

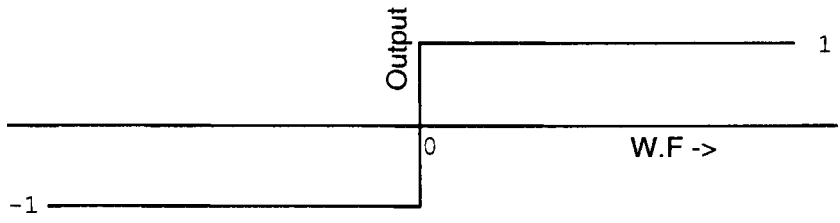


Figure 14: A simple threshold function

With this simple function, it is impossible to determine where the error occurred in a multi-layer network and so correct it. Most neural nets, as they are now termed, use a smooth function called a sigmoid function, as shown in Figure 15.

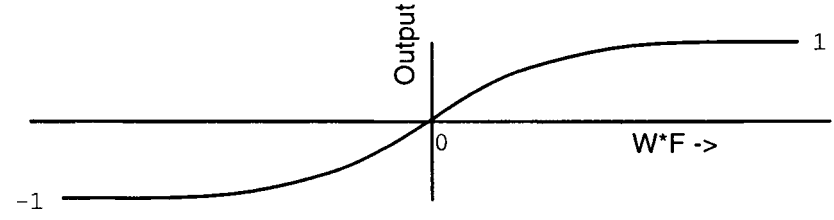


Figure 15: Sigmoid or logistic function

This has the disadvantage that the results are not "crisp." However they can be made "crisper" by steepening the gradient at the crossover point, although this has to be exercised with caution as, when it comes to training the system, it may not stabilise. A compromise can be reached, which will be discussed below.

The particular sigmoid function shown above is bipolar (i.e. gives (–1, +1)). A suitable function is:

$$sig_1(X, T) = 2/(1 + \exp(-X/T)) - 1$$

A 0/1 sigmoid function (giving (0, +1)) may be obtained from:

$$sig_0(X, T) = 1/(1 + \exp(-X/T))$$

The T term may be likened to temperature, thus when reduced the curve is steepened so that decisions are more "crisp" or "solid". Increasing T

allows more exploration freedom for the net but results in less crisp results.

Almost any function of the same form may be used. The notation used is that values of Y represent outputs and values of X represent intermediate values. In general, it will not always be the case that all inputs for a particular neuron will have come from the previous layer, so the inputs are taken to be the set $\{F_i\}$ in order to be consistent with the previous sections. The set $\{W_i\}$ link the input nodes to the current node. The diagram in Figure 16 shows the flow.

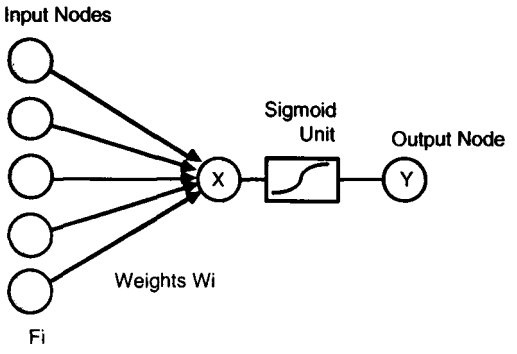


Figure 16: The flow of values through a single neuron.

The values of X are the results of the scalar products of the layer before i.e.

$$X = \sum_{i=0}^{i=n} W_i.F_i$$

This provides a forward propagation system.

$$Y = sig\left(\sum_{i=0}^{i=n} W_i.F_i, T\right)$$

The training algorithms are more complex than for the simple perceptron but many of the same techniques can be used.

A measure of how badly the network is performing with its current set of weights is first obtained. An error measure is computed by summing the squares of all the errors. A term R_c is introduced, which is the

required state of output Y in case c. The error is then the difference between the actual state Y in case c and the correct state R_c. The error measure chosen is given by:

$$E = 0.5 \sum_c (Y_c - R_c)^2$$

the sum is over all cases. Only the correct output in the top (final) layer is known but changes can be induced in the previous layers; this is described later.

This error can be minimised by repeatedly changing the weights by a small amount proportional to the rate of change of E with respect to W. So change to W, ΔW should be proportional to the size of the error caused by W and be of opposite sign. This gives the following for ΔW:

$$\Delta W = -e\,(\partial E / \partial W)$$

for training case c. The value of this training parameter is determined empirically but should be higher for smaller samples and lower for larger samples; 0.01 seems to work quite well as a general purpose value. Choosing a value for e that is too low results in lengthy training sessions, whereas choosing a value that is too high can result in the network weight values oscillating wildly. In conventional control theory, the value of e is analogous to the weight on the proportional component of the control signal. Indeed, much of conventional control theory may be used and there are all the usual problems of overshoot and damping, plus the fact of working in discrete time steps.

This training procedure is guaranteed to find the set of weights that gives the least squares error, i.e. minimises E. This minimisation is done by gradient descent and so the minimum computed may in fact be a local minimum and not the global minimum generally required. $(\partial E / \partial W)$ cannot be computed directly as E is not expressed as a direct function of W so intermediate variables need to be used.

$$(\partial E / \partial W) = (\partial E / \partial Y)\,(\partial Y / \partial X)\,(\partial X / \partial W)$$

These terms can be expressed as more tractable and understandable quantities:

$(\partial E / \partial Y)$ is the error associated with the output in case c and so is merely the quantity, $(Y - R_c)$.

$(\partial Y / \partial X)$ is the differential of the sigmoid function at the point given by X. Inspection of the sigmoid function shows that the rate of change is greatest near the changeover point (origin) and so changes the weights that have settled to large values the least with the weights that are nearly changed the most. The network weights are changed by the least amount possible commensurate with getting the value right.

The differential of the simple threshold function is zero at all points except the origin where it is infinite and so is of no value in an analytical approach to a solution.

The final term $(\partial X / \partial W)$ is given by differentiating:

$$X = \sum_{i=0}^{i=n} W_i . F_i$$

which is simply F_i.

Finally, therefore:

$$(\partial E / \partial W_i) = (Y - R_c)\left(\partial(sig(X, T)) / \partial X\right)(F_i)$$

And so the amount of correction to apply to the weights can be calculated accordingly as:

$$\Delta W_i = -e\,(Y - R_c)\left(\partial(sig(X, T)) / \partial X\right)(F_i)$$

The amount of error, for which the previous layer was responsible, may also be deduced and made available. The quantity, ΔF_i, needs to be calculated in terms of known quantities, such as the weights and the errors in the previous layer. Each error in this layer, ΔY, will have received contributions from the errors of ΔF_i. What is required is a similar quantity for F_i that was obtained for W_i. In that case:

$$\Delta W_i = -e\,(\partial E / \partial W_i)$$

and so a similar function for F_i would be:

$$\Delta F_i = -e\,(\partial E / \partial F_i)$$

The e in this case though could be a different one to the one used to modify the weights.

Following through the same logic as before a similar chain of differentials may be used giving:

$$(\partial E / \partial F_i) = (\partial E / \partial Y)(\partial Y / \partial X)(\partial X / \partial F_i)$$

this is identical except for the last term where the differentiation is performed with respect to F instead of W and so

$$X = \sum_{i=0}^{i=n} W_i . F_i$$

differentiated simply gives W_i.

Combining this with the differential of the sigmoid gives:

$$\Delta F_i = -e(Y - R_c)\left(\partial\left(sig(X, T)\right) / \partial X\right)(W_i)$$

This is then summed over all nodes affected by the node that is about to be corrected.

The time taken to train a net can be quite long and training cycles of several thousand can be employed. It is also possible that the system can get "stuck" in a local minimum.

There is NO way whatsoever of "correctly" assigning the error correction. The best that can be hoped for is that with sufficient training samples the best "democratic" solution is obtained. Essentially, if all the nodes vote that a node is too low over all cases then there is a pretty strong case for raising it.

The sigmoid function, used above, had a term referred to as temperature which affected the gradient of the sigmoid function. Typically, the sigmoid function can be steepened for any given set of inputs by either increasing the weights associated with the various inputs, or decreasing the temperature, thereby giving a larger input to the sigmoid function resulting in answers closer to the [0,1] limits. This temperature can be altered to change the sigmoid function as it becomes appropriate. The same back propagation policy can be applied here as before and so the correction strategy would be as before:

$$\Delta T_i = -e\left(\partial E / \partial T_i\right)$$

Fletcher and Hinde (1994) showed that by using a generalised activation function convergence could be accelerated and smaller networks could also be obtained.

5. Activation functions

Having explored the sigmoid function it will have become clear that one of the main properties that enable training to be performed is its differentiability. There are a lot of other differentiable functions that could be used instead; in particular *sin* is useful. Like the *sigmoid, sin* is an odd function but is also periodic. The disadvantage is that on its own it cannot be sharpened to a threshold function by merely increasing the weights. Fletcher and Hinde [1994] solved this problem by superimposing a sharpening function over the basic activation function.

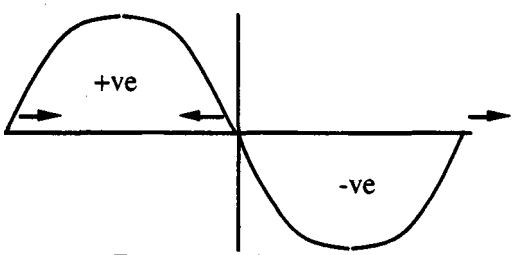


Figure 16: A sine wave.

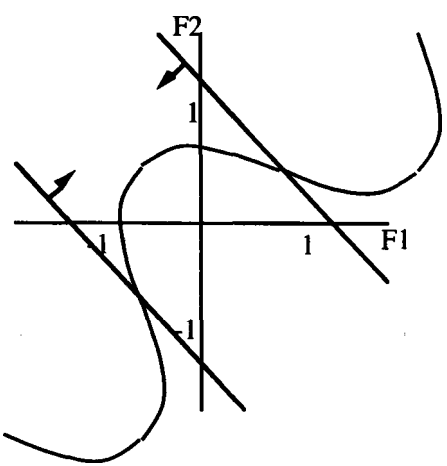


Figure 17: A set of decision regions implementable by a 'sine' activation function.

Although the sigmoid function is almost perfect for implementing functions such as AND, OR, NAND etc., it is unable to implement XOR, which, as has been shown, required two lines to separate the concept of XOR. The *sin* function provides an infinite set of parallel lines. Figures 16 and 17 show that a *sin* activation function is almost perfect for implementing XOR, although less than perfect for implementing the other conventional Boolean functions. A combination of both, however, can implement any Boolean operator in one neuron.

6. Designing a Neural Network

The foregoing gives much of the theory of neural networks, however it is only of modest help when actually faced with the problem of setting up and training a suitable neural network. It is also relevant to ask what sort of applications are suitable for a neural network. Neural networks are able to map a set of inputs of arbitrary size on to a set of outputs between 0 and 1 or between -1 and +1 depending on how the network has been constructed. They are most suitable for mapping Boolean or real valued inputs on to Boolean outputs, they are also very useful where there is a large training set available and where there is no agreed set of rules or algorithms for getting from the input values to the output values.

A suitably designed network will reflect the problem domain and so some experimentation is usually required before a suitable network is found. If a network can be found that is fairly close to the correct network, or even one that just isn't too far away, then this can be refined until it is optimal. Strangely a network that is far too large tends not to converge very well; unfortunately the same is true for one that is too small and undersupplied networks will not converge at all. By large and small is meant the number of hidden nodes and the number of layers. For real valued inputs, the first layer implements a region separation layer. Although it is theoretically possible for this layer to implement several disjoint regions, this doesn't happen in practice unless the initial weights are close to the required weights. The initial weights tend to implement lines going through or near the origin and the gradient descent approach will move these outwards and will tend to approximate one large region. For a real valued input there is a quantisation layer which separates the real values into regions, followed by a hidden Boolean layer followed by the outputs.

Having selected an initial structure for the network, it is then trained on a suitable set of training data. Convergence is then checked on a separate testing set, as this increases the confidence that the network

has converged to the right function. The number of training epochs or cycles through the training set can be several thousand and a training run of 30,000 epochs is not unusual or unreasonable. However, with a suitably structured network and a set of weights, which have been selected with knowledge of the function to be implemented, the number of cycles can be reduced to a very small number indeed.

After the network has been trained with, hopefully, very small errors, most of the weights will have become relatively large. For all layers, except the first, any links that have small weights associated with them may be removed. A Hinton diagram, which highlights the large weights, is useful for this and a cut of the small weights is known as a Hinton cut. This leaves a smaller network, or at least one with fewer connections, and any node with no incoming or outgoing connections can be eliminated. Experience has shown that cutting out the small weights has little or no impact on the performance of the network and the associated errors are changed very little. In one experiment every connection with a weight numerically less than 4 was removed and the network remained error free with no further training. On the other hand, when the weights greater than 5 were removed the network made a large number of errors and could not be trained. Again from experience this technique of training and cutting appears to work well and results in robust concise networks. Although the values of 4 and 5 were appropriate for a particular case, in general the values to cut have to be determined empirically.

The errors made in the network in question were of the order of 0.01 for a Boolean output and it was possible to prune the network from 96 connections to only 22. This also had the beneficial side effect that the network ran considerably faster. Reasonable rules could also be extracted from the pruned network, whereas the larger network gave very confusing results.

7. Hopfield nets

Hopfield nets are recurrent nets which are typically used as a model for content addressable memory. The network has neurons which essentially update one another and rather than feed forward or backward, they feed one another. The neurons have an update rule which updates the neuron values according to some schedule, which may be random, until a minimum has been reached.

The diagram in Figure 18 shows a simple three node Hopfield network and the interconnections between the nodes. The interconnection weight

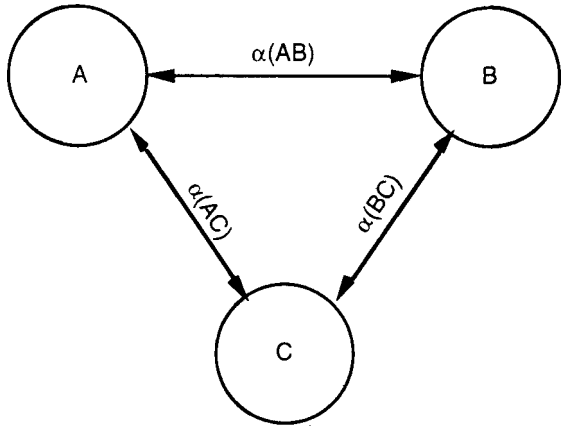


Figure 18: A three node Hopfield network showing the three nodes *A, B, & C* and their connections.

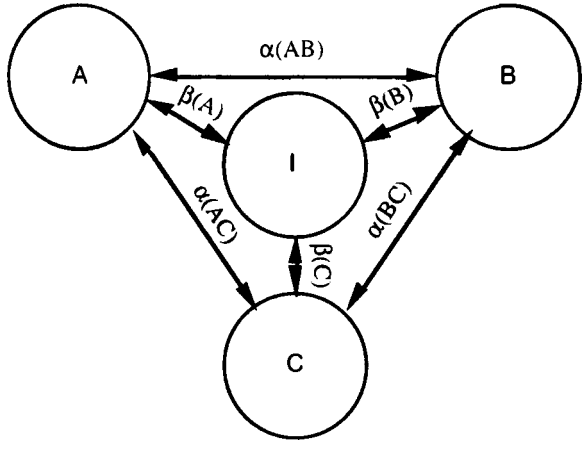


Figure 19: The constant node I provides a "bias" for the nodes *A, B & C*. The association $\alpha(A, I)$ between the node A and the constant node I is rewritten as $\beta(A)$ to denote that it is the bias of the node *A*.

between, say, the two nodes, *A* and *B*, is denoted by $\alpha(A, B)$. Each neuron is updated according to some update schedule according to whether its input neurons multiplied by their associations are greater than or less than 0.0. This implies a simple threshold function but, of course, a sigmoid could be used if it was required to denote values in (-1.0,1.0)

instead of {-1.0,1.0}. Using a sigmoid in this situation, however, does not provide any real advantage and the learning rules are different. What this simple diagram doesn't show is the possibility of a bias in the neurons.

It is desirable to contain all the parametrisation in the network in the associations and so a constant node *I*, whose value is 1.0, is introduced, see Figure 19.

Given an understanding of feed forward networks, it is not difficult to determine what the weights should be for a Boolean AND. If *C* is *A & B* then $\beta(C)$ could be −1.5, $\alpha(A, C)$ could be 1.0 as could $\alpha(B, C)$. If the values of *A* and *B* are set appropriately and then *C* is updated, *C* will take on the value of *A & B*. In fact, in this case, since there is only one free node requiring one update, the network resembles a feed forward network rather than a recurrent network.

If the number of nodes and the number of free nodes are allowed to rise, then the system is one which will settle into one of a number of minimum energy states. These minimum energy states should correspond to the trained or remembered patterns. The training system for a Hopfield network is based on Hebb's learning rule and involves basing the associations between two neurons on the frequency with which they coincide. This is very similar to a correlation coefficient except that no assumptions are made about the normality of the data. So the association $\alpha(A, B)$ is based on the training pairs of the values of *A* and *B*. Common sense suggests that if *A* and *B* take on the same value for each of the training pairs then their association is 1.0, if they take on opposite values then the association is −1.0, and if there are an equal number of matches and mismatches then the association is 0.0. Formally, the association between two neurons over a training set *T* is given by:

$$\alpha(A, B) = (\text{matches} - \text{mismatches})/(\text{matches} + \text{mismatches}).$$

This provides a training rule for setting up a recurrent network based on a set of training data and an update rule for extracting stored patterns.

The difficulties occur in almost precisely the same places with the Hopfield networks as they do with feed forward networks, XOR, for example, requires a "hidden node." The problem is that, unlike the feed forward networks, there is no back propagation method and so the hidden node needs to be trained deliberately as part of the training

process rather than induced by back propagation, as in the feed forward networks.

Nonetheless, in many practical applications extra nodes are not required as the size of the pattern being stored or retrieved is sufficient to include enough neurons to provide the necessary "slack." Hertz, Krogh and Palmer [1991] give an analysis, which indicates the reliability of Hopfield networks and provides concrete performance indicators. Essentially, the proportion of patterns that can be stored in an N neuron network is proportional to N, never higher than $0.138N$ if a few errors are acceptable, and is proportional to $N/log(N)$ if it is necessary to recall most of the patterns perfectly. If it is borne in mind the difficulty of training the Hopfield net with "hidden" nodes then the number of patterns in the basic Hopfield model is limited by the size of the retinal plane. Overall, it would appear that the feed forward model is more controllable and can be better understood than the Hopfield model

8. Applications

8.1 An Adhesive Dispensing Machine

Our example application is the dispensing of adhesive in the manufacture of "mixed technology" P.C.B.s in which through hole and surface mount components are present on the same board. The surface mount components are secured to the board, prior to a wave soldering operation, by a small (0.0002 to 0.005 CCs depending on component) amount of adhesive. The amount of adhesive dispensed is critically dependent upon several process variables (e.g. temperature, humidity, erratic thixotropic behaviour of the adhesive, air bubbles in the flow and variations in the P.C.B. substrate).

The dispensing unit consists of a syringe of adhesive coupled to a pressure control unit. The unit consists of a solenoid valve, pressure regulator, temperature sensor and a pressure transducer to monitor the variation of pressure within the syringe. The dispensing unit is fixed to a SEIKO RT3000 robot which moves the syringe to locations on the P.C.B., where the adhesive is to be dispensed. Feedback data is collected by an image processing system (Imaging Technology ITI 151) coupled to a Pulnix TM-460 CCD camera, incorporating a magnifying optical system.

The original software was developed using the MUSE real time AI tool kit. MUSE is a hybrid modular system supporting a range of knowledge

representation paradigms; PopTalk, a procedural language with object oriented programming extensions, a forward chaining rule language, a backward chaining language, data directed programming through the use of daemons and support for general relations between objects.

Particular support for real time operation includes agenda based priority scheduling, interrupt handling and fast data capture.

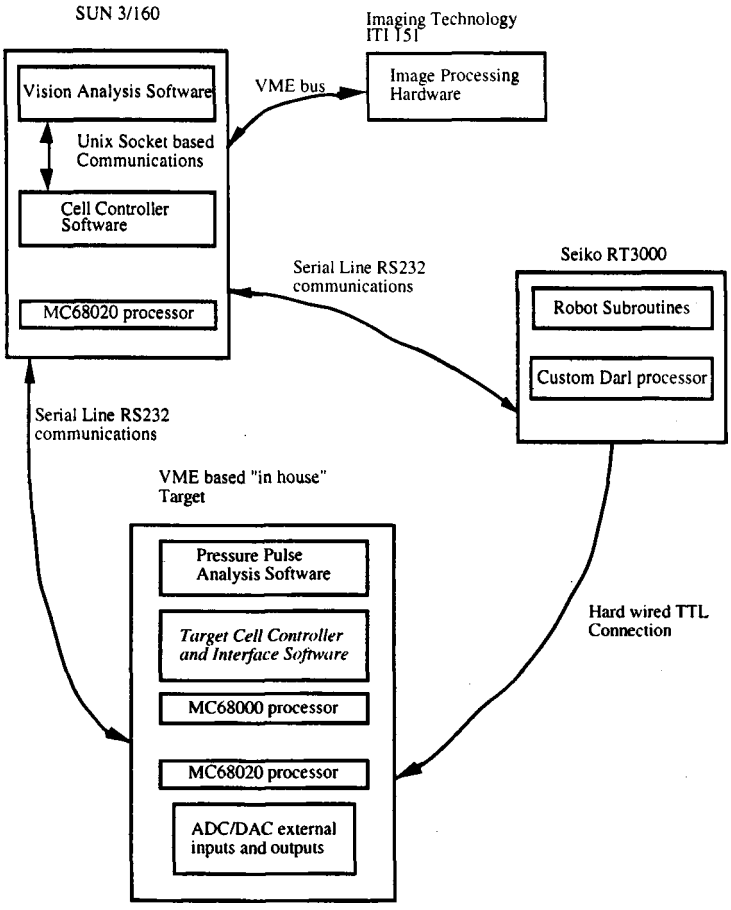


Figure 20: The hardware configuration of the adhesive dispensing system.

The physical distribution of the individual processors and the software functions performed by the adhesive dispensing control system is illustrated below. The system consists of four processors (one SUN, two

Target processors and one robot processor) with a pipeline vision processing system attached to the VME bus of the SUN. Communication between the software modules is either via the VME bus, serial line communications or the UNIX sockets mechanism. The "in-house" built Target was designed to allow the cell controller software, developed on the SUN to be ported to a "black box" solution suitable for an industrial system

A neural network system, Figure 21, was derived for controlling the adhesive dispensing machine. This is the type of problem that has typically been tackled by the use of a rule induction package. The trained neural network should therefore be equivalent to a set of rules that could have been learnt by such a package.

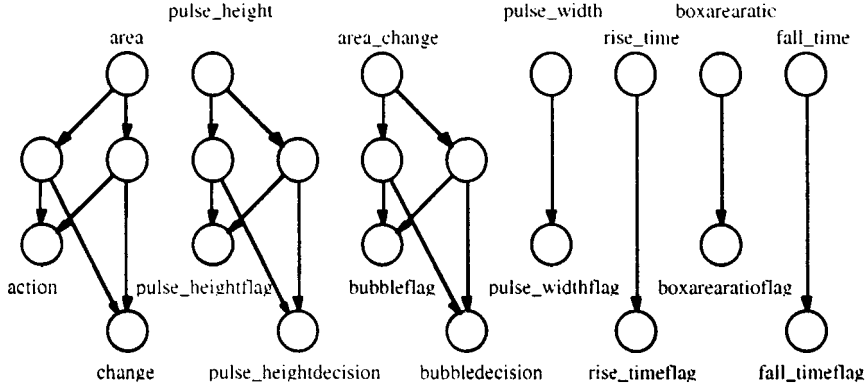


Figure 21: The net implemented for controlling an adhesive dispensing machine

Step one in the analysis is to remove the quantisation layer of the network and to express this part as a set of inequalities. This step does not simplify the information but does display it in a manner that is much more natural to read than a set of weights on a diagram.

```
IF  rise_time < 1.25
THEN rise_timeflag
ELSE ¬rise_timeflag
.
.

IF  area > 1.02581868
THEN midnode1
ELSE ¬midnode1
```

```
.
.

IF  area > 0.972214383
THEN midnode2
ELSE ¬midnode2
IF  pulse_height < 1.025
THEN midnode3
ELSE ¬midnode3
.
.

IF  pulse_height > 0.975
THEN midnode4
ELSE ¬midnode4
.
.
```

After this the remainder of the network can be converted to a set of Boolean functions using the method presented above. This will produce a set of rules in a very similar format.

```
IF  (midnode1 / ¬(midnode2))
THEN action
ELSE ¬action

IF  ¬((midnode1 / midnode2))
THEN change
ELSE ¬change
.
.
```

It is now possible to implement directly this set of rules in a rule base without altering the action of the system. For clarity some general simplification of the Boolean rules is required and substitution of the inequalities produces much more natural looking results. These processes produce the final rules for the network.

```
IF  rise_time < 1.25
THEN rise_timeflag
ELSE ¬rise_timeflag
.
.
IF  (area > 1.02581868 / area < 0.972214383)
THEN action
ELSE ¬action
```

The network, from which these rules were derived, was trained using simulated data derived from the rule set implemented to control the adhesive dispensing machine. These rules are similar to the rules implemented to control the machine and serve to illustrate the usefulness of the interpretation system for a medium sized control network. This type of result is most useful when it is necessary to check that what the network has learnt is reasonable and when it is necessary for a system to explain its actions, something classic neural networks cannot do. A more complete comparison of the control strategies used may be found in West, Williams & Hinde [1995].

8.2 Robot dynamics

An application that is almost perfect for neural networks is that reported by [Miller, Hewes, Glanz & Kraft, 1990] on real-time dynamic control of an industrial manipulator using a neural-network-based learning controller. The manipulator was equipped with a neural net at each joint where each network was given inputs taken from all the joints and was intended to correct the power to each joint, given that the positions, velocities and loads of each joint were known, so that the robot would describe the correct path under load and while in motion.

The mathematics of multi axis robot manipulators is complex and the constants will vary from robot to robot, even between those of the same make. The manipulator was trained to describe various paths under load and then tested on a variety of other paths. The experiment proved a success and training using a circle under load proved to be adequate as most situations occurred during that particular training cycle. It is clear that using dynamic feedback as training data a robot with a partially trained network can be further trained to learn the dynamics of the individual robot.

As there is no "operator knowledge" available, the mathematics is difficult and would need to be repeated for each individual robot. Expert system correction is likely to be difficult, as is development of a fuzzy controller unless some form of rule induction or fuzzy learning is applied. It is likely that the only reasonable choices in this situation are either a neural network or a fuzzy learning system.

9. Conclusions

This paper has covered a large area and indicated some of the significant results which are interesting and may be useful. It has shown that it is possible and useful to extract rules from neural networks

and that examination of these rules can help to verify that the network has "learnt" the correct concepts. Rule extraction can also be used as an aid to data mining although as with all data mining exercises care and common sense must be used to avoid jumping to strange conclusions. Where rules are inappropriate to describe the hypothesis of a network a feed backward system has been described and exemplified. The mechanisms and theory of training networks are described and the advantages of extending the activation function shown. Hopfield networks are described, however they are similar in many respects to feed forward and many of the results shown earlier are applicable. The applications are shown where appropriate in the text, however the control of robot dynamics is especially interesting as there are few techniques available for this particular problem. The adaptive and noise resistant nature of neural networks is especially useful in industrial situations.

References

AMAREL, S., "On representations of problems of reasoning about actions," Machine Intelligence 3 (ed. Michie, D.), American Elsevier, New York, 131-171, 1968.

ASHBY, W.R., "Design for a Brain," Chapman and Hall, 1952.

ASHBY, W.R., "An Introduction to Cybernetics," Chapman and Hall, 1958.

BOOLE, G., "An Investigation of the laws of thought on which are founded the mathematical theories of logic and probabilities," 1854.

BRYSON, A.E. and HO, Y.C., "Applied Optimal Control," New York, 1969.

FLETCHER, G.P. and HINDE, C.J., "Learning the Activation Functions for the Neurons in Neural Networks," ICANN'94, 1(1 & 2), Marinaro, M. and Morasso, P.G. (eds.), Springer-Verlag, Proceedings of the International Conference on Artificial Neural Networks, Sorrento, Italy, May 1994, pp 611-614, ISBN 3-540-19887-3, 1994.

FLETCHER, G.P. and HINDE, C.J., "Using Neural Networks as a Tool for Constructing Rule Based Systems," *Knowledge Based Systems*, 8(4), August 1995, pp 183-189, ISSN 0950-7051/95, 1995.

FLETCHER, G.P. and HINDE, C.J., "Producing evidence for the hypotheses of large neural networks," Neurocomputing Vol 10 no 4, 1996.

FREGE, G., "Begriffsschrift, a formula language, modelled upon that of arithmetic, for pure thought," in van Heijenoort 1967, pp. 132-213, 1879.

HEBB, D.O., "The organisation of behaviour," Wiley, 1949.

HERTZ, J., KROGH, A. and PALMER, R.G., "Introduction to the theory of neural computation," Lecture notes Volume 1 Sante Fe Institute Studies in the sciences of complexity, Addison Wesley, 1991.

HINTON, G.E., "Connectionist learning procedures," Artificial Intelligence, Vol 40, 185-234, 1989.

HOPFIELD, J.J. and TANK, D.W., "Computing with Neural Circuits: A Model," Science, 233, pp 625-633, 1986.

MCCULLOCH, W.S. and PITTS, W.H., "A logical calculus of the ideas immanent in nervous activity," Bull. Math. Biophys. Vol 5, 115-133, 1943.

MESSOM, C.H., "Engineering reliable neural networks," Ph.D.thesis, Loughborough University, 1992.

MICHIE, D., "Machine Intelligence 3," American Elsevier, New York, 1968.

MILLER, W.T., HEWES, R.P., GLANZ, F.H. and KRAFT, L.G., "Real-time dynamic control of an industrial manipulator using a neural-network-based learning controller," I.E.E.E transactions on robotics and automation, Vol. 6 No. 1, 1990.

MINSKY, M.L. and PAPERT, S., "Perceptrons," M.I.T. press, 1969.

NILSSON, N., "Learning Machines," McGraw-Hill, 1965.

PYLYSHYN, Z.W., "Computational models and empirical constraints," The Behavioural and brain Sciences 1, pp 93-127, 1978.

ROSENBLATT, F., "Principles of neurodynamics," Spartan books, 1962.

VAN HEIJENOORT,J., "From Frege to Godel: a source book in mathematical logic," 1879-1931, Harvard University Press, 1967.

WASSERMAN, G.D., "Molecular Control of Cell Differentiation and Morphogenesis: a systematic theory," Marcel Dekker, New York, 1972.

WERBOS, P.J., "Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences," Ph.D. Thesis, Harvard University, USA, 1974.

WEST, A.A., WILLIAMS, D.J. and HINDE, C.J., "Experience of the Application of Intelligent Control Paradigms to Real Manufacturing Processes," IMechE Part I: Journal of Systems and Control Engineering, 209(I495), pp 293-308, 1995.

Biographies

Chris J Hinde

Chris Hinde has been a senior lecturer in Computer Studies at Loughborough University since 1991. Previously he worked in Computer Aided Architectural Design at Bristol University and on production control systems in industry. He holds a degree in Mathematics from Bristol University and a PhD in Cybernetics from Brunel University. His research interests are broadly in Artificial Intelligence and specifically in Neural Networks and their interpretation and verification. He has worked on the computer aided design of golf clubs, small electro-mechanical devices and computer aided architectural design. Work in fuzzy systems includes a version of Fuzzy PROLOG, derivation of fuzzy operators from example expressions and the expression of neural network hypotheses as fuzzy sets.

Graham P Fletcher

Graham Fletcher holds a BSc degree in computer science from Royal Holloway College, London and obtained his PhD in artificial intelligence from Loughborough University in 1995. Since 1995 Graham has been a lecturer at the university of Glamorgan specialising in the formal analysis of neural systems.

Andrew A. West

Andrew West holds a BSc degree in Physics and obtained his PhD in Astrophysics from Leeds University in 1987.

From 1987 he has been involved in Manufacturing Engineering research at Cambridge and Loughborough Universities. He is currently a member of the Manufacturing Systems Integration Research Institute, Loughborough University and a lecturer in the Department of Manufacturing Engineering. His current interests are the application of AI techniques to manufacturing and the design and control of the next generation of reusable, reconfigurable manufacturing systems.

David J Williams

David Williams has been Professor of Manufacturing Processes at Loughborough
University since 1989 and was Head of Department for four years until 1995.

He holds a First Class BSc in Mechanical Engineering from UMIST and a PhD in Engineering from Cambridge University. His industrial experience includes a student apprenticeship with GKN, line specification and installation with Metal Box (now Carnaud Metalbox) and a part time sabbatical leave with Lucas Electronics.

He has held (with others) EPSRC, European and industrial contracts to a value approaching £3m. He has published over 200 papers and edits two journals.

His appointment at Loughborough followed six years as a Lecturer in Engineering at Cambridge University and a three year Research Fellowship funded by Ford of Britain.

David Williams is a Fellow of both the IMechE and the IEE.

Short-term currency forecasting using Neural networks

O.V.D. Evans

Advanced Research, ICL Bracknell, Berkshire

Abstract

The use of an artificial neural network is one way of predicting the behaviour of time-sequential data. A method of training such a network to forecast the behaviour of a selected foreign exchange (Forex) price 5 to 10 days into the future is described that has achieved acceptable precision over the time interval selected. The method depends not only on selecting the right training data but, equally importantly, on filtering it of noise fluctuations so that a behavioural 'model' of the data sequence can be accommodated in a relatively modest number of neurons. Details are given of the back-propagation neural network and the Kalman filtering process used, so that similar systems may be designed for other sequential data.

1. Introduction

The powerful pattern recognition properties of neural networks have been recognized for some considerable time in capital markets and have in many cases outperformed contemporary modelling techniques.

The merit that neural networks share with non-parametric regression techniques and genetic algorithms is that they can model the behaviour of systems in which invariant behaviour is not observable—or at least not to the extent that rules can be extracted from them. Such data-rich environments are termed *model-weak,* and model estimation depends on the use of statistical inference techniques such as non-parametric regression. From a statistical point of view, a neural network can be regarded as such a model.

Ideally, strong models, where justifiable, are preferable to weak non-parametric ones—but can introduce bias into the modelling process if the assumptions about the data are not sustainable. In weak models no such *a priori* assumptions can be made about the data. All that can be said about the data is inherent in the observations.

It is clearly an advantage for organizations trading in the currency exchange markets to be able to work with the strongest possible models they can justify since this gives them an insight into the short and long term fluctuations in the exchange rates, and they can plan suitable trading strategies.

Any other kind of organization trading internationally also has, perforce, an interest in the behaviour of foreign exchange (Forex) markets if only so that international cash transfers can be timed, as far as possible, to occur during periods when the exchange rate is least adverse for the transactions concerned. That this occurs as a matter of course can be verified by anyone who has settled an overseas expense with a charge card—the time taken for the charge to be debited to the payer can vary considerably since the card company is clearly working its own optimization. For this purpose, some form of short term prediction of Forex trends is sufficient since there is a limit to the time that charges can be delayed before settlements. For the same reason these is little to be gained from detrending the data since settlements may not correspond to seasonal variations.

This paper describes the results of an exercise in the application of a neural network to the short term forecasting of exchange rates by first training the network on published historical data and then using the same data to assess the degree to which the resulting forecasts ranging from five days ahead onwards corresponded to the actual rates. For the purpose of the exercise the network was trained on data derived from Forex rates based on the four principal international currencies, US$, German DM, Japanese ¥ and £UK, processed in various ways, with the £UK/DM cross-rate used as the target currency for forecasting purposes. In practice, any of the currencies used can be treated as the target currency for forecasting purposes, or any other currency in which settlements are regularly made.

The body of the paper describes the network configurations investigated and the steps needed in processing the raw Forex data into a training set and a recognition set for testing their performance, the data filtering needed to improve learning times and forecasting precision, and concludes with the results achieved.

These results indicate that with the data available, a reasonably reliable forecast for five days ahead seems to be achievable. However, a *caveat* needs to be added; such systems are only able to perform their function reliably under conditions approximating to a 'steady state' and

cannot be depended upon when exchange rates are being influenced by extraneous episodic events such as wars, significant changes of Government and the like. It would clearly be up to the user to refrain from using the system during such times of instability.

2. Neural Networks

Neural networks originally started as a way to simulate the behaviour of networks of biological nerve cells—known as neurons. Each simulated neuron, like its biological counterpart, is designed to carry out a simple threshold calculation by collecting signals at its multiple inputs and summing them. If this sum exceeds a set threshold, the neuron 'fires' by sending out its own signal. The transformation between input sum and output is determined by a non-linear function which can vary from a simple 'gap acting' (hard on/hard off) to the more common sigmoid (S-shaped) response.

2.1 Back-propagation networks

Although the simulated neuron bears some similarity to its biological precursor, the neural networks built around them have now evolved into a number of standard types that bear little resemblance to their biological equivalents other than the ability to learn and recognize patterns. For the purpose of the current exercise, the network used is known as a standard back-propagation multilayer perceptron henceforth termed the ANN—for Artificial Neural Network.

Networks of this type comprise an input layer of neurons, one (occasionally two) hidden layer(s), and an output layer. Every input layer output connects to every hidden layer input. Likewise every hidden layer output connects to every output layer input. The key feature of these connexions is that they are *weighted*. At the start of a training session, the process whereby an ANN is 'taught' to recognize a particular set of input patterns, these weightings are set to random values. During the course of a training session, these values are changed by the process of back propagation, so that by the end of the training session the 'memory' of the ANN is represented entirely by the analogue values of all the weightings in the ANN.

Figure 1 is an example of a typical neural network—in this case with eight input layers, thirteen hidden layer units and a single output layer, the latter representing the currency for which a forecast is desired.

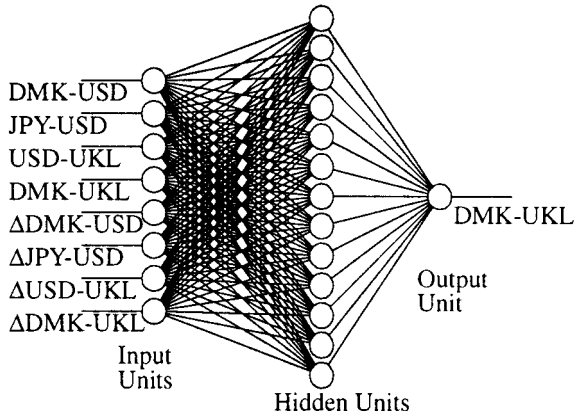


Figure 1: Example of an ANN

Each cycle of the learning process starts with the generation of an error vector from the difference in value of the input signals as propagated forward through the ANN to the output units and a target 'expected' value vector, which is then used to update the weightings in two stages, first the output unit weightings, and then the hidden unit weightings. Appendix A describes the precise algorithms used for training the ANN. The learning process is cycled through the set of training patterns until either each of the error signals falls below some pre-set tolerance threshold value, or a preset number of learning cycles is completed. The rate at which the error vector elements will converge down to the set threshold, or whether it converges at all, will depend on the nature of the input data and on the size of the hidden layer. If the data is full of conflicting associations, these will inhibit the learning process due to the presence of what are termed *malicious* vectors. As will be seen later, techniques exist for filtering out their worst effects. The number of hidden units determines the total number of weighted links between the various layers. Since the 'memory' of the ANN resides in these links, enough of them must exist for the training to be recorded, and is therefore related to the number of training patterns being used. However, having too many hidden units may cause the ANN merely to memorize the training patterns—which would prejudice the reliable recognition of approximate matches. The rule that has been adopted is to start with less than the optimum number and work up until a satisfactory compromise is reached between learning rate and level of recognition. In practice the recognition rate remained substantially constant over a wide range of hidden layer sizes and one could be chosen to minimize the number of learning cycles.

Figure 2 shows the results achieved—resulting in a choice of 15 hidden units as a good compromise between high learning cycle counts and the learning time instability evident to the right of the plot[1].

In the Forex application, the network is trained on input data preceding the target exchange rate by the desired forecast interval. Then, with historical data available, the ANN can be assessed for precision of prediction. Before this can be done, the available data has to be subjected to a number of preprocessing steps.

Normally an ANN works best with inputs that fall within the central near-linear section of the sigmoid function range. Since the weights are initially set to random values in the range —0.1 to 0.1 it is necessary to normalize the input values to ensure that these limits are not exceeded. Since currency exchange rates over the short term rarely fluctuate more than one or two percent points outside their mean, it is clearly necessary to do something more that merely scale the values if any significant reaction is to be expected from the ANN.

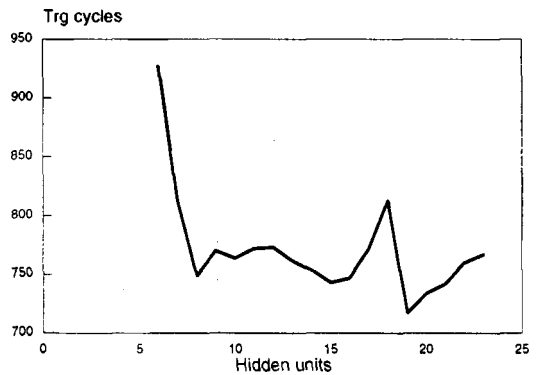


Figure 2: Effect of hidden layer on learning cycles

The policy adopted has been to remove completely the invariant component of the exchange rates used and then to scale up the fluctuating component to fill a range between 0.1 and 0.9 for the set of values to be used for training. This tactic is applied to all input data regardless of its nature. A corollary of this is that the output of the recogniser then needs to be converted back if the results are to have any practical significance. For example, the US$/DM (bid) rate during the

[1] These results were achieved with filtered data—see Section 3 below

period from 15 February 1996 to 7 May 1996 recorded a minimum of 1.4464 (DM/$) and a maximum of 1.5341 in daily mean rates. By removing the constant component of the minimum value and renormalising, these values become 0.1 and 0.9 respectively, with all other rates recorded during the period being somewhere in between.

2.2 Selection of data and training strategy

The obvious place to start in selecting training data is with the exchange rates associated with the leading traded convertible currencies, the US$/DMark, £/US$, US$/JP¥ and £/DMark, the last being a cross-country pair included as it is the rate selected for forecasting trials. By themselves these inputs are not regarded as sufficient for learning purposes since price movements are non-stationary and quite random in nature and need to be supplemented with other Forex-derived data [Mehta, 1995].

If $P_1, P_2,...P_t$ is a price sequence sampled at regular intervals, *then* $P_2 - P_1, P_3 - P_2,... P_t - P_{t-1}$ are said to be the *first differences*. They are considered to be the best way of generating data sets for neural network learning, and they have been included in the input data set in the ANN under discussion. In this particular case, the first differences of the above price sequences have been taken using the raw data and then subjected to the normalization treatment described above. The use of first differences, along with the normalized Forex prices themselves, have improved the learnabilty, compared to using just the latter, to the extent that supplementary inputs such as the FTSE-All Share Index or random noise have not been needed in order to ensure convergence. All the results reported are based on a network input configuration that includes both the four Forex prices listed above and their first differences.

The data selected for conversion for network training consists of the closing average bid prices of the four selected currencies during the 72 (UK) working days starting 15 February 1996 and ending on 29 May 1996. Accurate historical Forex data is available on a daily basis on the World Wide Web [Olsen, 1996], which includes such data as mean bid/ask prices, the daily spread and the number of transactions contributing to the statistics.

The training strategy adopted was to create training data sets by taking successive overlapping 15-day sequences each advanced by one day on its predecessor and using as an 'expected result' target the £/DMark rate five days after the last set in the sequence. Table 1 shows what

the starting data of a typical sample looks like before processing. The right-hand column contains the target forecast prices, which can be seen to be five days in advance of the entries in the fourth column.

$/DM	*$/JY*	*£/$*	*£/DM*	*5d f/c*
1.4752	106.18	1.5239	2.249	2.2585
1.476	106.17	1.524	2.25	2.2584
1.4837	106.55	1.5203	2.2559	2.2586
1.4822	106.46	1.521	2.2543	2.2585
1.4751	106.43	1.5255	2.2505	2.2705
1.481	107.45	1.5246	2.2585	2.2782
1.4809	107.37	1.5245	2.2584	2.2703
1.4802	107.0	1.5255	2.2586	2.2703
1.48	106.98	1.5255	2.2585	2.2774
1.4906	108.07	1.5251	2.2705	2.2764
1.4987	108.45	1.5197	2.2782	2.2732
1.5013	108.6	1.5115	2.2703	2.277
1.502	108.56	1.5122	2.2703	2.2805
1.5103	108.48	1.5075	2.2774	2.2935
1.508	108.16	1.5092	2.2764	2.297

Table 1: Typical source Forex data for training

3. Filtering

The rate of convergence achieved with the processed raw Forex prices and their first differences turned out to be disappointingly low, with training times averaging 7038 cycles for the 15 training patterns used and with some patterns still not converging to within a 5% RMS error after 12000 training cycles. Raw Forex rates, however, are basically averages of random price movements within bounds determined by the minimum and maximum daily excursion. If the assumption is made that these fluctuations are uncorrelated, then a smoothing technique called Kalman filtering can be applied to the Forex data.

Kalman filtering is probably rather less well known than neural networks. A summary description of how it works is therefore given in Appendix B, since the use of the technique as an analysis tool in capital markets is quite common, in addition to its original role in control systems. The great merit of using Kalman filters is that the only assumption that has to be made is that the system being observed is subject to random noise—the system itself may be subjected to random fluctuations, and likewise any measurements of the state of the system.

Since practically all actual systems are noisy and since this is particularly the case with the kind of data available from the operation of financial and capital markets, this is an assumption that is easy to justify. The only other prerequisite to using a Kalman filter, which follows from the first assumption, is that the variance of the noise components should be known—although the technique remains very useful even if only an intelligent guess can be made at the noise components.

With the Forex data available it is possible to do better than make a guess. One source of Forex historical data [Olsen, 1996] publishes minimum and maximum excursions for each price, together with the 75% interquartile values. From these figures it is possible to quantify the 'noisiness' of the daily rate and then derive its variance. The process used is described in Appendix B.

$/Dm	*$/¥*	*£/$*	*£/Dm*	*Δ$/Dm*	*Δ$/¥*	*Δ£/$*	*Δ£/Dm*	*£/Dm*
0.3627	0.46	0.5269	0.3087	0.4542	0.4108	0.1456	0.1	0.4026
0.37	0.4582	0.5284	0.3185	0.5626	0.6027	0.5725	0.6126	0.4016
0.4403	0.5273	0.4716	0.3769	0.7289	0.7081	0.415	0.7299	0.4036
0.4266	0.5109	0.4823	0.3611	0.5072	0.5811	0.5974	0.5503	0.4026
0.3618	0.5055	0.5514	0.3235	0.3723	0.5973	0.7549	0.4976	0.5213
0.4156	0.6909	0.5376	0.4026	0.6855	0.8811	0.5311	0.7802	0.5974
0.4147	0.6764	0.5361	0.4016	0.541	0.5838	0.5642	0.5862	0.5193
0.4083	0.6091	0.5514	0.4036	0.5265	0.5054	0.6098	0.5934	0.5193
0.4065	0.6055	0.5514	0.4026	0.5386	0.6	0.5684	0.5862	0.5895
0.5032	0.8036	0.5453	0.5213	0.7988	0.9	0.5518	0.876	0.5796
0.5771	0.8727	0.4624	0.5974	0.7386	0.7081	0.3446	0.7731	0.548
0.6008	0.9	0.3365	0.5193	0.606	0.6459	0.2285	0.3994	0.5855
0.6072	0.8927	0.3472	0.5193	0.5602	0.5946	0.5974	0.5886	0.6201
0.6829	0.8782	0.275	0.5895	0.7434	0.5838	0.3736	0.7587	0.7487
0.6619	0.82	0.3012	0.5796	0.488	0.5189	0.6389	0.5647	0.7833

Table 2: Processed training data ready for use

The results of the application of Kalman filtering to the processed and normalized Forex prices are shown in Table 2. The table also illustrates how the preprocessing widens the small price excursions into a range suitable for training the network.[2]

Not only did this improve the precision of the forecasting process as described in the next section, but also reduced the mean training time for

[2] In practice, some additional processing involving the repetition of the last five patterns and the random shuffling of the pattern order is applied before a training run.

the fifteen patterns used from over 7000 cycles to 446—over an order of magnitude. A further measure that can be taken if considered necessary would be to take the first differences from the filtered, rather than from the raw, data.

4. Results

To assess the consistency of prediction for the network configuration adopted with the data preprocessing steps taken, a series of 58 overlapped runs were made starting with Day 0 of the Forex data and advancing each successive training run by one day. In each case, fifteen successive working days of Forex data were processed to produce a training pattern with a training target being the £-DM rate five days ahead—as shown in Table 2. After each training run a recognition run was executed using the entire 72 days-worth of recorded Forex data, pre-processed in exactly the same way as the training sets. Since the output of each recognition run is a sequence of 5-day predictions of the £-DM price, is a simple matter to compare them with the actual price not only for that day, but also for any other days. In this way it can be tested whether or not the trained ANN can be used to predict prices over intervals greater than five days.

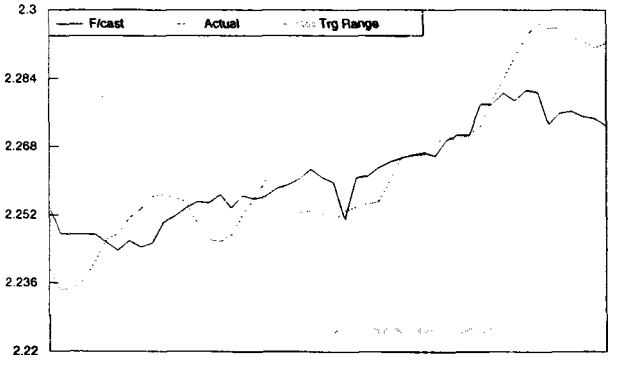


Figure 3: £/DM price: Actual vs Forecast for one training run

Figure 3 shows a typical plot of forecast value against actual price. In this case the output of the ANN has had the pre-processing reversed so that the comparison with the raw price is direct. Also, the comparison has been made with the filtered price, since this can be regarded as a more accurate estimate of price trend than the observed rate. As

expected the predicted price tracks the actual price within the error limits set during the training period, but diverges progressively as the prediction interval lengthens. Training the ANN to a smaller error allowance than the one used in the figure (0.075), is not a useful option, since, apart from considerably lengthening the training time, the effect of overtraining is to reduce the ability of the network to make good matches for inputs that only approximate the training set values. This is easily confirmed by calculating the root mean square error between the two plots for the entire 72-day sequence. This will start to deteriorate as the training allowance is progressively reduced. As can be seen from Figure 4, the effect of varying the error tolerance is so slight for the measured sample that the value of 0.075 is a good compromise between speed of learning, and the ability of the network to generalise from its training. Little benefit is to be gained from setting a wider tolerance since cases have been encountered where the combination of initial random network weightings and the training data values all give rise to outputs within the set tolerance level and no learning will occur at all. The behaviour of the network in those cases is purely random and any apparent recognition purely coincidental.

Although probably not very useful, because of the existence of historical data, the ANN is also capable of retrospective price prediction.

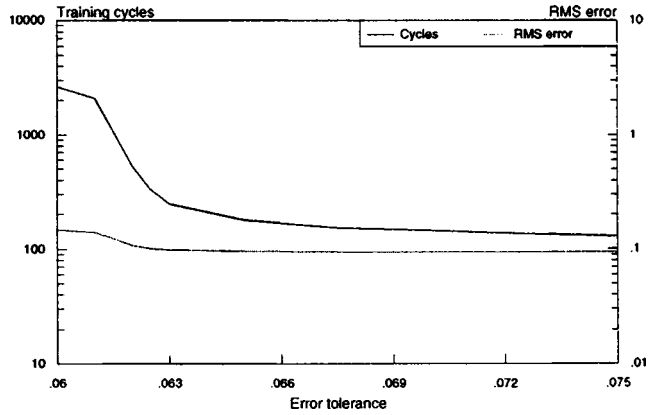


Figure 4: Sensitivity to training tolerance

For each of the 58 runs, the prediction performance of the network was measured for the 5-day point immediately following the training set,

and for successive days thereafter to the 10-day point. For each run the error between the predicted and actual price was calculated as a percentage of the actual price range used for training purposes. The mean value for all the 5-day forecasts, 6-day forecasts, etc., up to the 10th day were taken as representative of the performance of the network across the sample price range. The results are plotted in Figure 5 both for filtered data and for unfiltered data. The mean error achieved on the 5-day forecasts with an untrained network on the sample data is 24%. This happens to be fortuitously low due to the price being on a rising trend during the sampling period and because an untrained network will give a substantially constant output regardless of input, so that input data close to the intersection will give misleadingly optimistic results.

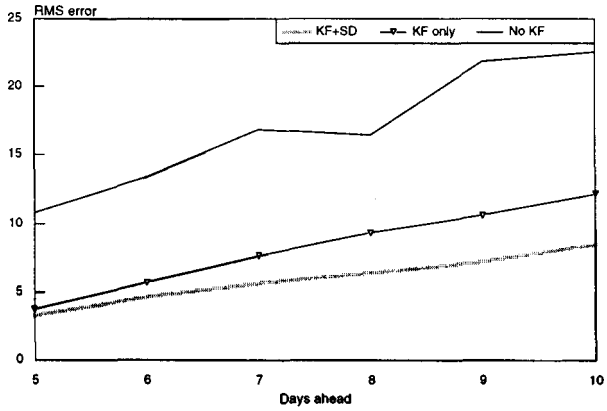


Figure 5: Dilution of forecast precision

The effect of the Kalman filtering is seen to reduce the 5-day prediction error for unfiltered data down to the equivalent of nine days in the case with filtering. The bottom trace illustrates the effect of both repeating the training values for the most recent five days and then randomly shuffling the order of the resulting 20 patterns. This tends to reduce the probability of repeated bad patterns being juxtaposed and has a significant added effect on prediction accuracy.

The results to-date are large enough for variance analysis to be applied. For the sample used all the forecast intervals from five days through to ten pass the standard χ^2 goodness of fit tests at the $\alpha = 0.05$ level of significance, as shown in Table 3 where n is the number of

degrees of freedom for chi-square after the standard restrictions to the number of rankings, K have been applied. This is a somewhat unexpected result in that a network trained on prices five days into the future has proved to be consistent enough to predict prices on a timescale outside its training parameters. The degree of fit achieved is shown in Figure 6, where the shaded bars correspond to the observed density plot of forecast values and the white bars the nearest fit normal density distribution.

Days f/c	σ^2	K	n	χ^2_n	$\chi^2_{n:0.05}$
5	14.40	11	8	9.639	15.51
6	26.40	11	8	2.292	15.51
7	31.98	13	10	10.67	18.31
8	31.52	15	12	19.86	21.03
9	34.35	14	11	10.87	19.68
10	34.90	14	11	11.57	19.68

Table 3: Goodness of fit results

If a plot is made of the successive 5-day forecasts achieved during the trial period and the result compared against the original (Kalman filtered) price, it will be seen from Figure 6 that the forecast follows the ups and downs of the actual price fairly closely. Since the forcasts are the result of successive training runs there is no divergence as in Figure 3.

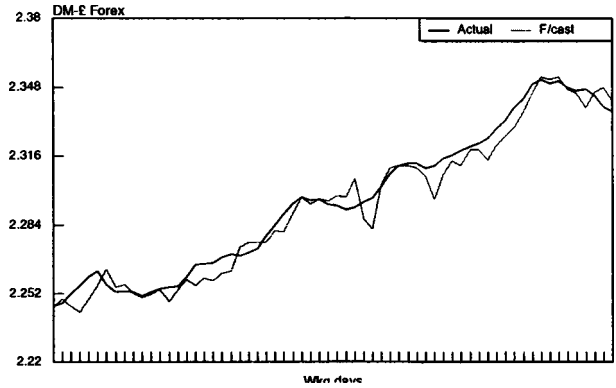


Figure 6: £/DM price: Plot of successive forecasts against actuals

5. Conclusions

The main conclusions to be drawn from the results so far are that with the pre-processing and smoothing described above, not only can a statistically significant price forecast be made for the currency cross-rate selected, but that training on data for a five day forecast is also capable of giving reliable results up to ten days ahead. The results show that the effects of Kalman filtering have the single most significant effect on forecast accuracy. Speed of learning was also a factor in determining the extent to which the raw data needed to be pre-processed. If the input prices and their differences were normalised to between zero and one instead of between 0.1 and 0.9 sometimes no significant convergence occurred. Later trials have produced promising results where the allowable excursion on the first difference inputs have been restricted to between 0.2 and 0.8.

The work to-date has concentrated on forecasting the £/DM price in order to study the viability of short term currency exchange rate prediction. The next step is to examine whether the performance of the ANN is maintained over a longer sample, say 6 months to a year, and to try the predictor on currencies that are not cross-rates on the principal traded world currencies but are of more immediate interest to commercial treasury departments.

A number of further possibilities remain to be tried to refine the performance of the predictor:

- Expand the sample size so that statistical variance analysis can be applied to a larger sample—it will be useful to ascertain whether prediction accuracy is maintained for a full year, taking into account seasonal variations

- Evaluate the ability of the neural net to model the behaviour of the selected currency movements over progressively longer time intervals. This will involve running a series of trials to measure the combined effect of the number of hidden units and the number of training patterns on reducing the mean square error resulting when a long series of input data are presented to a trained network

- Lengthen the prediction interval to a month or quarter by training the network on weekly averages and treating the daily swings as noise for filtering purposes.

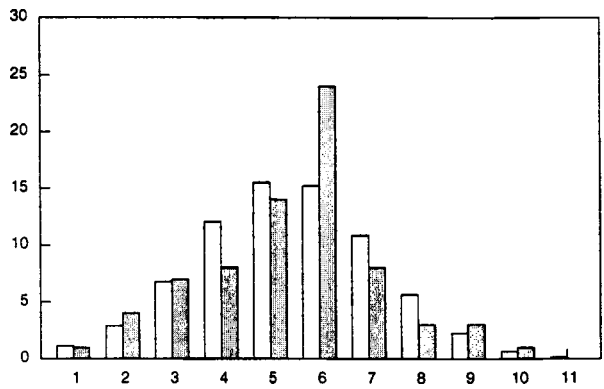


Figure 7: χ-squared goodness-of-fit: 5-day forecast

Appendix

A: Neural Network basics

Neural network technology is now in the public domain and at least 33 different implementations are freely available over the Internet [NeuroNet Web site]. The implementation described below was written in Prolog and follows closely a model originally written in Mathematica—also cited in the reference.

The neural network configuration adopted for use in the Forex trials described is a three-layer multilayer-perceptron (MLP) with training via back propagation—also known as a supervised neural network. The layers are respectively the input units, the hidden units and the output units. For each step of a training session, the input data is presented to the input units and forward propagation occurs to produce an output—this is the same process used for recognition. An error value is then calculated by comparing the outputs of each output unit with a corresponding target 'expected' value. This error vector is then back-propagated to update the input weightings of the output and hidden layers. Training is continued until all the elements in the error vector fall below some pre-set threshold.

A.1 Forward Propagation

The input layer units have unitary transformation functions. For each unit in the hidden and output layers, the sum of the weighted inputs to each unit is transformed by means of an *asymmetric sigmoid function* (Figure 6). The following functions apply during forward propagation:

$$x_i = sigmoid\left(\sum_{j=1}^{N} W_{ji} + \theta_i\right); \quad sigmoid(x) = 1/(1+e^{-x})$$

where for the N inputs to each unit, x_i is the sum of the inputs i_j to a unit, each respectively multiplied by the current weighting value W_{ji} at that input. The output is the this sum transformed by the sigmoid function. The weightings are initialised to random values before training begins.

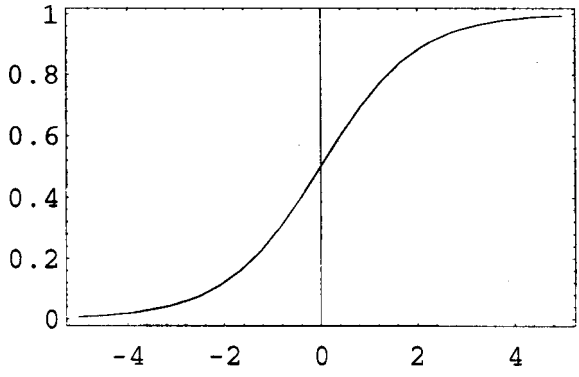


Figure 8: Asymmetric sigmoid function

A.2 Back Propagation

Forward propagation is common both to the training and recognition phases of a neural network. Back propagation is used only during the training phase. The system adopted in this case is based on the *generalised delta rule* [Rumelhart et al, 1986].

Error propagation starts at the output layer with the calculation, for the k^{th} output unit at the p^{th} pattern, of the quantity $\delta^o_{pk} = \delta_{pk} o_{pk}(1 - o_{pk})$ where o_{pk} is the unit output. The weight update to be

applied to each of the inputs from the hidden layer to the output units is given by

$$W^o_{kj(t+1)} = W^o_{kj(t)} + \eta\delta^o_{pk}\, i_{pj}$$

where η is the learning-rate parameter, i_{pj} is the input from the j^{th} unit of the hidden layer and t is the time step corresponding to the p^{th} pattern.

Since the outputs of the hidden layer do not contribute directly to the output delta-error value, a different delta value based on the weighted sum of the output deltas is used instead, derived as follows:

$$\delta^h_{pj} = i_{pj}(1 - i_{pj})\sum_{k=1}^{M} \delta^o_{pk} W^o_{pk}$$

where M is the number of output units. The weighting update for the i^{th} input unit to the j^{th} hidden unit is then given by:

$$W^h_{ji(t+1)} = W^h_{ji(t)} + \eta\delta^h_{pj} x_{pi}$$

where x_{pi} is the input from the i^{th} input unit, again for pattern p.

It is common practice to add a *momentum* term to the delta-weighting values to improve the rate at which the network learns its training set. This involves adding a fraction of the last delta-update to the next, to act as a smoothing factor. The delta updates then become

$$W^o_{kj(t+1)} = W^o_{kj(t)} + \eta\delta^o_{pk(t)} i_{pj} + \alpha\delta^o_{pk(t-1)}$$

$$W^h_{ji(t+1)} = W^h_{ji(t)} + \eta\delta^h_{pj(t)} x_{pi} + \alpha\delta^h_{pj(t-1)}$$

Typical values of η and α range from 2 to 5 and from 0.4 to 0.5 respectively. θ is an optional offset that can be applied to the input to the sigmoid—it can be given non-zero values to help some classes of network to converge. The ANN used was implemented in Prolog.

B: Kalman filters

B.1 Basic Theory

The treatment that follows applies to systems where the system state can be represented at any given time k by a single value $x(k)$. All scaling factors and linear transformation terms are therefore also single

valued. In the general case the state of the system would be represented by a vector and the transformation terms by matrices. For a more general treatment of the subject an appropriate textbook is recommended [Meditch, 1969].

For any system, in general

$$x_{k+1} = Ax_k + Bu_k + w_k \quad (1)$$
$$y_k = Cx_k + v_k \quad (2)$$

where (1) is referred to as the process equation and (2) as the observation equation. y_k is an observation at time k and x_k is the system state at that time; w_k and v_k are process and observation noise components respectively and A, C are linear scaling factors.

The estimate $\hat{x}_{k|k}$ of the state at time k is a linear combination of the observation at time k and a prediction of the state at time k, based on data up to time $k-1$:

$$\hat{x}_{k|k} = J_k \hat{x}_{k|k-1} + K_k y_k \quad (3)$$

where $\hat{x}_{k|k-1}$ is the estimate of the state at time k based on data up to time $k-1$. y_k is an observation at time k and J, K are weighting factors.

Ideally, the difference between the estimated state and the actual state should average over time to zero; that is, if E is an averaging operator,

$$E[\hat{x}_k - x_k] = 0 \quad (4)$$

A *minimum mean square error (mmse)* can be defined:

$$\min E[\hat{x}_k - x_k]^2 \quad (5)$$

Substituting into (3) from the observation equation (2), the state equation becomes

$$\hat{x}_{k|k} = J_k \hat{x}_{k|k-1} + K_k(Cx_k + v_k) \quad (6)$$

and taking expectations

$$E[\hat{x}_k] = J_k E[x_k] + K_k CE[x_k] + K_k E[v_k] \quad (7)$$

The noise term v_k can be asumed to have zero mean, so $E(v_k) = 0$ and the equation becomes

$$(1 - J_k)E[\hat{x}_k] = K_k C E[x_k] \tag{8}$$

and since by (4) we have $E[\hat{x}_k - x_k] = 0$ it follows that

$$J_k = 1 - CK_k \tag{9}$$

Substituting from this into (3) gives finally the recursive *Kalman filter equation*, embodying a combination of predicted and observed values:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) \tag{10}$$

The term K_k is the *Kalman gain*. This now needs to be determined in terms of measurable and observable values. From (10), $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1})$, and subtracting x_k from both sides gives

$$\begin{aligned}\hat{x}_{k|k} - x_k &= (error)\\ &= \hat{x}_{k|k-1} + K_k(y_k - C\hat{x}_{k|k-1}) - x_k\\ &= (1 - K_k C)\hat{x}_{k|k-1} - x_k + Kv_k\end{aligned}$$

Squaring both sides and taking expectations:

$$\begin{aligned}E[x_{k|k} - x_k]^2 &= (mean\ square\ error)\\ &= E\left[(1 - K_k C)(\hat{x}_{k|k-1} - x_k) + Kv_k\right]^2\end{aligned} \tag{12}$$

Since noise sequences are assumed to be uncorrelated, the expectation of cross products involving v_k is zero. If the following relations are defined:

$$E[v_k.v_k] = \sigma_v^2, \quad E[\hat{x}_{k|k} - x_k]^2 = P_{k|k}, \quad E[\hat{x}_{k|k-1} - x_k]^2 = P_{k|k-1}$$

where σ_v^2 is the observation noise variance. Equation (12) then reduces to

$$P_{k|k} = (1 - K_k C)^2 P_{k|k-1} + K_k^2 s_v^2$$

which now has to be minimised with respect to K_k in order to arrive at the *mmse* requirement. The simplest procedure in this particular case is to expand and re-order the squared terms and minimise by inspection.

$$P_{k|k} = P_{k|k-1}\left(1 - 2K_kC + K_k^2C^2\right) + K_k^2\sigma_v^2 = K_k^2\left(C^2P_{k|k-1} + \sigma_v^2\right) - 2K_kCP_{k|k-1} + P_{k|k-1}$$

$$= \left(C^2P_{k|k-1} + \sigma_v^2\right)\left(K_k^2 - \frac{2CK_kP_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2}\right) + P_{k|k-1}$$

$$= \left(C^2P_{k|k-1} + \sigma_v^2\right)\left(K_k^2 - \frac{2CK_kP_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2} + \frac{C^2P_{k|k-1}^2}{\left(C^2P_{k|k-1} + \sigma_v^2\right)^2}\right) + P_{k|k-1} - \frac{C^2P_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2}$$

$$= \left(C^2P_{k|k-1} + \sigma_v^2\right)\left(K_k - \frac{CP_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2}\right)^2 + P_{k|k-1} - \frac{C^2(P_{k|k-1})^2}{C^2P_{k|k-1} + \sigma_v^2}$$

Clearly, $P_{k|k}$ will be minimum when the squared term vanishes, that is when

$$K_k = \frac{CP_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2} \tag{13}$$

and the corresponding minimum value is

$$P_{k|k} = P_{k|k-1} - \frac{C^2(P_{k|k-1})^2}{C^2P_{k|k-1} + \sigma_v^2} \tag{14}$$

$$= \frac{\sigma_v^2P_{k|k-1}}{C^2P_{k|k-1} + \sigma_v^2} \tag{15}$$

Substituting from (13) into (14) enables $P_{k|k}$ to be expressed in terms of the best current expectation based on previous data multiplied by a linear term involving the Kalman gain K_k.

$$P_{k|k} = (1 - CK_k)P_{k|k-1} \tag{16}$$

From the process equation it is evident that for a given control input Bu_k the state change can be represented as in equation (1), which includes a noise term, or as $\hat{x}_{k+1} = A\hat{x}_k + Bu_k$. Elimination of the control term gives a relation between the estimated states and the actual states:

$$\hat{x}_{k+1} - x_{k+1} = A(\hat{x}_k - x_k) - w_k$$

By squaring and averaging,

$$E[\hat{x}_{k+1} - x_{k+1}]^2 = A^2 E[\hat{x}_k - x_k]^2 - 2E[w_k . w_k]$$

which reduces to,

$$P_{k+1|k} = A^2 P_{k|k} + \sigma_w^2 \tag{17}$$

where σ_w^2 is the process noise variance.

B.2 Filtering out observation noise

Although a system will in general be subject both to process and observation noise, it is possible to apply Kalman filtering to observed (noisy) data without necessarily knowing the process function since the first step in the filtering cycle, the calculation of the Kalman gain, is a function only of the expectation *mmse* based on previous data and the variance of the observation noise. The steps that would need to be taken to estimate the system state from successive observations involve cycling though the following sequence:

1. calculate the Kalman gain, using $P_{k|k-1}$ (equation 13)
2. input a new observation y_k and update the estimate (equation 10)
3. update the current *mmse* $P_{k|k}$ (equation 16)
4. derive $P_{k+1|k}$, using equation 17, assuming if necessary zero process noise $\left(\sigma_w^2 = 0\right)$ and set this value as $P_{k|k-1}$ for the next observation
5. repeat the sequence for the next observation.

To start the sequence, initial values have to be chosen for the mean square error, $P_{k|k-1} = P_{1|0}$, and for an initial estimate, $\hat{x}_{k|k-1} = \hat{x}_{1|0}$. Values for the noise variances, σ_v^2 and σ_w^2, are assumed to be known.

B.3 Example of Kalman filtering

The following example illustrates the effect of Kalman filtering on a set of "noisy" observations produced by adding successive random values varying between – 0.1 and + 0.1 to the Bessel function $| J_1(k) |_{0 \le k \le 256}$. The

measured variance of the noise input, σ_v^2 , was 0.003. The degree of smoothing can be influenced by adjusting the values of the scale factors *A* and *C*. Too much smoothing has the effect of reducing the bandwidth of time-varying functions and needs to be used with discretion. Figure 8 illustrates the effect of using the measured value of σ_v^2 (0.003) and with *A* and *C* set to 1.15 and 1.1 respectively; reducing the value of *A* has the effect of "slugging" the filter response. The only other input is a value for the initial estimate of state, $\hat{x}_{1|0}$, which can be arbitrary, here taken as 0.1. The way the procedure of the previous section is used to filter the noisy Bessel function is shown in Table 4.

Observation	*P(k|k-1)*	*K(k)*	*Estimate*
-0.0120915	0.0125645	0.759266	0.00730008
0.115737	0.00273853	0.477125	0.0586899
0.0528222	0.0017209	0.372468	0.0543184
0.144218	0.00134343	0.31948	0.0813042
...	...	...	...
-0.0334884	0.000799587	0.221688	0.0214364
0.101983	0.000799587	0.221688	0.0388174

Table 4: Conversion of *Observation* values to *Estimates*

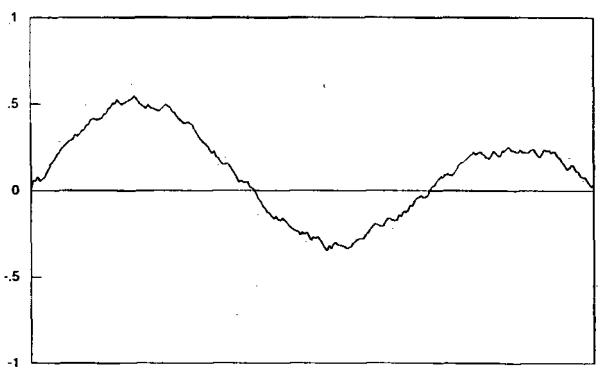


Figure 9: Effect of Kalman Filtering

Starting with the initial state estimate, $\hat{x}_{1|0}$, the entry value of the *mmse*, may be calculated as:

$$P_{110} = \{0.1 - (-0.0120915)\}^2 = 0.0125645$$

This enables K_1, the Kalman gain, to be derived. Hence, from Equation 13 using a scaling factor $C = 1.1$ and a measured value of 0.003 for σ_v^2:

$$K_1 = CP_{110} / \left(C^2 P_{110} + \sigma_v^2\right) = 1.1 \times 0.0125645 / \left(1.1^2 \times 0.0125645 + 0.003\right) = 0.759266$$

Using equations 16 and 17 and taking this value (K_1) of the Kalman gain, together with $k = 1$, $A = 1.15$ and $\sigma_v^2 = 0$, a value for P_{211} may be calculated:

$$P_{211} = 1.15^2 \times 0.0125645 \times (1 - 1.1 \times 0.759266) = 0.00273853$$

Using equation 10 and $k = 1$, a value for $\hat{x}_{111}$ may be obtained:

$$\hat{x}_{111} = \hat{x}_{110} + K_1\left(y_1 - C\hat{x}_{110}\right) = 0.1 + 0.759266 \times (-0.0120915 - 0.1 \times 1.1) = 0.00730008$$

The values of $\hat{x}_{111}$, P_{211} may now be used in the next cycle with $k = 2$ to complete the second row of the table and so the process continues.

After several iterations of the Kalman cycle, the filter will start to approach a steady state with both the Kalman gain and the transferred expectation tending towards constant values—as shown by the elements at the bottom of Table 4. In this steady state, it can be seen that the *filter equation* reduces to:

$$\hat{x}_{k+1} = 0.221688 y_k + 0.7561432 \hat{x}_k$$

This is Equation 3 with constant values for J and K. Clearly the greater the noise in the system the more significantly the prediction component will feature in the state estimate. Conversely, by artificially increasing the effect of noise, the apparent smoothing can also be increased—but at the expense of impaired filter response. The filter model is easily implemented on a spreadsheet and, in a high level declarative language such as Prolog, the basic recursive cycle can be coded in five lines.

B.4 Filtering Forex Data

Foreign exchange (Forex) data is published daily by various organisations. Some of these publish historical Forex data that includes not only mean bid/ask rates going back to January 1990 [Olsen,

1996] but also the daily maximum, minimum and 75% interfractile values. These daily fluctuations can be regarded as the noise in the system. There is furthermore enough information in these figures for an estimate of variance to be made—if a few assumptions are made.

Although the distribution of bid rates recorded in a working day, normally several thousand, is not published, the assumption will be made that it is random within the published interfractile limits. The variance of any random sequence within these limits will therefore be assumed to approximate the variance of the published Forex rates. Since, however, neural networks normally accept input unit values in the range between zero and unity, the variance needs to be calculated on the daily fluctuations of the normalised exchange rates. Figure 9 illustrates the result of filtering the normalised DM to US/$ exchange rate over 66 working days using a variance of $\sigma_v^2 = 0.00263$. This is probably a pessimistic estimate of the noise, as it was calculated on noise based on the maximum 75% interfractile values encountered during the sampling period, at a time when there was a continuously rising overall trend. The same scale factors were used as in the previous example.

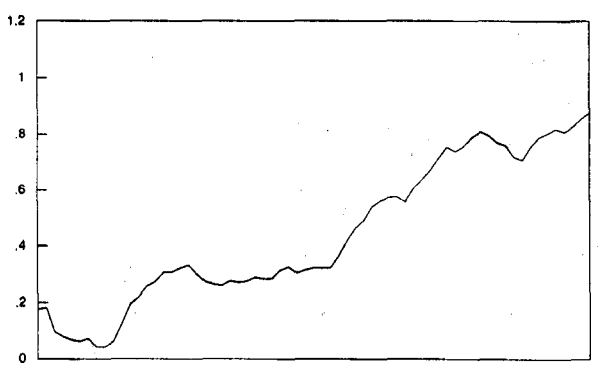


Figure 10: Effect of filtering DM-US$ rate

Bibliography

MEDITCH, J.S., "Stochastic Optimal Linear Estimation and Control," McGraw-Hill, 1969.

MEHTA, M, "Foreign Exchange Markets, in Neural Networks in the Capital Markets," Apostolos-Paul Refenes ed. Wiley, 1995.

NEURONET WEB SITE,
http://www.neuronet.ph.kcl.ac.uk/neuronet/software.html; the co-ordinating node for NeuroNet maintained by King's College, University of London.

OLSEN ASSOCIATES WEB SITE,
http://www.olsen.ch/cgi-bin/exmenu, Olsen Associates,
Seefeldtstrasse 233, 8008, Zurich, Switzerland.

RUMELHART, D.E., HINTON, G.E., and WILLIAMS, R.J., "Learning Internal Representations by Error Propagation," Parallel Distributed Processing: Vol 1, edited by Rumelhart et al, MIT Press, 1986.

Biography

Owen Evans joined the Advance Research and Development laboratory of ICT Engineering (an ICL predecessor) in 1963 from the oil industry. The research facility has continued in unbroken line to its present guise of the Research and Advanced Technology centre of ICL Group HQ. During his time in the research centre which has spanned virtually the entire evolution of the computer industry, he has worked on computer architecture, memory and microprogram design, system performance evaluation, compiler design and logic languages. In the course of his career at ICL he has managed various projects supported by the Advanced Computer Technology Project, the Alvey programme, and the EC Esprit and Fourth Framework initiatives in the areas of high-level language emulators, text databases, human-computer interaction and constraint logic programming. His current interests are in the areas of constraint logic programming, data mining and neural networks.

Mr. Evans graduated in Engineering from Cambridge in 1959.

Helping Retailers Generate Customer Relationships

Uri Baran

ICL Retail, Willoughby Road, Bracknell, Berkshire

Abstract

People are fickle and motivated by conceptually simple things such as price, performance and service. Most retailers would rather differentiate their offering through service than price on the basis that people don't mind paying for good service and they tend to come back for more. Service has value and good value is more profound than low cost and it tends to result in a more profitable business. The problem is that good service is bespoke service and more difficult to define and much more difficult to achieve. The application of new technologies in this area is in the process of revolutionising the opportunities for retailers.

1. The Problem

Business theory has for some time now been concerned with the achievement of competitive advantage.

Two approaches are particularly pertinent. The first ascribes competitive advantage to how value is produced for customers. The second concerns the relationship of technology to competitive advantage.

The first approach, particularly that expounded by Treacy and Wiersema [Treacy and Wiersema, 1995], defines the means to produce value in the following terms:

1: Operational Excellence
2: Product Leadership
3: Customer Intimacy.

Operational excellence is the strategy adopted by companies seeking to lead in price and convenience for the production and delivery of their products and services. They improve processes, reduce overheads, cut transaction costs in a relentless fight against customer inconvenience and cost.

Product leaders aim to produce continually new and better products and services. They are creative and react quickly; they set the pace of technology and change in their industry.

Customer Intimacy is the value discipline that aims to treat the customer as an individual. This is achieved by tailoring products and services to meet requirements of individuals and customer groups. The customer is viewed as having a lifetime value as opposed to the value of a single transaction. The key to the customer intimate approach is the application of knowledge to customer interactions and continuing customer contact. The management of this customer knowledge is fundamental.

Regarding the second approach (i.e. that relating technology to competitive advantage), it is widely accepted [Baran, 1994] that technology is one of the main drivers for change and competition and is a major basis for adding value. Porter [1985] sees technology's role in the pursuit of competitive advantage through its ability to help firms achieve low cost and/or differentiation where low cost and/or differentiation are *the* routes to competitive advantage.

From a traditional retailer's perspective, the means to achieve competitive advantage can be viewed as mapping on to the three value disciplines described earlier. Product leadership is becoming increasingly difficult for retailers, as most do not themselves manufacture their products, and thus have to rely on obtaining their supplies from producers who have other channels to market: so the consumer can invariably obtain similar products elsewhere.

The retail equivalent of operational excellence can be viewed as the supply chain and store operations, the means to obtain the right products and to make them available in the right place and at the right time for the customer. This area has been a major battleground for retailers over the last decade with costs being squeezed out, replenishment times reduced and automated warehousing and supply systems developed. Although there is scope for further improvements in this area, leading retailers are now turning to the third discipline that of customer intimacy as the strategic source of competitive advantage.

For a retailer, customer intimacy is about recognising that the market consists of many groups of customers with each group being defined by having similar characteristics in terms of its requirements. In the past

it has been the policy to produce a standard set of products and services for all customers, yet all customers are not equal and not the same.

When stores were small and large chains non-existent, it was possible for each store keeper to recognise each customer and serve them individually on the basis of a relationship of sorts (another way of describing intimacy). Yet in today's era of mass production and mass marketing, this relationship has been lost and with it the customer's loyalty to that retailer. So how do retailers set about tailoring their product and service offerings to ever finer customer segments even down to the individual? How can a communication between individual customers and the retailer be re-engaged to support such a relationship?

2. The Issue

These problems are essentially 'people' issues in the rather grey areas of relationships, communication, loyalty and service. They stem directly from business strategy and are described in distinctly non-engineering terms. So how does one set about providing a solution to them? How does one design a solution with an output called a customer relationship?

Engineering is about solving technical problems and the practical use of science and technology; yet real world business oriented problems, such as those described above, do not relate well to the engineering discipline, which struggles with the real world grey that resides between the black and white world of computing.

2.1 Approaches to assist in its solution

One useful scientific problem-solving principle is to find a process, or a set of steps, that take one from where one is to where one wants to be and several clues existed that would help define a starting point, as well as a means to progress:

1: Customer Loyalty a structured approach

2: Knowledge management

3: Gartner group Models of Information System Development practice.

2.1.1 Customer Loyalty a structured approach

This has been a high profile topic in Retail for many years and there have been several different definitions of it: for example, Liz

Mandeville's [Mandeville, 1994]. "Systems which make use of modern retail systems and other new technologies to persuade individual shoppers to prefer one retail group to its competitors through spending related rewards, better marketing or both."

Although loyalty and intimacy may seem the same, loyalty, in fact, was, and is, largely a collection of schemes that retailers produced that involved giving their customers an identification card and rewarding them with discounts, or other benefits, in an attempt to bribe them to keep shopping with you. Intimacy is a concept more concerned with the fundamentals of generating loyalty.

It was apparent about two years ago with a plethora of these schemes available from petrol stations to supermarkets, that if such a system were to be developed, it should be done from a deep understanding of the basic principles involved. The main problem was that there was very little literature that did anything other than comment on existing schemes. As a consequence, a small joint venture was set up between ICL Retail Systems and ICL High Performance Systems (HPS) aimed at using a new methodology being developed by Graham Pratten of (at that time) HPS and Professor Peter Henderson of Southampton University. This methodology is called POSD (Process Oriented System Design). At that time, it was in its infancy and needed some business problems to attack to ensure that it was up to the job. My task was to attempt to use it to provide a structured approach to loyalty system design.

After several months work reading literature, identifying and analysing existing loyalty schemes, a generic loyalty model was constructed, see Figure 1.

Using this and the work that had gone into its development, it became possible to categorise and characterise all loyalty schemes using a common notation, as well as design schemes from first principles based not only on requirements but also on a deep and clear understanding of the options and reasons for choice between them.

This work provided some key insight into the topic of loyalty and the design of operational loyalty schemes. One of the fundamental insights it provided was that, if done properly, loyalty schemes consisted of two parts. The first part was the operational part consisting of the registering (enrolling) of customers, communication of inducements and redemptions. The second part was called *Targeting*

and it consisted of using the operational part in a tailored (targeted) manner to communicate with individual customers. It became clear that most of the existing loyalty schemes were in fact just operational schemes with no targeting mechanism and there was little or no understanding of the means to target.

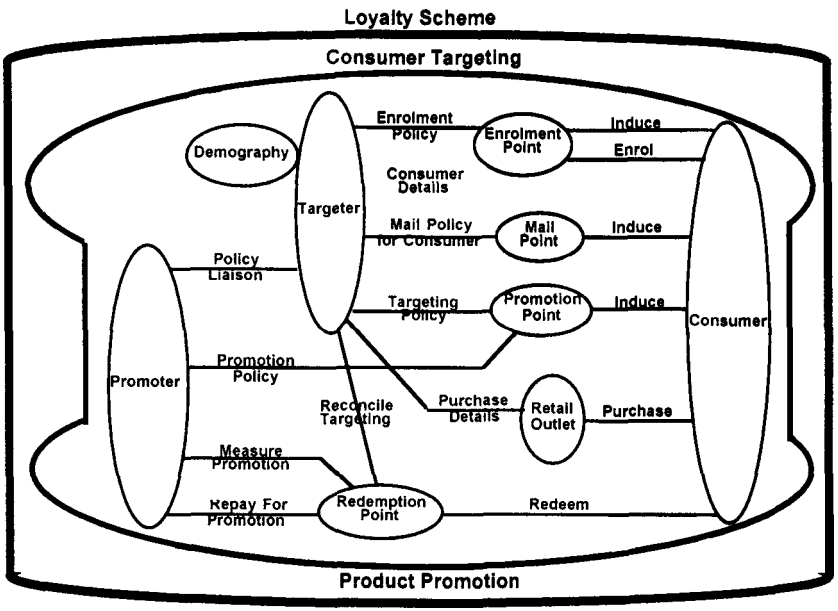


Figure 1: A generic loyalty model

2.1.2 Knowledge Management

A few years ago, I attended a series of lectures on the Management of Information Systems and one of the slides (originally from British Petroleum, I believe) appeared to have great relevance to this problem.

It can be represented as follows:

$$\text{Data} \xrightarrow{+\text{ Structure}} \text{Information} \xrightarrow{+\text{ Context}} \text{Knowledge} \xrightarrow{+\text{ Experience}} \text{Wisdom}$$

Data is information without structure. The structuring process adds value to the data and transforms it into information. The extraction, formatting, aggregation and transformation of financial facts or retail data turns them into information

Knowledge is the range of information and the accumulation of it. The context defines the boundaries and applicability of the information.

Wisdom comes from experience; to know alone is not enough to be wise, experience is about living through events and adds its own value to the knowledge.

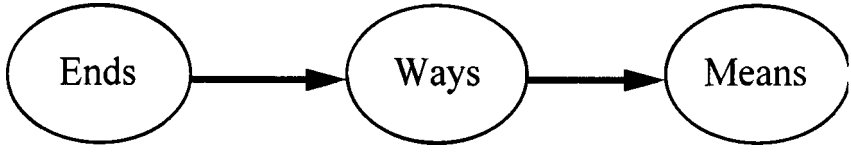
This process can be likened to the human process of ageing and wisdom is usually related to age.

This seemed to provide a structure for a process that starts with data and ends with its wise usage.

2.1.3 Gartner group: Models of Information System Development practice

The third clue stems from a Gartner Group article [Gartner Group, 1995].

Gartner Group describe standard Information Services development practices in terms of the model below.



This model describes the planning and implementation route through 'Ends' such as goals and objectives driving 'Ways' such as programs or policies which are supported through 'Means' including technology, information, people and processes. It describes a process that relies upon analysis, observation, cause and effect and quantification. It relies upon the problem staying fairly static to achieve its desired results while the process is taking its course.

The problem with this approach is that it is best suited to environments where:

1: It can be applied to business situations with objectives for increased speed, efficiency, accuracy, cost and effectiveness.

2: The outcomes are known and quantifiable

3: The process is stable and structured

4: Improvement is incremental.

It is less suited to environments where:

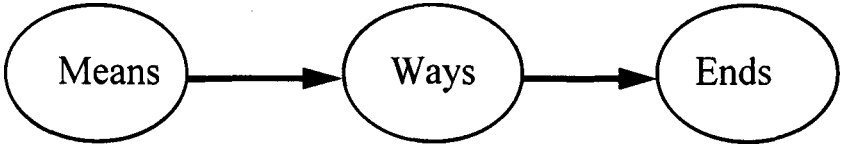
1: The business environment is uncertain, changeable and complex

2: Strategies are forward-leaning, opportunity-orientated and focused on decision making

3: The emphasis is on strategic issues such as shaping new markets

4: There is a premium on rapidly delivering new products or deploying a pre-emptive response to competitive threats.

Retailers have both structured and unstructured business environments. The structured environments, such as the supply chain with efficiency and cost objectives and well defined processes and tasks, are well suited to the EWM model but the unstructured environments, such as micro-marketing and mass-personalisation with decision based subjective operations, very complex processes, fast changing functions and large amounts of data, require a different approach.



By reversing the model so that it becomes a Means – Ways – Ends process, it starts with suspected sources of advantage and uses strategic means to create demonstrably new or better ways to satisfy prevailing ends or create new ones. This turns traditional thinking upside down with IT usage driving business objectives rather than the norm of the opposite.

The starting point (or suspected source of advantage) is people, process, information and technology. In real terms, this means the application of strategic Data Discovery techniques to transform large data sets to discover knowledge leading to new sources of competitive advantage.

3. A solution - The Corema Data Discovery and Exploitation Process

Putting the three clues together suggested the use of retail customer data as the starting point, a loyalty type environment, a learning model to take advantage of experience and Data Mining tools as a possible means for advantage with unclearly defined ends.

ICL Retail embarked on two parallel actions.

3.1 Corema – a Generic Operational Loyalty Scheme

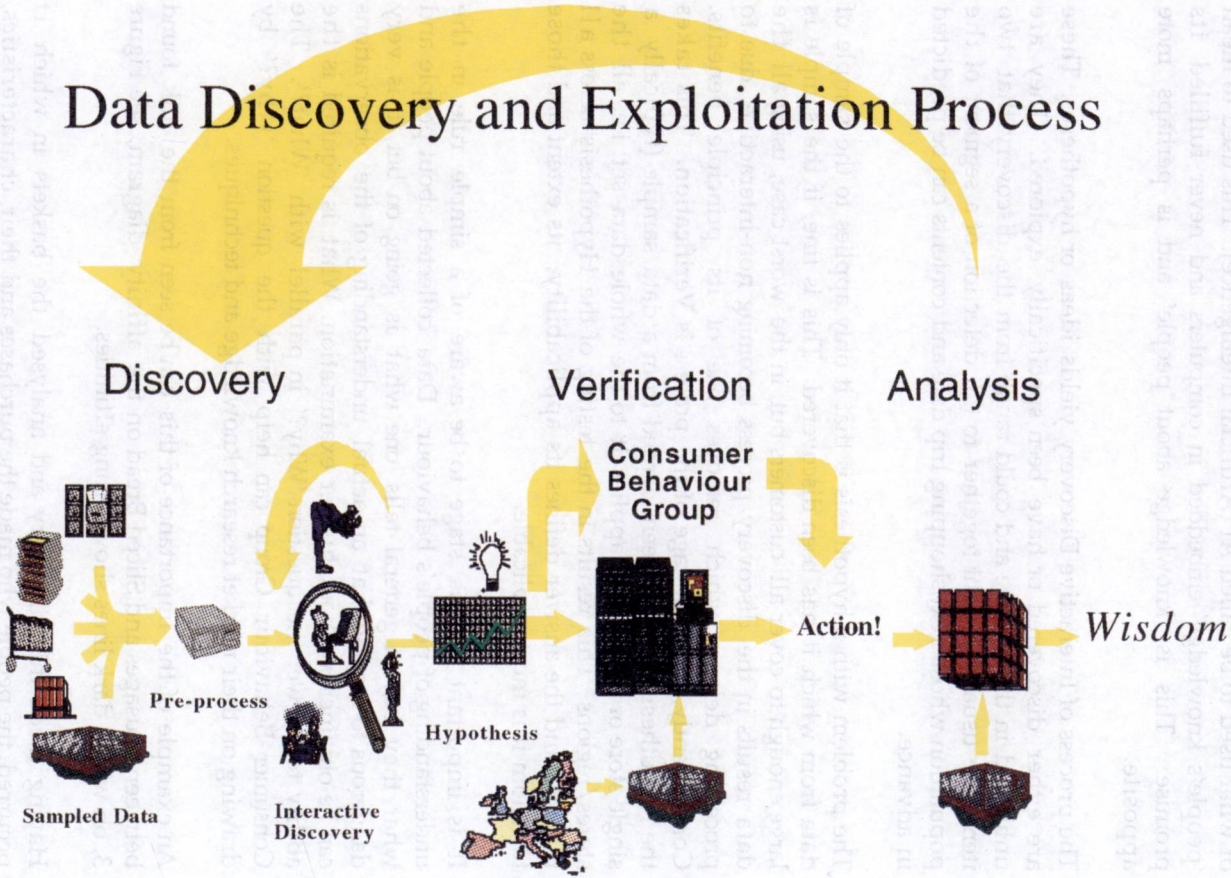
One of these was the development of an Operational Loyalty Scheme for retailers. This was to be the starting point for the development of customer relationships. Without a means to motivate, enrol, and reward customers, so that they could be recognised, there was no way of collecting detailed information about them. Without the detailed information, there could be no Data Discovery. Using Rapid Application Development techniques (specifically, Huron Objectstar), an operational loyalty scheme was developed called Consumer Relationship Marketing or COREMA based on some of the principles outlined in the POSD work on loyalty scheme modelling.

The second of the actions was to develop skills in the use of Data Discovery tools on real high volume retail data. What emerged was the following process model for Data Discovery and exploitation.

3.2 The Data Discovery and Exploitation Process

This model, shown on page 311, starts with the data, it can be internal to the organisation or external and in any form. The slide from the lectures on Managing Knowledge suggested the need for structuring data. This is akin to pre-processing the data and hence suggests the need for a pre-processing stage. Raw data is useless. This data collection and structuring process is today known as the Data Warehouse, if it is taken to its logical extreme. In fact, the diagram shows a representation of a Data Warehouse with the caveat of it being used for sampled data. It was found that for worst case conditions, i.e. large supermarket chains with Terabytes of data, Interactive Data Discovery cannot be applied to all the data and a sample must be taken. Because stores have identities and characteristics arising from their unique local demography and competitive situation, sampling at the store or region level is both technically possible and sensible.

The next stage is Interactive Discovery. This is the act of applying data discovery tools and techniques in an appropriate flexible and powerful environment on the pre-processed data. The process must be able to be both human and machine led and, as a consequence, must take advantage of any suitable technology that aids the tasks of discovering information and knowledge from the data. These technologies range across neural networks, rule induction, genetic algorithms, advanced data visualisation techniques and standard statistical techniques.


Data Discovery and Exploitation Process
Discovery
Verification
Analysis
Consumer Behaviour Group
Pre-process
Hypothesis
Action!
Wisdom
Sampled Data
Interactive Discovery

The key is to apply technologies that can provide answers to questions about the data that were not known a priori. Perhaps this is a variant of what used to be called AI (Artificial Intelligence). This used to mean 'people's knowledge' embedded in computers and never fulfilled its promise. This is 'knowledge about people' and is perhaps more apposite.

The process of Interactive Discovery yields ideas or hypotheses. These are either discovered or have been specifically explored. They are unlimited in their scope and could range from the discovery that two items are usually bought together to the detection of a segment of the population whose next shopping trip date and contents can be predicted in advance.

The problem with a hypothesis is that it only applies to the sample of data from which it has been discovered. This is fine, if the sample is large enough to cover all customers but, in the worst case, using all the data results in the discovery process becoming non-interactive, due to processing delays, which removes one of its principle benefits. Consequently, the next stage of the process is Verification. This takes the Hypothesis that has been learned from a data sample (typically a single store or region) and applies it to the whole data set, i.e. all the stores/regions. This results in the testing of the Hypothesis across all the data and the answer defines its applicability, its extent and those to whom it is most applicable.

It is important at this stage to be aware of a simple rule in the understanding of people's behaviour. Data collected about people and what they buy in general tells one what is going on but it is very dangerous to assume that an actual understanding of the observations can be obtained without further examination. What is required is the ability to answer the question 'Why?' in parallel with 'What?'. The Consumer Behaviour Group can help with the question 'Why?' by drawing on their market research knowledge and techniques.

An example of the importance of this can be seen from the link found between Sausages and Sliced Bread on the affinity diagram, see Figure 3, on which dark lines show strong affinities.

Having made this discovery and analysed the baskets in which it occurred, the people who made the purchases and their characteristics, at least gives one the opportunity to ask them why, before diving head first into sausage sandwich promotion.

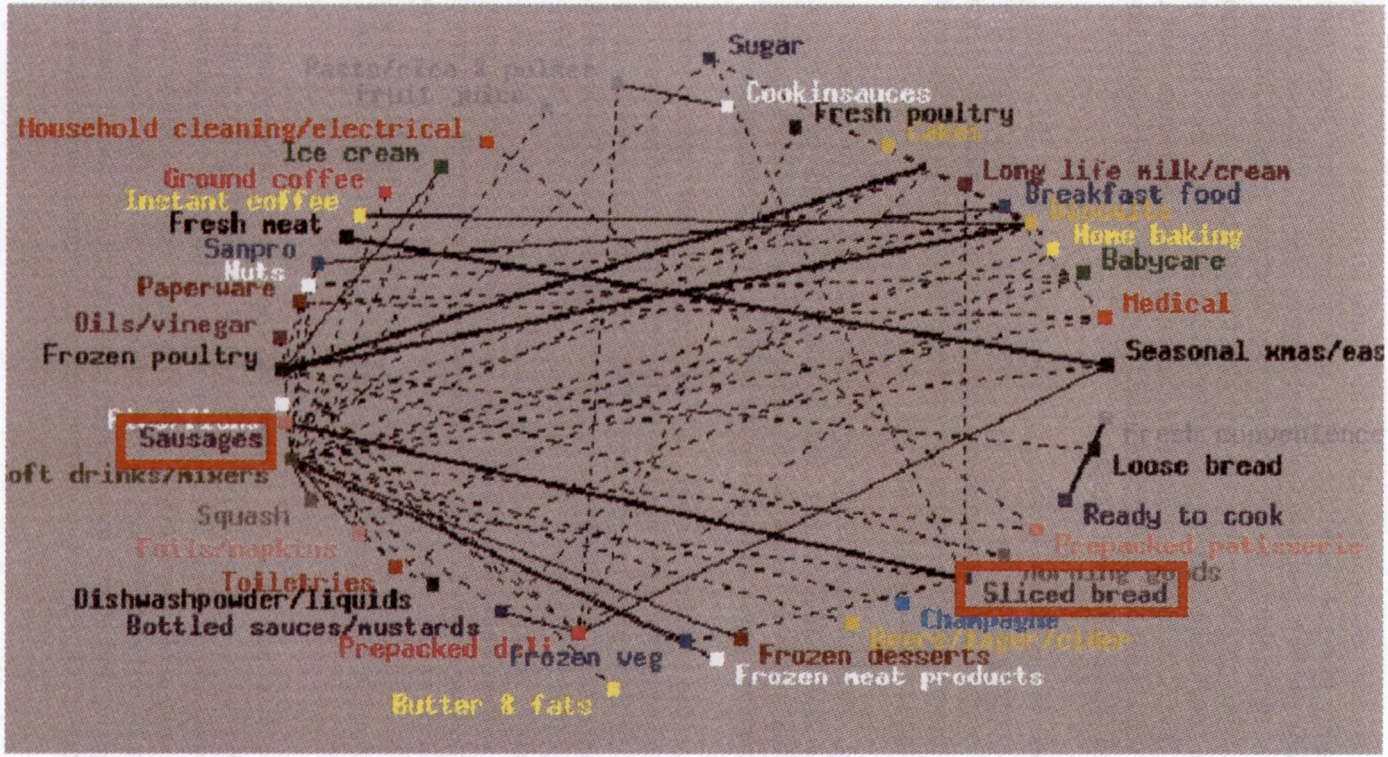


Figure 3: Inter-Group Affinity Map

What is now an industry cliché in terms of linked items is that supposedly found at Walmart, when it was noticed that there was a link between beer and nappies [Derbyshire, 1996]. Without further analysis determining that it was evening shopping trips by fathers of babies, who came in for nappies and also picked up a six-pack, various strange conclusions may have resulted. In fact when this link was found, it was decided to sell beer next to nappies and beer sales shot up. If the why question hadn't been asked, who knows what might have resulted—beer flavoured nappies?

The next stage of the process is called Action. It is the means of communicating with the customer. In our case, it is the operational loyalty scheme described earlier. This provides the means to identify and communicate with customers via any means be it in the store, via direct mail or through the Internet.

Having communicated with them, the next step is to measure and analyse the results of this and to feed the results back into the Discovery stages to create the learning cycle and process suggested by the experience required to add to the knowledge to create wisdom.

Beyond the 'Action' stage is Analysis. When some form of communication has been made with the customer, e.g. direct mail, the results of this must be analysed. They must be compared with the expected results and deemed a success or a failure accordingly but most important is the fact that the exercise itself has provided a learning experience, which can be fed back into the cycle so that retailers can then use their data more wisely. The analysis stage is represented by a cube made up of other cubes. This is because multidimensional database analytical methods (OLAP—On-line Analytical Processing) are best suited to cutting and dicing the data by pre-defined dimensions in order to answer the retrospective analysis requirements of this stage of the process.

Corema has now become a name for the whole process and has developed into a set of deliverables aimed at assisting the retailer to implement the process. It consists of product offerings with packaged and configurable applications and managed services, allowing the retailer to test out the effectiveness of parts or all of the process in an inexpensive and staged manner and have consultancy for all aspects of the process. One of its key differentiators is its end-to-end nature with

the focus on integrating and automating the process and the embedded technologies.

4. What about Customer Relationships?

Providing retailers with a means to acquire information about their customer behaviour, on which they can act via a range of different communication media, does not automatically generate customer relationships. Even making the whole process automated, integrated and quick will not do this.

But such a facility is a support process for retailers who are serious about embarking on a customer based strategy. If they approach the problem from a basis of attempting to serve their customers better and profitably, as well as trying to recognise segments of their customers base, which can be better served by treating them separately from the rest, then they have started on the right approach.

Once it is realised that a service that the customer will value begins when communications (be it offers or just plain information) start to become more relevant to the people who receive them. What results is loyalty. There is more to this insofar as the use of Data Discovery and Exploitation techniques, aimed at the merchandising side of a retailer's business, i.e. the supply side of what is made available to customers and how, also has value in inducing loyalty, but initially we are focusing on the customer element.

5. Corema's Relationship to Data Warehousing

Data Warehousing is a topic currently high on the agenda of most IT managers [Derbyshire, 1996] and customer relationships is a topic high on the agenda of most Managing Directors. Data Warehousing is one of the means to support customer relationships. So what is a Data Warehouse?

A description is given [Kelly, 1996] in terms of six key characteristics that differentiate it from other database systems in the enterprise:

1: *Separate* from the operational systems in the enterprise and populated by data from these systems

2: *Available* entirely for the task of making data available to be interrogated by business users

3: *Integrated* on the basis of a standard enterprise model

4: *Timestamped* and associated with defined periods of time

5: *Subject orientated,* most usually on the basis of customer

6: *Accessible* to users who have limited knowledge of computer systems or data structures.

Another way of looking at it is to consider Data Warehousing as a means of enabling an organisation to extend its use of data from operational to strategic.

It rapidly becomes clear that the Data Warehouse and all its associated concepts underpin the concept of Consumer Relationship Marketing. In fact, without a Data Warehouse in mind, an organisation may well not be taking itself seriously when aiming for customer relationships.

ICL's Business Warehouse Solutions and High Performance Systems are heavily involved with the Corema initiative. Similar retail based initiatives going on in the US and it is becoming apparent that the retail market is leading in the development of such systems, based on both Relationship Marketing and Merchandising applications.

6. The Future

The future is really about making it work for organisations and making it do so in a manner that can be quantified. I am aware of no one who has fully implemented a process similar to that described earlier in a retail environment. What this means is that there is little experience and expertise available in the market and little or no academic theory to back it up, so some rather painful pioneering is going on.

Today, active attempts to generate customer relationships are the domain of very few organisations. As has been shown, it is a non-trivial task involving large investments and long term backing. Most organisations are turning to it slowly as they are usually held back by existing commitments and infrastructure. It seems as though it has become a priority for organisations who are suffering and are therefore attempting to use it to become more competitive or, alternatively, for those leading organisations determined to maintain their position.

From ICL's perspective, it is certainly a technology based initiative while not being a technology led one. For once technology has created an opportunity that does more than enable business strategy. Discovery is more than analysis and the opportunity created by the ability to use

technology to tell one what is in the data without knowing which specific question is the right one to ask means that it is impossible to specify what is going to emerge and what can be done with it. It can certainly be harnessed to support the generation of customer relationships but what are the implications of being able to predict who is going to come into a store, when they are going to come and what they are going to buy?

7. Conclusions

I believe the fundamental conclusion of this paper is that technology should be aimed at solving fundamental business problems and not abstracted from them. By the time that technologists usually get involved, the level of abstraction is so great that the business problem has been lost from the equation and all that is left is a technical problem that requires solution.

There are other conclusions, because technology is the greatest driver for change, technologists should be in a powerful position to drive change for the benefit of their companies, yet this rarely seems to happen. Is it because most engineers (in my experience) look down at the computer and not up at the business? Is it because people get fixated with a technology rather than technology per se? The result of this is a tendency to look for technological silver bullets rather than look for technology based solutions.

Finally, if one looks into what is going to make Corema work, it will be the creativity, intuition and imagination of the people driving it because it supports them and not replaces them.

References

BARAN, U, "What are the Opportunities for the Development of Sustainable Competitive Advantage through the Application of Technology within the Commercially Focused Environment of ICL Retail Systems," MBA Dissertation, Henley Management College, 1994.

DERBYSHIRE, M.H., "AnArchitecture for a Business Data Warehouse," *ICL Systems Journal*, Vol 11(1), May, 1996.

KELLY, S., "Data Warehousing – The Route to Mass Customisation," John Wiley & Sons, 1994.

LOVE, B., "The Strategic Data Mining Paradigm," The Gartner Group, 1995

MANDEVILLE, E., "RMDP report on Customer Loyalty," 1994

PORTER, M.E., "Competitive Advantage – Creating and Sustaining Superior Performance," The Free Press, 1985.

TREACY & WIERSEMA, "The Discipline of Market Leaders," 1995.

Biography

Uri Baran is currently 'Manager, Data Discovery' for the Precision Retailing group of ICL Retail Systems. He has responsibilities for the tools and technologies incorporated into the data analysis and exploitation part of Precision Retailing initiatives. He also provides consultancy services to retailers to help them exploit their data.

After obtaining a degree in Electrical and Electronic Engineering, Uri Baran worked as a hardware design engineer in diverse areas encompassing voice processing, video distribution and communications processing.

He joined ICL Retail Systems in July 1990 specialising in interconnection. Subsequent work has involved the development of a retail specific Structured Cabling scheme and various interconnection projects and consultancy. Whilst a project manager with responsibilities for ISDN/Multimedia technology exploitation and interconnection, he studied part time for the degree of MBA.

After obtaining his MBA, he helped establish and run ICL Retail's 'Cave' which was the centre for the development, integration and demonstration of the application of new technologies to the Retailing business. The centre also did much of the theoretical work that underpins ICL's approach to Customer Loyalty.

The Systems Engineering Excellence Model

Brian Chatters

ICL High Performance Systems, Manchester, UK

Abstract

The Systems Engineering Excellence Model (SEEM) provides a framework for the sharing and deployment of best practices in systems development. Best practice is defined as a set of generic processes based on Capability Maturity Models, Quality Standards, Development Life Cycle Models, and internal best practices currently in use within ICL High Performance Systems. SEEM is a Windows Help system, exploiting hypertext technology to link existing quality system documentation, best practice definitions and existing on-line control systems into an effective development route tool kit.

1. Introduction

In late 1994, the management team of the Software Systems development group of ICL High Performance Systems carried out an in-depth review of its business and operations. This review led to the following conclusion:

> *"Traditionally, the main business of Software Systems has been the design and development of operating systems and associated software for corporate servers (mainframes). The processes, tools and methods used to develop such products have evolved and matured to a level where they are well established and well understood by the engineers. However, if Software Systems is to grow and meet its business objectives, it must establish a more diverse product portfolio. Such diversification will require different ways of working to meet the increasing demands for lower costs, faster time to market, and improved quality. The old ways of working may no longer be entirely appropriate."*

One of the outputs from the management review was to define a set of Critical Success Factors (CSF), one of which was:

"To establish a framework for defining and developing core software engineering competencies, methods and products to achieve industry best practice."

The following objectives were defined to support this CSF:

- to promote the concepts of a "learning organisation" in a systems engineering context
- to establish an agreed set of core engineering processes which are essential for the development of any programme of work within ICL High Performance Systems
- to provide a method of evaluating the systems engineering processes and their deployment across ICL High Performance Systems against externally-defined world's best practice
- to establish mechanisms for defining, selecting and implementing improvements to the processes, tools, methods and technologies
- to provide a framework for researching and exploiting innovations in the science of systems engineering
- to establish an engineering information strategy, including mechanisms for the technology transfer of systems development innovations and for the sharing of local information and knowledge across the engineering community.

A working party was set up to develop a strategy to achieve these objectives. The use of the Capability Maturity Model (CMM) developed by the Software Engineering Institute (SEI) in the USA, was recommended to be used as the benchmark for industry best practice. The output from the working party was a documented software engineering improvement strategy, published in June 1995, which formed the basis of an implementation plan to establish a framework to achieve the objectives. This paper describes how the framework was developed.

Although a number of on-line systems exist, their main usage is to provide a document library and the use of hypertext is primarily to link documents (passive role). Some quality management tools are available but they mainly only provide checklists for standard quality management processes and documentation standards. They are designed mainly for use by quality engineers.

The Systems Engineering Excellence Model (SEEM) provides a practical tool to design appropriate development routes for implementation within ICL High Performance Systems projects. It is aimed at all project technical staff and it provides a knowledge base of best practice which is continuously capturing state of the art improvements. It is an evolution from current TickIT based documents so provides adequate support for quality system documentation required to maintain ISO 9001 conformance. Once a developnent route has been established in SEEM, it can be used proactively to support the operations within a development project.

2. Overview of the Systems Engineering Improvement Strategy

Although initially defined for software engineering, the strategy was developed in such a way that it could easily be adapted and applied to other engineering disciplines.

The implementation of best practice requires that a definition exists of what it is and that mechanisms are in place to enable the deployment of the practice within the development projects. Traditionally, such implementation has been achieved through the ICL corporate policy of conformance to quality system standards (ISO 9001). The main limitations of this approach are that the standard encourages a minimum level of practice ("just enough to pass audits"), and it does not help to identify ways in which the requirements can be satisfied. Local quality systems, as defined within ICL High Performance Systems, tend to be prescriptive and constrain the ability to be flexible in their application. The Systems Engineering Improvement Strategy is designed to address these issues. Figure 1 illustrates the key elements of the strategy.

Central to the strategy is the Systems Engineering Excellence Model – SEEM – which provides the framework in which best practice can be effectively deployed. At the implementation level (the 'operational system' on the diagram), deployment manifests itself as a quality plan which defines tasks to be carried out, the tools and methods to be used to carry out the tasks, the staff who will carry out the tasks (authorities), and the sequence in which they will be done (based on a Development Life Cycle Model). This definition is often referred to as the 'development route.'

Reuse of existing development routes is normal practice (this is typically what the local quality systems define) and these

development routes are captured within the model. However, SEEM also includes a 'generic process' which is based on both ISO standards and best practice models. The strategy provides flexibility in the choice of best practice models allowing both hybrids, based on models such as CMM and SPICE, and state of art practices to be captured.

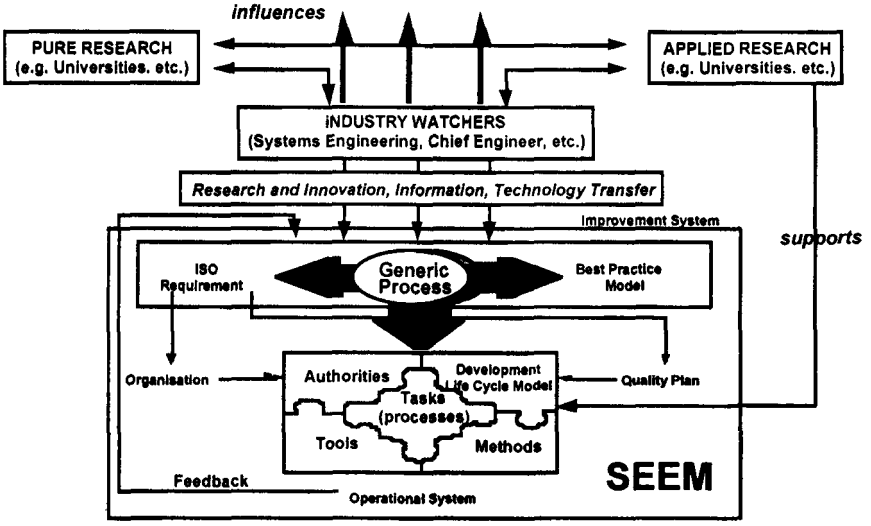


Figure 1: The Systems Engineering Improvement Strategy

The definition of best practice will evolve through four stages:

1. a definition of the requirements for minimum conformance to corporate and divisional policies, and ISO 9001 and ISO 9000-3 (TickIT) quality system standards
2. the capture of internal practices as defined by the processes, methods and tools used within ICL High Performance Systems (evolved from existing local quality systems)
3. a definition of world's best practice as defined through internationally recognised standards and models
4. the capture of new processes, tools and methods based on the exploitation of leading research ideas and internal innovations in the field of software and systems engineering.

Progress through the levels of best practice and improvements to the generic process captured in SEEM is planned to be achieved by using Software Process Assessment (SPA) techniques as defined by CMM and

SPICE. Specific development routes captured within SEEM will be compared with the generic model and improvements made to them, as appropriate.

3. Initial Development of SEEM

The initial requirement for the implementation of SEEM was to evolve from the current approach to the management of development routes, based on ISO conformance, into a more flexible approach to address the issues raised by the management review in 1994.

The Software Systems quality manual was paper based, consisting of a top level requirements document supported by more detailed procedures, forms and checklists, as appropriate. In the main, most of the underlying documents are only applicable to VME developments. The documents are stored on a network server as a set of Word for WINDOWS files which can be printed or downloaded. In addition to these documents, a number of Word for WINDOWS templates exist which facilitate the implementation of standards. These templates are supported by an on-line Help system which describes how they are used. Some of the templates have a degree of automation provided by Word macros. Some users have chosen to produce local versions to address their specific requirements.

The top level requirements document within the Software Systems Quality System documentation provides referenced links to the underlying procedures, forms and checklists for VME but experience shows that such links are little used. Most engineers are familiar with the underlying documents so they tend to address them directly. The prime users of the top level document are those responsible for quality assurance (e.g. quality managers and auditors).

The aims of the initial implementation plan were:

- to establish a framework within the first stage of best practice definition which was not VME specific
- to allow any software development team to establish an appropriate development route for their local operations which is fully conformant with the ISO 9001 quality system requirements
- to provide better links between the top level documents and the underlying procedures, forms, checklists and templates.

A decision was made to develop the framework (SEEM) as an on-line Help system. The rationale for this decision was that the standard platform within Software Systems was based around Microsoft Windows and the existing Help system for the Word templates had, in general, been well received. In addition, the HDK Product ('Help Development Kit') available from Virtual Media Technology was in use by the Customer Information group and had demonstrated the ability to convert existing Word documents into Help files with minimal cost. The use of World Wide Web technology had been considered but it was rejected at this stage because of the lack of simple conversion tools – it would have been necessary to re-process each Word document manually. However, the ability to switch to WWW at some stage in the future was identified as a future requirement as many of the new software teams were using UNIX as their development platform (rather than MS Windows).

4. Development of a Prototype

A prototype of SEEM was developed based on the Software Systems quality documentation which, at the top level, was already structured around the ISO 9000-3 TickIT guidelines. The standard development route within the model was based on the VME 'V-model' (but much simplified containing nine phases compared to VME's eighteen phases). The target users for the prototype were the same as those for the top level requirements document within the existing Software Systems quality system; that is, the quality assurance staff.

To allow HDK to work, each document was checked to ensure that it made use of standard Word 'styles.' Specifically, each document needed a heading using style 'heading 1' and the use of further headings needed to be strictly hierarchic. Conversion of the underlying procedures could then take place automatically, resulting in the production of a Help file with an associated hot-linked contents list. Thus, a user of the system can either step through the system using 'forward' and 'backward' buttons or he/she can jump directly to a section from the contents list.

Each section of the Software Systems Quality System top level requirements document contained a list of specific requirements and the introduction to the 'product development' section contained a picture of the V-model. Specific requirements included references to procedures or standards as appropriate. In some cases, the references were to generic processes such as the Software Inspection Process (Handbook). In other cases, the references were to specific VME documents. Minor

modifications were made to enable a list to be included of alternative references for different software development routes. The document was then converted to a Help file using HDK. The hypertext technology was further exploited by inserting hot links to the referenced documents (many of which had been converted to Help files) and hot links from each phase of the development life cycle depicted in the V-model to the corresponding list of specific requirements. This latter feature enabled an experienced user to jump directly to the information about a specific phase of the development life cycle.

In the cases where the referenced documents were part of the set of Word templates, two links were set up. The user had the option of jumping to the existing Word template Help file or loading the template into Word so that it can be processed. Similarly, links were set up to automatically load non-template forms and checklists directly into Word. A complete list of all templates, forms and checklists was included in the top level document as many of them were not directly referenced from specific requirements. Thus, the user was able to use SEEM as an interface to load Word templates, forms and checklists.

5. Organisational Changes

The reorganization of ICL High Performance Systems in September 1995 resulted in the dissolution of Software Systems and the formation of project led activities within business streams. The Technology and Engineering unit (T&E) set up a Systems Engineering unit with responsibility for:

> *"defining development strategies, processes and methods required to deliver the business objectives and ensure integrity across the whole development life cycle for all ICL High Performance System products."*

The responsibility for SEEM was transferred to T&E with the consequence that, on one hand, the scope of SEEM was narrowed to only cover engineering development route activities (project management and quality management activities were no longer in scope) and, on the other hand, the scope was extended to cover hardware, systems integration and systems validation engineering activities. Further, the target audience for SEEM was extended to cover the wider engineering community rather than just the quality infrastructure. SEEM was renamed the 'Systems Engineering Excellence Model' (previously, the 'Software Engineering Excellence model').

6. Revised Development of SEEM

Following the prototype of SEEM, the next stage of development was to include a second software development route which was not VME-based. The quality system for DAIS was selected as a suitable example, to be captured into the model. A review of how to achieve this immediately highlighted two major issues. Firstly, the structure of the DAIS documentation did not naturally fit into the generic model that was based on the ISO 9000-3 guidelines. The consequence of this was that, if the structure used in the prototype was maintained, the top level requirements would need to be rewritten each time a new specific development route was introduced. The second issue was that the structure, being based on quality system requirements, did not adapt easily for use by the general engineering population. As a consequence, an architecture for the structure was designed.

6.1 SEEM Architecture

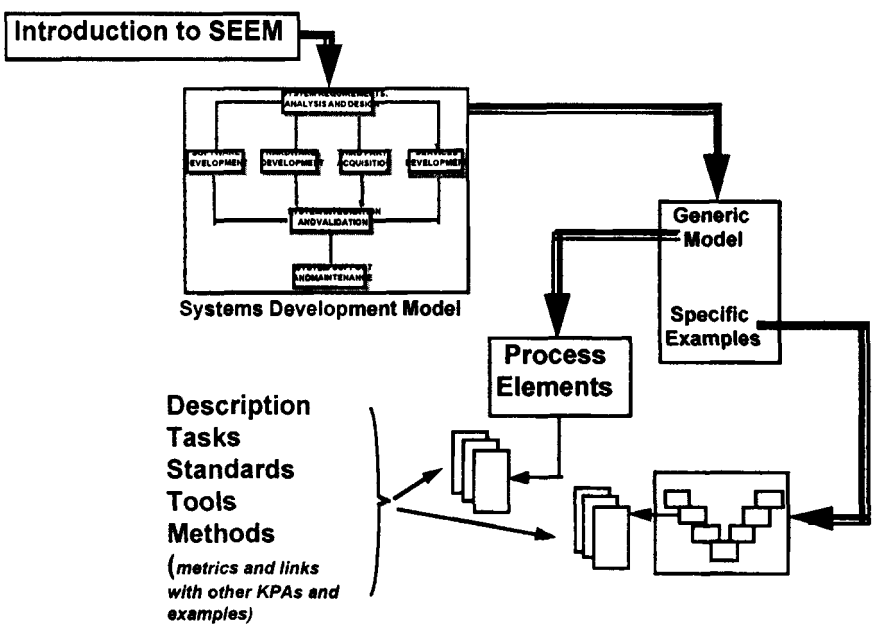


Figure 2: Architectural Model of SEEM

Entry into SEEM is through the 'Systems Development Model' which structures the major components of the systems development life cycle into:

- **System requirements, analysis and design**; carried out at the system level and resulting in the identification of the components that will make up the system
- **Software development** and/or **Hardware development** and/or **Third party acquisition** and/or **Services development**; which will typically be carried out using different types of development processes, tools and methods. These categories may be further subdivided into different types to reflect alternative approaches for different technologies, types of products, and scale and complexity of the development activities
- **System integration and validation**; integrates the constituent components of the system, validates that the system meets customer requirements and releases the system to the customers
- **System support and maintenance**; provides support to the customer base, including problem resolution and the provision of enhancements.

SEEM is implemented as a set of interlinked Help systems. The Overview module contains background information on development routes and capability maturity models and it provides links to the generic and specific development routes for each of the key components of the systems development model. Each generic and each specific development route is implemented as a separate Help system and can be implemented independently from each other. Once registered in the Overview module, new versions of the separate systems will be picked up automatically.

Each development route contains a control module which provides links to 'Process elements,' which are defined components of a development route. For specific examples of development routes, the control module contains the specific life cycle model used by the project and this model is used to link to the individual process elements. If the specific example to be captured in SEEM already has a description of its development route in an existing Word document, that document can be easily converted to the control module. For the generic models, the process elements are based on external best practice standards.

The process elements are described in a consistent way, based on the standard used for developing the VME system. The description includes:

- a **definition** of the process element, based on externally recognised terminology where practical

- a **list of tasks** to be performed. For specific examples, links to detailed procedures are included where applicable. For generic examples, the list of tasks reflects best practice
- references to **standards, tools and methods**. For specific examples, the references are to actual standards, tools and methods used by the system. For generic examples, the lists are a consolidation of known standards, tools and methods used within the specific examples and supplemented by further external examples where possible.

(It is intended to extend this list in the future to include metrics and links with other process elements (referred to as KPAs – key process areas – in the diagram)).

Forms, checklists and Word templates which are not converted to Help files are held in a separate directory which is accessible from any of the Help systems, subject to them following standard rules for setting up the links.

7. Deployment of SEEM

The prototype of SEEM was published in November 1995 and it demonstrated that the hypertext technology was a suitable way to document development route requirements. In February 1996. SEEM was used to assess the development route for a new series 39 project and it provided a ready-made checklist to facilitate the assessment. The assessment was more thorough and took less time to carry out. With hindsight, the use of SEEM to define the development route in the first instance would have prevented many of the issues (in the words of the development manager: "If only I knew about this tool before I spent all that time putting the development route together!").

The second version of SEEM, published in early June 1996, contained significant restructuring and captured the DAIS development route. It was demonstrated to the DAIS project team and was well received, creating an opportunity to provide on-line access by all staff to their development route procedures. Further demonstrations to a number of other units have also been well received, the major benefit being that the model has the potential to provide a single interface into an effective capture and retrieval system for the division's key engineering procedures which is independent of project structures and, hence has permanence after the life of a project.

A third release of SEEM was made in September, 1996 which added a number of further enhancements:

- The generic process for software development was included, based on external best practice models (CMM, SE-CMM, SPICE, Trillium (the Telecommunications variant of the CMM developed by Bell Atlantic), and ISO 12207 (Standard for Software Development Life cycle Processes))
- Life cycle models in addition to the ICL High Performance Systems standard V-model was included to give more flexibility to the way that projects meet their requirements
- New modules were introduced for Systems Integration and Validation (including Release processes) based on current ICL High Performance Systems practices.

8. How to use SEEM

The Systems Engineering Excellence model is used for three main purposes:

1. to assess an existing set of development processes against evolving best practice
2. to establish a set of development processes to support a project's requirements
3. to retrieve and process project forms and documents on-line.

There are two ways to establish the process. The first and simplest way is to select an existing set of processes which have been used on a previous project. After the initial system requirement analysis phase, the project will have defined a top level design which will determine the types of products to be produced and whether they will be developed in-house, through acquisition from a third party, or a combination. A number of different development routes are defined for each of the major product types (software, hardware, service, or third party). The project team can select an existing route which best fits the characteristics of the new project.

However, a review of the selected route should also be carried out to check how well it meets the requirements of the generic development processes. Process improvement actions should be put into place to address any shortfalls which are deemed to be necessary to meet the project's needs and to ensure ongoing conformance to ISO 9001.

The second way to establish the development process is to design it from scratch using the generic processes as the starting point. Each generic process identifies the key tasks to be performed and provides a list of tools and methods which may be used to execute the tasks.

SEEM can also be used to provide general awareness and technology transfer on all aspects of development routes.

9. Example Usage of SEEM

This usage is illustrated by the following example based on current implementation:

- On entry to the on-line Help system, the Systems Engineering logo is displayed in the left hand window and the table of contents are displayed in the right hand window. The user can browse the introduction to gain an understanding of the rationale behind SEEM or may go directly to the Systems Engineering Development Life Cycle Model, Figure 3

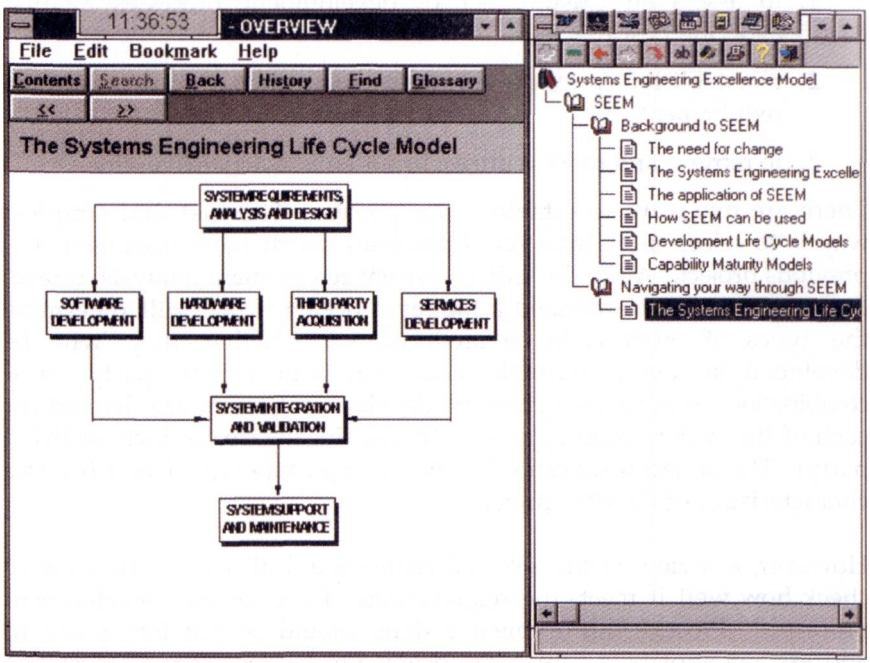


Figure 3

- The Systems Engineering Development Life Cycle Model provides linkages to further on-line Help systems. For example, clicking on "Software Development" will jump to a linkage module

- The linkage module, Figure 4, links a generic (best practice) definition of the process and specific development routes for software development projects. The specific routes will vary from project to project as they are based on the existing quality system documentation. The next steps illustrate the on-line Help system for VME

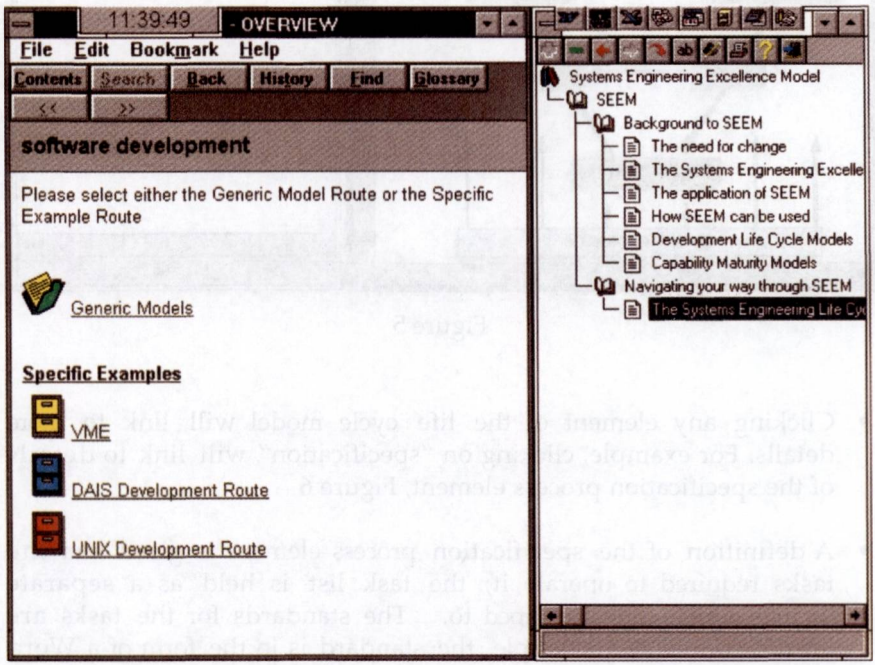


Figure 4

- As with the overview module, the user has the option of browsing background information or he/she can jump directly to descriptions for parts of the development route or to a list of checklists and Word templates which can be downloaded for printing or processing on-line. In this example, we will link to the development life cycle, Figure 5

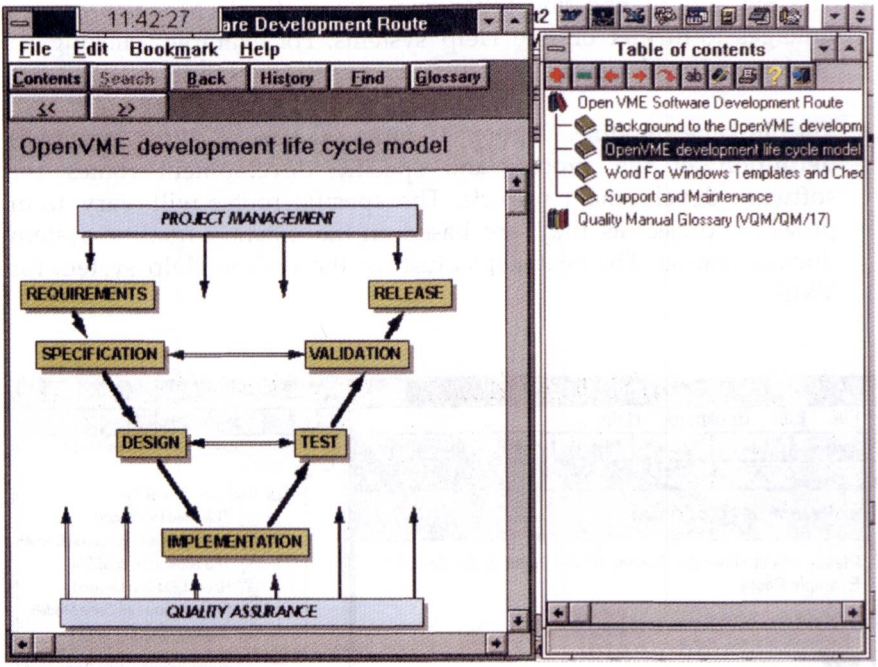


Figure 5

- Clicking any element of the life cycle model will link to more details. For example, clicking on "specification" will link to details of the specification process element, Figure 6

- A definition of the specification process element is given and the tasks required to operate it; the task list is held as a separate section which can be jumped to. The standards for the tasks are identified. In this example, the standard is in the form of a Word for Windows template. Clicking on the title will link to the existing Help system for Word for Windows templates

- Alternatively, clicking on the red button will download the template into Word. Note that the current implementation loads the .DOT file which can be modified and printed, or saved to an alternative location. The system is not able to create a new file using the named template.

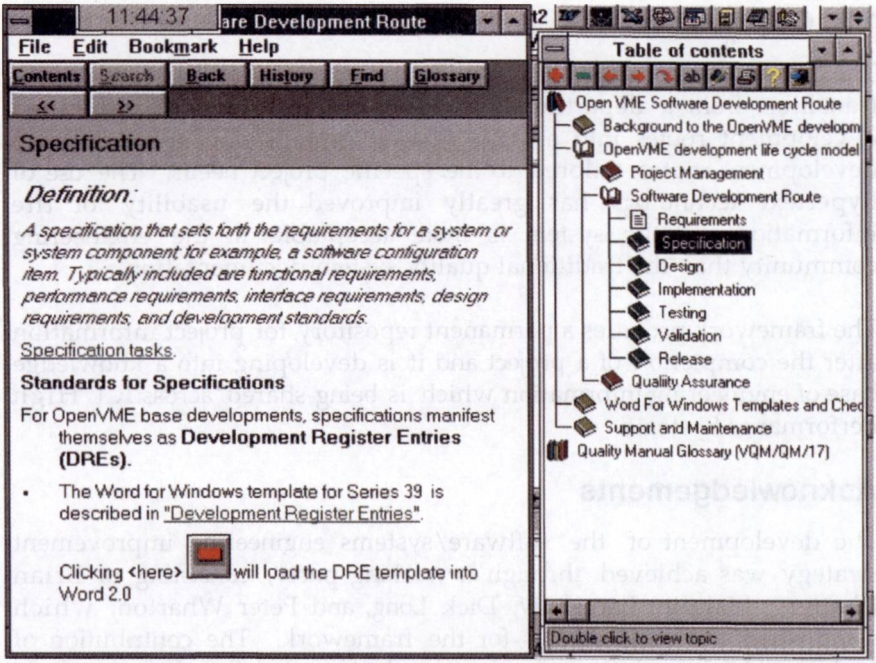


Figure 6

10. Future Enhancements

It is intended that SEEM will become the repository for all development route information within ICL High Performance Systems, creating a knowledge base which can be tapped by project engineers.

Considerations are being given to converting the system to HTML files so that they can be processed through Web Browsers. In addition, the structures and interfaces into SEEM are being reviewed with the aim of making them more acceptable as on-line tools to steer the engineers through their development route when actively working within a project.

The links between the generic processes and the specific examples are, in many cases, only loosely defined. This makes assessment of specific processes against the generic process more difficult (and, hence, it is more difficult to provide a measurement against the CMM or other best practice model). Further work is planned to address this issue.

11. Conclusions

The introduction of SEEM has provided an effective tool to support the transition from a departmentalised organisation using well-defined development routes into a project-based organisation using bespoke development routes, tailored to the specific project needs. The use of hypertext technology has greatly improved the usability of the information and the system is more acceptable to the engineering community than the traditional quality system documentation.

The framework provides a permanent repository for project information after the completion of a project and it is developing into a knowledge base of engineering information which is being shared across ICL High Performance Systems.

Acknowledgements

The development of the software/systems engineering improvement strategy was achieved through a working party, consisting of Brian Chatters, Malcolm Earnshaw, Dick Long, and Peter Wharton, which established the requirements for the framework. The contribution of Malcolm, Dick and Peter in the development of the framework is gratefully acknowledged.

A special thanks to John Kazimierczyk, MSc student from Stafford University, who has helped to turn the idea of the framework into a reality.

References

COLLIER, F. et al, "TRILLIUM: Model for Telecom Product Development and Support Process Capability, Release 3.0," Bell Canada, 1994.

DTI, "TickIT: Guide to Software Quality Management System Construction and Certification," DTI, 1992.

HERBSLEB, J., CARLETON, A., ROZUM, J., SIEGEL, J., ZUBROW, D., "Benefits of CMM-Based Software Process Improvement: Initial Results," CMU/SEI-94-TR-13, Software Engineering Institute, 1994.

ISO, Information Technology - Software Engineering Software Process Assessment (SPICE), Parts 1 - 9, ISO/IEC/JTC1/SC7N/1405 - 1413, ISO, 1995.

"ISO, Information Technology – Software Life Cycle Processes," ISO/IEC 12207, ISO, 1995

PAULK, M. C., CURTIS, W., CHRISSIS, M. B., et al, "Capability Maturity Model for Software," CMU/SEI-91-TR-24, Software Engineering Institute, 1991.

"SEI, A Systems Engineering Capability Maturity Model," Version 1.1, SECMM-95-01, Software Engineering Institute, 1995.

Biography

Brian Chatters is a Software and Systems Engineering Consultant within ICL High Performance Systems, responsible for developing the organisation's engineering capability. From 1987 until 1995, he was the Quality Manager for Software Systems and he has worked in the Software Industry for 30 years in various roles including programming, strategic design and project management. He graduated from Bristol University with an honours degree in Mathematics, he is a Fellow of the Institution of Electrical Engineers and a Chartered Engineer.

Cochise: a World Wide Web interface to TPMS applications.

Alan Beale

OpenPLUS Ltd, High Performance Systems, ICL, Kidsgrove, Staffs.

Abstract

The last two years have seen an explosive growth of the World Wide Web. Originally conceived as a mechanism to enable scientific researchers to share documents and other resources, the Web is fast becoming ubiquitous. Newer generations of graphical Web browsers, available on all common desktop platforms, and powerful, fully-featured Web servers, available on a wide variety of server platforms, coupled with the prevalence of the TCP/IP protocol suite as the network connectivity medium of choice are revolutionising the IT industry, the way organisations do business, and the way individuals spend their leisure hours.

Against this background, the traditional mainframe system—hidden away in its controlled environment, supporting dozens of applications, thousands of users and millions of transactions—would seem to be completely isolated from the possibilities opened up by the Web.

This is not the case, however, for OpenVME enterprise servers—mainframe being a deprecated term these days. The availability of VME/X on S39 systems makes it possible to marry the best of the old—large, powerful proprietary systems—with the best of the new systems based on open standards. This paper describes *Cochise* (originally named Geronimo): a project aimed at integrating existing TPMS applications with the Web. A prototype has been in operation for some months, used for development and customer demonstrations, and the first release of the product was scheduled for release in the last quarter of 1996.

1. History of the project.

OpenPLUS was set up to focus on the 'open' aspects of OpenVME. In practice, this has meant exploiting the capabilities provided by TCP/IP on S39, and VME/X, the XPG4-branded Unix environment. Apart from participating in ports of commercial software to the VME/X platform—products such as Ingres, Informix, Oracle, Legato's Networker (known as CustodianPLUS), and TeamOffice—we have ported a number

of non-commercial software packages because they seemed likely to be useful to us and to our customers. Examples of this are the Berkeley spooler, Perl—a powerful interpreted scripting language, widely used for Unix system administration tasks, and SAMBA—a Unix-based implementation of the Server Message Block protocol which underpins Microsoft's LAN Manager and Workgroups networking. All of these have become part of the *InterworkingPLUS* portfolio, a collection of software and services aimed at enabling our OpenVME customers to integrate their S39 systems with their networks of PCs, Unix, and other systems.

Just over two years ago we ported Plexus, a World Wide Web server written in Perl. Initially, the aim was to see what use we could make of the Web internally. The Web server proved useful for dissemination of a variety of technical documentation, and also for product descriptions and similar marketing collateral. The Plexus server set up at ICL's development centre in Manchester is still running; undoubtedly the oldest VME-based Web server in the world. Around the same time, I had been experimenting with a variety of Unix based SQL engines, with a view to harnessing them to ReView, the SQL interface to IDMSX. I came across a couple of Web interfaces to SQL and decided to set up a Web interface to the ReView demonstration database, as an example of using VME/X as the integrator between the 'legacy' world of IDMSX and the 'open systems' world of the Web. This was demonstrated at the Nottingham Engineering Conference in September 1994 and at the AMSU event in October, and a number of customers expressed interest.

Although the Plexus server was quite adequate for a small scale server, it lacked functionality in a number of key areas, such as authentication and proxying. To overcome this we undertook the port of the CERN server. CERN was the original home of the Web, and their server was the first. It is still in widespread use today on a variety of Unix platforms, VMS and other systems. Indeed, ICL's gatekeeper service at Feltham runs a CERN proxy server—this is the common exit point to the Internet for all Web traffic originating within ICL. At the Keele Engineering Conference in September 1995 and at the AXiS event in October—AMSU had changed its name—we again demonstrated OpenVME based Web servers, with some conventional hypertext documents, SQL access to ReView, access to VME OCFs and a postcode look-up application we had ported for a software house at the request of one of their customers. Interestingly, the customer wanted the application on VME so it could be accessed from his TPMS applications;

we ported it to VME/X as a training exercise, and decided to extend it with a Web interface. Again, a number of customers expressed interest in being able to use a VME/X Web server as a front end to VME applications and data.

Towards the end of last year, one of the customers who had seen the Web demonstrations at AX*i*S asked *OpenPLUS* to assist in a pilot project based on a VME/X Web server. The project was to make available to an internal group of users a number of VME batch facilities via the Netscape Web browser. Users would be able to initiate the running of a number of standard reports, monitor the progress of the jobs they had submitted, specify whether and where the output should be printed, or download the results in a variety of different formats, including CSV, suitable for importing into PC spreadsheets. In addition, users would be able to initiate ad-hoc queries by specifying search criteria to a Querymaster service. Since the users would be from different departments, it was vital that they should only be able to access their own data, output files etc.

The pilot was successfully implemented, using a mixture of VME SCL and VME/X Perl scripts to interface the Web server forms with the VME applications. Other forms were developed to enable users to register with the service and to change their own passwords. The users who tested the service were enthusiastic, as were the customer's VME staff. I was involved in helping with the VME/X and Web aspects of the project, and it started to dawn on me, and on the customer IT manager, that many of the techniques we were using to drive batch processing on VME from the Web server could be extended to drive interactive access to TPMS applications.

Additional impetus came from another VME customer visiting Manchester from Spain—they were quite interested in the batch processing but very enthusiastic about the possibility of using Netscape as a front end to TPMS applications. In December, I gave a presentation on *InterworkingPLUS* to an ICL customer event in Paris and a number of customers expressed interest in being able to connect to their TPMS applications over TCP/IP—either via one of the *OpenPLUS* FTH sponsors or the Web.

It was clear that quite a few customers were interested in TCP/IP access to VME applications in general, and some particularly interested in the Web. At the beginning of this year I started work on a prototype, primarily as an out-of-hours activity. By the end of February I had a

working prototype, using the ICL Forms demonstration TP service and database, as that was all I could get my hands on. I will cover the technicalities in more detail in the last section of this paper. Here I will just outline the salient points.

While the initial version of the prototype was based upon the CERN server, a new Web server had appeared towards the middle of 1995. This was Apache, a descendant of the NCSA server, which was the second popular Web server to appear after the CERN server. The latest release of Apache, released earlier this year, offered a number of technical advantages over the CERN system—which will be dealt with in the next section of this paper—so I switched the prototype to Apache, once I had ported it to VME/X. This took about half a day. Since the prototype was now built on Apache, I thought *Cochise* would be an appropriate name for it—so *Cochise* became the internal name for what had previously been known as 'webby thing' or 'TP Web interface.'

I demonstrated the prototype to a number of customers visiting either Kidsgrove or Manchester over the next couple of months, in particular, to a group of ICL France customers visiting Manchester for a strategy and product update at the end of March. The IT department of one of these customers had decided that, since there was a business requirement to allow their own suppliers and customers to access certain of their VME applications, the best way to do this would be via the Internet. In this case, it seemed far more attractive to them to be able to offer TP access via a 'shrink-wrapped' client such as Netscape, rather than have the headaches of supplying or recommending terminal access software—quite apart from the obvious attractions of a GUI interface above a 'green screen'. This customer will be mounting an internal trial of *Cochise* at the beginning of September.

With growing customer interest in all things related to the Internet and especially the Web, the *Cochise* project has moved from a speculative prototype to a fully-fledged product development. A number of reviews have been held to discuss technical and resource issues, and to involve a number of *OpenPLUS* development staff in the project. A formal business case and marketing requirements have been prepared and technical specifications of the enhancements required to meet functionality, performance and scalability targets are underway. The technical issues are covered in the next section; I will close this section with some of the business requirements as expressed to me by a number of customers.

– *TCP/IP is the networking technology of choice.*

Regardless of the technical merits of OSI protocols versus TCP/IP, TCP/IP connectivity has become essential. Since Microsoft now bundle a TCP stack with their desktop systems it is difficult to justify buying other protocols. Reducing the number of protocols required on the desktop buys reduced support costs as well as better reliability and lower software costs. TCP/IP has been available on S39 since 1991 but ICL has not made the most of this.

– *the Internet and the Web.*

Internet connectivity implies TCP/IP. Everyone wants to be 'on the Internet'. The Web is the most visible manifestation of the Internet, far above mail and news, although the volume of traffic generated by the last two is huge and growing. The fact that software giants like Microsoft and Oracle are placing great emphasis on Web-oriented products, both client and server, underlines the importance of this phenomenon.

2. Cochise: what it is and how it works.

Functionally, *Cochise* is a layer of middleware, sitting between the Netscape browser client and a variety of servers—the TPMS applications. *Cochise* is unique because of its location, within the VME/X environment of OpenVME. All other products that I have encountered which attempt to address the same problems—including Gresham's Casablanca, and IBM's Web to CICS interface—require an intervening Unix or NT system to host the Web server and associated middleware.

The initial *Cochise* prototype is fundamentally a 7561 screen-scraper, with the addition of HTML templates generated from the DDS by a 'DDS scraper.' This gives a 'green screen' image of the TP screens in a Web browser window. The second generation prototype is currently under development and incorporates elements of ICL Forms technology to enable the inclusion of graphical attributes with the TP screens. This version will become the first release of the product—to be positioned as part of the ICL Forms product family, under the name *Cochise*, as trademark reasons prevented the use of the original name Geronimo.

2.1 Prototype Mark I

Cochise is predicated on use of the DDS. There is an 'off-line' component which extracts data about the screens comprising the application from the DDS and generates HTML templates, which are then stored in VME/X filestore for use by the runtime components. This 'DDS scraper' is a VME/X Perl script which processes the output of the DDS_UTILITY program. The runtime components consist of the Apache Web server, a Perl CGI script called 'tpscreen' and a Perl daemon called 'tpweb.' Basically, 'tpscreen' displays a form for the user to fill in; once filled in, the form contents are sent to 'tpweb' which then handles an intra-system connection to TP and passes the screen sent by the application back to 'tpscreen' having converted the EBCDIC data and format effectors into an ASCII screen image; 'tpscreen' then merges the data from the screen into the HTML template generated from the DDS and sends the results to the Web browser.

2.2 Prototype Mark II

There are two major changes from the first prototype, although the overall structure is the same. Firstly, the DDS extraction is done by a version of the Forms template generator, modified to output encoded templates which are then used to generate HTML at runtime. Secondly, instead of 'tpweb' connecting directly to the TP CVM and handling 7561 terminal screens, 'tpweb' now connects to the Forms Sponsor VM via an ADI link and processes Forms protocol. The HTML generation is now done by 'tpweb' rather than 'tpscreen' which becomes a simple message passing layer. The 7561 'screen scraping' capability is modified to become generic, without requiring templates from DDS—this will enable it to be used to interface to other applications such as conversational COBOL packaged MAC environments.

2.3 Overall design considerations

2.3.1 DDS based

From the beginning, it seemed sensible to use the DDS as the repository for all the application based data required at runtime by the Web middleware. This was primarily to avoid DDS extraction tools needing to cater for multiple input formats, particularly as there are tools which can import most other known formats into the DDS.

Apart from that technical consideration, it also seemed desirable in terms of product positioning to emphasise that *Cochise* was built firmly on the existing VME application development environment.

– *No change to existing TPMS applications*

The aim all along has been to enable existing applications to benefit from the new connectivity and client software made possible by the growth of the Web. It seemed therefore essential to be able to do this without needing to rewrite the applications, otherwise the costs could be seen to be prohibitive. This fits in well with the philosophy behind the Forms product family—as a first step, enable existing applications to benefit from the new front end technology without code change, thereby avoiding any 'big bang' effects; subsequently, take advantage of the new facilities, if so desired, in a phased manner.

– *Generate, don't write code.*

The *Cochise* middleware has been designed to be installed without requiring any user written code. Once the screen related data is in the DDS, default encoded templates for HTML generation can be generated immediately, without further work. This enables 'grey screen' look and feel within the browser. If modification and refinement of the GUI screen attributes is required, this can be accomplished with a modified version of Forms WindowMaker, a graphical screen design tool. This can be done in a phased manner, so there is no requirement for a 'big bang' approach.

The middleware components themselves are data driven, and because the data is partly derived from the DDS and partly from the application at runtime, the merger of the two requires no user hooks or data mapping modules. This is in sharp contrast with, Gresham's Casablanca, for example, and avoids one of the major pitfalls which has probably dissuaded a number of customers from taking advantage of the ReView product.

To enable client-side handling of certain processing, such as data validation, derived from rules in the DDS, the template generator has facilities to incorporate script modules in the HTML generated at runtime. Because of the volatility of the HTML specifications in this area, and the fact that different browsers support different scripting languages, it is likely that the first release of the product will only support Netscape's Javascript extensions. Future releases will need to support other scripting technologies, for example Microsoft's VBScript and Oracle's BASIC. Some ingenuity will be required in the HTML generation phase to ensure that the scripting modules embedded in the HTML are appropriate to the browser client.

– *Performance and scalability.*

Given that the data stream from the TPMS application needs to pass through a number of additional VMs on its way to the client browser, it is always going to be difficult to achieve the sub-second response times to which 7561 connected users are accustomed. Similarly, because the HTTP protocol used between Web clients and servers is inherently stateless, it is impossible to maintain client context information in the Web server, so *Cochise* overcomes this by maintaining a dedicated 'tpweb' process per client, which imposes limits on scalability due to VME/X restrictions on the number of processes which can be supported in a single VM. The design of *Cochise* has evolved over the last few months to deal with these issues.

There were two main reasons for switching from the CERN Web server to Apache: the process model, and the fact that Apache had a documented API and the ability to support the incorporation of extension modules.

The CERN process model involves forking a new process for every incoming client connection, and also involves a fork/exec to run any CGI scripts or programs, which is the normal method of invoking customised server functions. The Apache server employs a pre-forking mechanism whereby a configurable number of Apache processes are created when the server is started—this is analogous to the handling of AVM pools in TPMS. In addition, it is possible to build extension modules into the Apache server to handle the requests which would otherwise result in the fork/exec of another process.

So, by configuring Apache to pre-fork the required number of processes, and by incorporating an embedded Perl interpreter with direct access to the Web server API, *Cochise* can handle a client connection and run the 'tpscreen' script in a single process, instead of requiring two forks and an exec. This has had a dramatic effect on response times, reducing them by a factor of 8 to 10 from the original CERN prototype, and by a factor of 4 from the original Apache server without embedded Perl.

To address the scalability issue, the 'tpweb' software has been re-designed to pre-fork in the same way as Apache, with the option to run multiple daemon VMs if more processes are required than can be supported in a single VM. As *Cochise* moves towards using Forms protocol instead of 7561 screen protocol, the CPU time required to process the application output and generate the HTML for the client is

expected to drop dramatically, again reducing response times and overall VME resource utilisation.

In both the 'tpscreen' and 'tpweb' software, we are considering re-engineering parts of the processing in C rather Perl. This is part of an ongoing performance monitoring/profiling exercise which we have started recently, to ensure that the attractive new functionality and connectivity offered by *Cochise* is not compromised by unacceptable response times or excessive CPU utilisation.

– *Security and authentication.*

The current HTTP version 1.0 specifications define a method for requiring user authentication before access is permitted to parts of a Web server's resources. On most servers this is done in conjunction with Unix-style user and group identities, controlled by mapping files maintained by the system administrator. For *Cochise* this concept has been enhanced to enable VME based authentication—the user/person/password submitted by the Web client can optionally be verified against the VME catalogue.

A drawback of the HTTP 1.0 scheme is that the user and password are only base64 encoded before transmission across the network, which makes them potentially vulnerable to network sniffers. It can be argued that this is no more vulnerable than current terminal login access, when the cleartext password is potentially visible on the network. HTTP 1.1 defines an alternative authentication scheme in which the authentication data is encoded in an MD5 digest before transmission. This is certainly less vulnerable to network sniffing, but is of limited use until HTTP 1.1 compliant browsers are more widely available than is the case at present.

For *Cochise*, other authentication schemes are being investigated in the light of customer requirements, including Kerberos based authentication, which could potentially enable integration with ICL's Access Manager system.

Standards in the areas of Secure HTTP, Secure sockets and Secure Web transactions are evolving and, as they stabilise, *Cochise* will be in a position to offer optional security modules to meet customer requirements. For example, we already have a prototype of Apache-SSL underdevelopment. Another possibility could be a collaboration arrangement with a vendor of commercial Web server technology, such

as Netscape, to intercept the emerging standards and port the resultant implementations to VME/X.

Conclusions

Cochise brings the Internet and the 'Web' to VME.

Cochise, from prototype, to project, to product, is part of the evolution of the InterworkingPLUS portfolio of software, services and consultancy, aimed at enabling OpenVME customers protect their investment in S39 and add value to their existing applications and data.

Cochise capitalises on the unique position of VME/X on S39 and by exploiting the Unix environment on OpenVME, it is able to combine the strengths of 'open systems' with the proven strengths of VME.

Cochise enables ICL's customers to start to realise the benefits of 'client/server' computing without having to develop and deploy the client software components.

Cochise is now demonstrable in ICL's development centre at West Gorton, Manchester.

Glossary

VME – native operating system of Series 39.

VME/X – extension to VME to offer POSIX-standard operating system interfaces.

OCFs – Operational Control Forms within the VME operating system.

DDS – Data Dictionary System.

IDMSX – Codasyl-standard database system.

ReView – SQL interface to non-relational databases.

7561 – popular name for the character screens supported by ICL mainframes in the late 1970s and emulated by a succession of newer products ever since.

ICL Forms – user interface system to allow VME applications to be used from a Windows-style graphical interface.

TP CVM – Control virtual machine within ICL's TPMS transaction processing monitor system.

ADI: Application Data interchange protocol (interprocess communication, local or remote) packaged MAC: multi-access computing, the normal interactive environment for non-TP work (including programming), constrained to disallow access to the system command language.

Acknowledgements

The author wishes to thank Mark Bayliss, Stephen Bennett, Mike John, Cassandra Morris and Alan Pound for their contributions to the project and to this paper.

Biography

Alan Beale read History at Christ's College, Cambridge and received a BA degree in 1969. Although his experiences with computers at Cambridge were confined to helping to build the Computer Centre during a summer vacation, he gravitated towards the computer industry after graduation and joined the 'New Range' division of ICL at Kidsgrove, Staffordshire in May, 1973. He was involved in the testing, integration, validation and support of all the early VME/B releases.

In 1977 he was seconded to the EEC project in Luxembourg and experienced life on the 'customer' side of the fence for four years. On return to the UK in 1981, he eventually returned to Kidsgrove in 1982 and became involved in the testing and validation of what became the Series 39 range of systems. The VME based departments were transferred to West Gorton, Manchester in 1984-85, and Alan continued to work on Series 39 integration and validation. He then spent several years in the Key Accounts unit, working with VME customers in England and France, and in 1991 joined the Open VME Exploitation Centre to concentrate on helping customers to exploit the 'open' aspects of VME. In 1994 he joined *OpenPLUS*, a new organisation formed specifically to focus on these areas.

Obituary

Gordon G. Scarrott, F. Eng., formerly manager of ICL's Research and Advanced Development Centre in Stevenage, died on 26 October, 1996 at the age of eighty.

Peter Hall (retired Director of ICL) writes:

I first met Gordon Scarrott in the mid 1950's when Sir Vincent Ferranti appointed me General Manager of the Ferranti Computer Department. I remember very clearly our early meetings. He so impressed me that I made him responsible for Research and Development reporting directly to me. In that role he looked after our development of Pegasus and Mercury, kept a general eye on our collaboration with Manchester University on Atlas and, most importantly, was responsible for the development of the neuron technology (ballot box logic), which he then used in the development of the Sirius and Orion computers.

Gordon had established his reputation in the computer field, some years before I met him, by his invention of the torsional delay line store. These stores were not only used in Ferranti computers but were sold to third parties, particularly in the U.S.A.

It is fair to say, I think, that Gordon was happier working on research projects, and establishing new technologies, rather than managing large development projects like Orion. Later, after the Ferranti Computer Department was taken over by ICT, he was appointed manager responsible for research. With his own hand-picked team, and free of major development responsibilities, there followed a period of outstanding achievements. Not only was Gordon outstanding in his grasp of the underlying problems in the computing and information processing fields, he was able to come up with imaginative and practical solutions. Gordon was an inspiring leader who attracted the highest quality staff into his team. He gave them freedom to work on their own ideas, given that the possible outcome was in-line with the needs of the company as he saw them. Sadly, the output of his team was sometimes ahead of the company's readiness or capability to exploit it.

Gordon Scarrott, and his team, will be remembered for their contributions in the fields of computer architecture, character recognition, voice recognition

and synthesis, content addressable file stores and distributed array processors. In his later years Gordon was very much concerned with the "Software Crisis" and the lack of a fundamental theory of Information Science. He was to have given an important paper on the subject to the IEE the week following his death.

Many of us will particularly remember Gordon for the way he inspired his staff and for the way he fought for them and their ideas, through all the usual traumas inevitable in a large public company.

John Pinkerton (formerly Research Manager of English Electric Leo Marconi Computers and editor of the ICL Systems Journal) writes:

My first impressions of Gordon date from the late forties at the Cavendish Laboratory, where he was a research assistant and I was a research student. We did not then see much of each other. Our next encounter was brief at a Physical Society exhibition, where he demonstrated a decade counter using only five pentodes. The usual circuit then used eight valves in four bistable pairs.

In the late fifties we met several times at Ferranti over orders for magnetic drums for LEO II. There were arguments about setting up circuits involving delay lines which had to be adjusted on test.

Between 1963 and 1968 when LEO became progressively integrated with English Electric we must have met sporadically. But when in 1968 EE Computers merged with ICT to form ICL, the new company then found itself with two research departments one run by me at Acton, the other by Gordon, and it reasonably decided to fuse them under Gordon. I then joined the 2900 planning team under Michael Forrest. After that we interacted much more often. Gordon would phone me, hoping, say, that I might support Iliffe's Basic Language Machine as a basis for 2900. I did not, chiefly because I did not understand it. I did however try to win backing for CAFS (initially without success), notably from Maurice Wilkes.

Gordon and I agreed on many things but also had many differences of opinion. Earlier this year we had corresponded on his definition of Information. In discussion his attitude was invariably good-humoured and any rebuttals were courteous and patient.

I feel I have lost a good friend of whom I had grown very fond.

Charlie Portman (senior consultant, ICL Enterprises) writes:

Gordon had a wide range of interests from Motor Cars to Magnetic Recording, from Basic Science to the problems of Fabrication. His focus moved over the years during which he was Chief Scientist of Ferranti's Computer Department and Manager of ICT/ICL's Research and Advanced Development unit from circuit and component design through the design of large Subsystems to Processor and System Architecture.

His early work on torsional nickel delay lines made possible the 2000 bit lines used in Sirius.

He held many patents, including, among other things, one for variable valve timing for motor car engines.

His championing of the Distributed Array Processor (DAP) and the Content Addressable File Store (CAFS) made him an early proponent of parallel processing. His support for John Illiffe's Basic Language Machine (BLM) demonstrated his concern with the unreliability of computer systems and the "software crisis."

Gordon believed passionately that Computer Science was lacking a Theory of Information and that this was an essential component in devising new system architectures that would take account of both component technology and the use of information in human society. He believed that the "von Neuman" design of machine was holding back the effective utilisation of the available technology for information processing. After his retirement he spent much of his time pursuing these themes with information engineers.

Many of these attitudes are well shown in the text of his talk to the Computer Conservation Society given in March 1995 and printed in the society's journal, Resurrection, issues 12 and 13.

Gordon was ingenious, logical and persuasive. He identified original lines of development and pursued them with vigour against considerable opposition and inertia. The computer world has lost an innovator and champion of science based engineering.

Victor Maller (successor to Gordon Scarrott as manager of the Research Centre in Stevenage and now editor of the ICL Systems Journal) writes:

The fourteen years, during which I worked for Gordon in Stevenage, were among the most enjoyable and productive of my career. He was an inspira-

tional technical leader, always bubbling with new ideas, often in fields other than computing, and he possessed that wonderful facility of breaking down complex problems into their fundamental components and then suggesting a totally novel solution, which, when explained, seemed almost self-evident. He loved technical argument and knew the difference between an argument and a quarrel; he was often provocative, sometimes outrageous but always stimulating. He did not suffer fools gladly and detested what he called instant invention: the spontaneous ad hoc solution to a problem, which had not been thought through properly.

During the Second World War, he was an artillery officer and, rather than throwing things at the enemy, he worked on echo ranging systems, which introduced him to electronics. This military experience also influenced his views on management and there was one lesson on which he placed particular emphasis: "loyalty downwards has to be given but loyalty upwards has to be earned." That he put this principle into practice will always be remembered with appreciation and gratitude by his former staff.

His broad understanding of physics, and its application to engineering, was heavily influenced by his experience at the Cavendish Laboratory, Cambridge, where, after he had left the army, he worked under the Nobel prize winning physicist, Otto Frisch, in developing the world's first electronic pulse height analyser. His development of torsional delay line stores in this project led eventually to his moving to Ferranti Computers and thence to his subsequent career in ICL.

Gordon was appointed head of the Stevenage Research Centre in 1967 and remained there until his retirement in 1981. In 1968 the Centre was expanded by adding to it the research department of English Electric Computers, after the merger which formed ICL.

The combined department's activities were very broad, covering storage device research, processor architectures, high density component packaging, design automation, fluid logic control systems and speech recognition. Gordon recognised that the main thrust of computing research was moving away from hardware technology towards systems and architectures. The era of how to make computers was giving way to an era of how they were to be used effectively. Gordon then concentrated the research activities in four main areas: processor architectures, speech and pattern recognition, array processing and content addressable file stores.

The success of this strategy was recognised by the department winning the BCS award for technical innovation for the Distributed Array Processor and CAFS in successive years. CAFS then went on to win an award for Product

of the Decade (1973 – 1983) and later brought to ICL the Queen's Award to Industry for technical innovation. Not a bad record for a small department.

Computing, to Gordon, was an engineering discipline and not a science, although it was science based. He much preferred the term Information Engineering to Computing and the phrase, Information Technology, or IT, made him wince. His approach to the subject was holistic and he was very concerned about the field being prejudiced by a cultural split between software engineers and hardware engineers. Paraphrasing Clausewitz, programming, to Gordon, was no more than logical design by other means.

He believed that Information Engineering had to be underpinned by an Information Science and this philosophy was most eloquently described in his Clifford Paterson Lecture to the Royal Society in 1979: *From computing slave to knowledgeable servant: the evolution of computers.* In this paper, he cogently argued that there were no adequate theoretical foundations for Information Engineering and he likened the building of computers to the building of steam engines before the discovery of the laws of thermodynamics. If one suggested that this was perhaps pushing an analogy rather too far, he would retort that it was only after thermodynamics had been understood and the concept of the generalised heat engine developed, that it became possible for others to invent the internal combustion engine, the steam turbine and the gas turbine, and where would the twentieth century have been without these. He felt intuitively that something similar would eventually occur in the field of Information Science with consequential benefits to Information Engineering.

After his retirement, he pursued his theoretical investigations and this led him into considering the purpose and nature of information in human society. His studies embraced the evolution of natural language and took him into the fields of ethics and moral philosophy. Although an overall theory eluded him, he did ask many stimulating questions in the papers he wrote and he was still working on his last paper on this subject a few hours before he died. This paper was given posthumously to an IEE workshop a few days later.

Those who worked with him have lost not only a valued colleague but a true friend and his former staff at Stevenage will always be proud to say, "I was one of Scarrott's team."

Previous Issues

Vol.11 Iss.1 - May 1996

The Internet and how it is used
An Architecture for a Business Data Warehouse
Virtual Reality as an Aid to Data Visualization
Re-engineering the Hardware of CAFS
An Innovative Solution for the Interconnection of Future Component Packaging
Development of Practical Verification Tools
Coupling ORACLE with ECLiPSe
Integrating the Object Database System ODB-II with Object Request Brokers
SAMSON and the Management of SESAME

Vol.10 Iss.2 - November 1995

The Architecture of the ICL GOLDRUSH MegaSERVER
The Hardware Architecture of the ICL GOLDRUSH MegaSERVER
CAL in Higher Education - Potential and Pitfalls
The UK Technology Foresight Programme
Making the Internet Safe for Business
Developing Financial Services Kiosks
High Availability Manager
The Virgin Global Challenger
Design of the Format for EDI Messages Using Object-Oriented Techniques
New Aspects of Research on Displays

Vol.10 Iss.1 - May 1995

Object databases and their role in multimedia information systems
The ICL Multimedia Desktop Programme
Multimedia Information used in Learning Organisations
The Software Paradigm
Single Sign-on Systems
Why is it difficult producing safety-critical software?
Experiences using the Ingres Search Accelerator for a Large Property Management Database System
RAID
Improving Configuration Management for Complex Open Systems

Vol.9 Iss.2 - November 1994

Establishing Co-operation in Federated Systems
An ANSA Analysis of Open Dependable Distributed Computing
An Open Architecture for Real-Time Processing
Updating the Secure Office System
POSIX Security Framework
SQL Gateways for Client-Server Systems
Asynchronous transfer mode - ATM

The ICL search accelerator™, SCAFS™: functionality and benefits
Open Teleservice - A Framework for Service in the 90s
LEO, A personal memoire

Vol.9 Iss.1 - May 1994

Client-server architecture
How ICL Corporate Systems support Client-server: an Architectural Overview
Exploiting Client-server Computing to meet the needs of Retail Banking Organisations
A practical example of Client-server Integration
From a Frog to a Handsome Prince: Enhancing existing character based mainframe applications
Legacy systems in client-server networks: A gateway employing scripted terminal emulation
The Management of Client-server Systems
Dialogue Manager: Integrating disparate services in client-server environments
Distributed Printing in a Heterogeneous World
Systems Management: an example of a successful Client-server Architecture
PARIS - ICL's Problem & Resolution Information System

Vol.8 Iss.4 - November 1993

Toward the 4th Generation Office: A Study in Office Systems Evolution
IPCS - Integrated Product Configuring Service
CGS - The ICL Configurer Graphics Service
Location Transparency in Heterogeneous Networks
Future Office Interconnection Architectures for LAN and Wide Area Access
Parallel Lisp and the Text Translation System METAL on the European Declarative System
Detecting Latent Sector Faults in SCSI Disks

Vol.8 Iss.3 - May 1993

An Introduction to OPEN*framework*
The Evolution of the OPEN*framework* Systems Architecture
Creating Potential-for-Change
OPEN*framework* in Action at DEVETIR
Strategic Information Systems planning: A Process to Integrate IT and Business Systems
Describing Systems in the OPEN*framework* Integration Knowledge Base
Multimedia and Standards for Open Information
VME-X: Making VME Open
A New Approach to Cryptographic Facility Design
CHISLE: An Engineer's Tool for Hardware System Design
Distributed Detection of Deadlock

Vol.8 Iss.2 - November 1992

Open Networks - The Key to Global Success
Infrastructure of Corporate Networks in the Nineties
Broadband Networking
FDDI - The High Speed Network of the Nineties
The Evolution of Wireless Networks

Communications Technology for the Retail Environment
RIBA - A Support Environment for Distributed Processing
Information Technology: Support for Law Enforcement Investigations and Intelligence
Standard for Keyboard Layouts - The Origins and Scope of ISO/TEC 9995
ESS - A Solid State Disc System for ICL System for ICL Series 39 Mainframes

Vol.8 Iss.1 - May 1992

Defining CASE Requirements
ICL's ICASE Products
The Engineering Database
CASE Data Integration: The Emerging International Standards
Building Maintainable Knowledge Based Systems
The Architecture of an Open Dictionary
The Use of a Persistent Language in the Implementation of a Process Support System
ALF: A Third Generation Environment for Systems Engineering
MASP/DL: The ALF Language for Process Modelling
The ALF User Interface Management System
A New Notation for Dataflow Specifications

Vol.7 Iss.4 - November 1991

Systems Management: A Challenge for the Nineties - Why now?
The Evolution within ICL of an Architecture for Systems Management
Manageability of a Distributed System
Distribution Management - ICL's Open Approach
Experience of Managing Data Flows in Distributed Computing in Retail Businesses
Generation of Configurations - a Collaborative Venture
Operations Management
OSMC: The Operations Control Manager
The Network Management Domain
An Overview of the Raleigh Object-Oriented Database System
Making a Secure Office System
Architectures of Knowledge Base Machines
The Origins of PERICLES - A common on-line Interface

Vol.7 Iss.3 - May 1991

Introduction to the technical characteristics of ISDN
ISDN in France: Numéris and its market
The Telecoms Scene in Spain
Future Applications of ISDN to Information Technology
A Geographical Information System for Managing the Assets of a Water Company
Using Constraint Logic Programming Techniques in Container Port Planning
Locator - An Application of Knowledge Engineering to ICL's Customer Service
Designing the HCI for a Graphical Knowledge Tree Editor: A Case Study in User-Centred Design
X/OPEN - From Strength to Strength
Architectures of Database Machines
Computer Simulation for the Efficient Development of Silicon Technologies

The use of Ward and Mellor Structured Methodology for the Design of a Complex Real Time System

Vol.7 Iss.2 - November 1990

The SX Node Architecture
SX Design Process
Physical Design Concepts of the SX Mainframe
The Development of Marketing to Design: The Incorporation of Human Factors into Specification and Design
Advances in the Processing and Management of Multimedia Information
An Overview of Multiworks
RICHE-Rèseau d'Information et de Communication Hospitalier Européen (Healthcare Information and Communication Network for Europe)
E.S.F - A European Programme for Evolutionary Introduction of Software Factories
A Spreadsheet with Visible Logic
Intelligent Help - The Results of the EUROHELP Project
How to use Colour in Displays - Coding, Cognition and Comprehension
Eye Movements for A Bidirectional Human Interface
Government IT Infrastructure for the Nineties (GIN): An Introduction to the Programme

Vol.7 Iss.1 - May 1990

Architecture of the DRS6000 (UNICORN) Hardware
DRS6000 (UNICORN) software: an overview
Electromechanical Design of DRS6000 (UNICORN)
The User-System Interface - a challenge for application users and application developers?
The emergence of the separable user interface
SMIS - A Knowledge-Based Interface to Marketing Data
A Conversational Interface to a Constraint-Satisfaction System
SODA: The ICL interface for ODA document access
Human - Human co-operation and the design of co-operative mechanisms
Regulatory Requirements for Security - User Access Control
Standards for secure interfaces to distributed applications
How to Use Colour in Displays - 1. Physiology Physics & Perception

Vol.6 Iss.4 - November 1989

Time to Market in new product development
Time to Market in manufacturing
The VME High Security Option
Security aspects of the fundamental association model
An introduction to public key systems and digital signatures
Security classes and access rights in a distributed system
Building a marketeer's workbench: an expert system applied to the marketing planning process
The Knowledge Crunching Machine at ECRC: a joint R&D project of a high speed Prolog system
Aspects of protection on the Flagship machine: binding, context and environment
ICL Company Research and Development Part 3: The New Range and other developments

Vol.6 Iss.3 - May 1989

Tools, Methods and Theories: a personal view of progress towards Systems Engineering
Systems Integration
An architectural framework for systems
Twenty Years with Support Environments
An Introduction to the IPSE 2.5 Project
The case for CASE
The UK Inland Revenue operational systems
La solution ICL chez Carrefour a Orleans
A Formally-Specified In-Store System for the Retail Sector towards a Geographic Information System
...towards a Geographic Information System
Ingres Physical Design Adviser: a prototype system for advising on the physical design of an Ingres relational database
KANT - a Knowledge Analysis Tool
Pure Logic Language
The 'Design to Product' Alvey Demonstrator

Vol.6 Iss.2 - November 1988

Flexible Manufacturing at ICL's Ashton plant
Knowledge based systems in computer based manufacturing
Open systems architecture for CIM
MAES - An expert system applied to the planning of material supply in computer manufacturing
JIT and IT
Computer Aided Process Planning (CAPP): Experience at Dowty Fuel Systems
Use of integrated electronic mail within databases to control processes
Value engineering - a tool for product cost reduction
ASP: Artwork specifications in Prolog
Elastomer technology for probing high-density printed circuit boards
The effects of back-driving surface mounted digital integrated circuits
Reliability of surface-mounted component soldered joints produced by vapour phase, infrared soldering techniques
Materials evaluation
On the human side of technology

Vol.6 Iss.1 - May 1988

ICL Series 39 support process
The ICL systems support centre organisation
ICL Services Product Centre
Knowledge engineering as an aid to the system service desks
Logic analysers for system problem solving
Repair - past and future
OSI migration
A Network to Support Application Software Development
Universal Communications Cabling: A Building Utility

Collecting and generalising knowledge descriptions from task analysis data
The architecture of an automated Quality Management System
ICL Company Research and Development Part 2: Mergers and Mainframes, 1959-1968

Vol.5 Iss.4 - November 1987

Open Distributed Processing
The Advanced Network Systems Architecture project
Community management for the ICL networked production line
The X/OPEN Group and the Common Applications Environment
Security in distributed information systems: needs, problems and solutions
Cryptographic file storage
Standards and office information
Introducing ODA
The Technical and Office Protocols - TOP
X400 - international information distribution
A general purpose natural language interface: design and application as a database front end
DAP-Ada: Ada facilities for SIMD architectures
Quick language implementation

Vol.5 Iss.3 - May 1987

What is Fifth Generation? - the scope of the ICL programme
The Alvey DHSS Large Demonstrator Project
PARAMEDICL: a computer-aided medical diagnosis system for parallel architectures
S39XC - a configurer for Series 39 mainframe systems
The application of knowledge-based systems to computer capacity management
On knowledge bases at ECRC
Logic languages and relational databases: the design and implementation of Educe
The semantic aspects of MMI
Language overview
PISA - a Persistent Information Space Architecture
Software development using functional programming languages
Dactl: a computational model and compiler target language based on graph reduction
Designing system software for parallel declarative systems
Flagship computational models and machine architecture
Flagship hardware and implementation
GRIP: a parallel graph-reduction machine

Vol.5 Iss.2 - November 1986

The Management into the 1990s Research Programme
Managing strategic ideas: the role of the computer
A study of interactive computing at top management levels
A management support environment
Managing change and gaining corporate commitment
An approach to information technology planning
Preparing and organising for IPSE
Global Language for Distributed Data Integration

The design of distributed secure logical machines
Mathematical logic in the large practical world
The ICL DRS300 management graphics system
Performance of OSLAN local area network
Experience with programming parallel signal-processing algorithms in Fortran 8X

Vol.5 Iss.1 - May 1986

ICL company research and development, 1904-1959
Innovation in computational architecture and design
REMIT: a natural language paraphraser for relational query expressions
Natural language database enquiry
The *me too* method of software design
Formal specification - a simple example
The effects of inspections on software quality and productivity
Recent developments in image data compression for digital facsimile
Message structure as a determinant of message processing system structure

Vol.4 Iss.4 - November 1985

History of the ICL content-addressable file store, (CAFS)
History of the CAFS relational software
The CAFS system today and tomorrow
Development of the CAFS-ISP controller product for Series 29 and 39 systems
CAFS-ISP: issues for the applications designer
Using secondary indexes for large CAFS databases
Creating an end-user CAFS service
Textmaster - a document retrieval system using CAFS-ISP
CAFS and text: the view from academia
Secrets of the sky: the IRAS data at Queen Mary College
CAFS file-correlation unit

Vol.4 Iss.3 - May 1985

Overview of the ICL Series 39 Level 30 system
VME nodal architecture: a model for the realisation of a distributed system concept
Processing node of the ICL Series 39 Level 30 system
Input/output controller and local area networks of the ICL Series 39 Level 30 system
The store of the ICL Series 39 Level 30 system
The high-speed peripheral controller for the Series 39 system
Development of 8000-gate CMOS gate arrays for the ICL Level 30 system
Development route for the C8K 8000-gate CMOS array
Design automation tools used in the development of the ICL Series 39 Level 30 system
Design and manufacture of the cabinet for the ICL Series 39 Level 30 system
Manufacturing the level 30 system I Mercury: an advanced production line
Manufacturing the Level 30 system II Merlin: an advanced printed circuit board manufacturing system

Manufacturing the Level 30 system III The test system

Vol.4 Iss.2 - November 1984

Modelling a multi-processor designed for telecommunication systems control
Tracking of LSI chips and printed circuit boards using the ICL Distributed Array Processor
Sorting on DAP
User functions for the generation and distribution of encipherment keys
Analysis of software failure data(1): adaptation of the Littlewood stochastic reliability growth model for coarse data
Towards a formal specification of the ICL Data Dictionary

Vol.4 Iss.1 - May 1984

The ICL University Research Council
The Atlas 10 computer
Towards better specifications
Solution of the global element equations on the ICL DAP
Quality model of system design and integration
Software cost models
Program history records: a system of software data collection and analysis

Vol.3 Iss.4 - November 1983

Expert system in heavy industry: an application of ICLX in a British Steel Corporation works
Dragon: the development of an expert sizing system
The logic language PROLOG-M in database technology and intelligent knowledge-based systems
QPROC: a natural language database enquiry system implemented in PROLOG
Modelling software support

Vol.3 Iss.3 - May 1983

IPA networking architecture
IPA data interchange and networking facilities
The IPA telecommunications function
IPA community management
MACROLAN: a high-performance network
Specification in CSP language of the ECMA-72 Class 4 transport protocol
Evolution of switched telecommunication networks
DAP in action

Vol.3 Iss.2 - November 1982

The advance of Information Technology
Computing for the needs of development in the smallholder sector
The PERQ workstation and the distributed computing environment
Some techniques for handling encipherment keys
The use of COBOL for scientific data processing

Recognition of hand-written characters using the DAP
Hardware design faults: a classification and some measurements

Vol.3 Iss.1 - May 1982

Software of the ICL System 25
Security in a large general-purpose operating system: ICL's approach in VME/2900
Systems evolution dynamics of VME/B
Software aspects of the Exeter Community Health Services Computer Project
Associative data management system
Evaluating manufacturing testing strategies

Vol.2 Iss.4 - November 1981

Architecture of the ICL System 25
Designing for the X25 telecommunications standard
Viewdata and the ICL Bulletin System
Development philosophy and fundamental processing concepts of the ICL Rapid Application Development System RADS
A moving-mesh plasma equilibrium problem on the ICL Distributed Array Processor

Vol.2 Iss.3 - May 1981

A dynamic database for econometric modelling
Personnel on CAFS: a case study
Giving the computer a voice
Data integrity and the implications for back-up
Applications of the ICL Distributed Array Processor to econometric computations
A high-level logic design system
Measures of programming complexity

Vol.2 Iss.2 - November 1980

The ICL Information Processing Architecture, IPA
VME/B: a model for the realisation of a total system concept
Birds, Bs and CRTs
Solution of elliptic partial differential equations on the ICL Distributed Array Processor
Data routing and transpositions in processor arrays
A Bayesian approach to test modelling

Vol.2 Iss.1 - May 1980

Security and privacy of data held in computers
CADES - software engineering in practice
ME29 Initial Program Load: an exercise in defensive programming
Project Little - an experimental ultra-reliable system
Flow of instructions through a pipelined processor
Towards an 'expert' diagnostic system
Using Open System Interconnection standards

Vol.1 Iss.3 - November 1979

Meteosat 1: Europe's first meteorological satellite
An analysis of checkpointing
Statistical and related systems
Structured programming techniques in interrupt-driven routines
The content addressable file store - CAFS
Computing in the humanities
The data dictionary system in analysis and design

Vol.1 Iss.2 - May 1979

Computers in support of agriculture in developing countries
Software and algorithms for the Distributed Array Processor
Hardware monitoring on the 2900 range
Network models of system performance
Advanced technology in printing: the laser printer
The new frontier: three essays on job control

Vol.1 Iss.1 - November 1978

The origins of the 2900 series
Sizing computer systems and workloads
Wind of Change
Standards for open-network operation
Distributed computing in business data processing
A general model for integrity control

To order back issues

Contact

Sheila Cox

Research and Advanced Technology,

ICL, Lovelace Road, Bracknell, Berks., RG12 8SN

Telephone +44 (0)1344 472900

Fax +44 (0)1344 472700

Email: S.D.Cox.bra0114@oasis.icl.co.uk

or

The Editor, V.A.J. Maller

Telephone +44 (0)1438 833514

Email: V.A.J.Maller@ste0418.wins.icl.co.uk

ICL Systems Journal

Guidance for Authors

1. Content

The ICL Systems Journal has an international circulation. It publishes high standard papers that have some relevance to ICL's business. It is aimed at the general technical community and in particular at ICL's users and customers. It is intended for readers who have an interest in the information technology field in general but who may not be informed on the aspect covered by a particular paper. To be acceptable, papers on more specialised aspects of design or application must include some suitable introductory material or reference.

The Journal will usually not reprint papers already published but this does not necessarily exclude papers presented at conferences. It is not necessary for the material to be entirely new or original. Papers will not reveal matter relating to unannounced products of any of the ICL Group companies.

Letters to the Editor and book reviews may also be published.

2. Authors

Within the framework defined in paragraph 1, the Editor will be happy to consider a paper by any author or group of authors, whether or not an employee of a company in the ICL Group. All papers are judged on their merit, irrespective of origin.

3. Length

There is no fixed upper or lower limit, but a useful working range is 4000-6000 words; it may be difficult to accommodate a long paper in a particular issue. Authors should always keep brevity in mind but should not sacrifice necessary fullness of explanation to this.

4. Abstract

All papers should have an Abstract of not more than 200 words, suitable for the various abstracting journals to use without alteration.

5. Presentation

5.1 Printed (typed) copy

Two copies of the manuscript, typed 1_/2 line spacing on one side only of A4 paper, with right and left margins of at least 2.5cms, and the pages numbered in sequence, should be sent to the Editor. Particular care should be taken to ensure that mathematical symbols and expressions, and any special characters such as Greek letters, are clear. Any detailed mathematical treatment should be put in an Appendix so that only essential results need be referred to in the text.

5.2 Disk version

Authors are requested to submit a magnetic disk version of their copy in addition to the manuscript. All artwork and diagrams to be supplied in their original source format. The Editor will be glad to provide detailed advice on the format of the text on the disk.

5.3 Diagrams

Line diagrams will, if necessary, be redrawn and professionally lettered for publication, so it is essential that they are clear. Axes of graphs should be labelled with the relevant variables and, where this is desirable, marked off with their values. All diagrams should have a caption and be numbered for reference in the text and the text marked to show where each should be placed - e.g. "Figure 5 here". Authors should check that all diagrams are actually referred to in the text and that all diagrams referred to are supplied. Since diagrams are always separated from their text in the production process, these should be presented each on a separate sheet and, most important, each sheet must carry the author's name and the title of the paper. The diagram captions and numbers should be listed on a separate sheet which also should give the author's name and the title of the paper.

5.4 Tables

As with diagrams, these should all be given captions and reference numbers; adequate row and column headings should be given, also the relevant units for all the quantities tabulated.

5.5 References

Authors are asked to use the Author/Date system, in which the author(s) and the date of the publication are given in the text, and all the references are listed in alphabetical order of author at the end. e.g. in the text: "...further details are given in [Henderson, 1986]" with the corresponding entry in the reference list:

HENDERSON, P. Functional Programming Formal Specification and Rapid Prototyping. IEEE Trans. on Software Engineering SE*12, 2, 241-250, 1986.

Where there are more than two authors it is usual to give the text reference as "[X et al ...]".

Authors should check that all text references are listed; references to works not quoted in the text should be listed under a heading such as Bibliography or Further reading.

5.6 Style

A note is available from the Editor summarising the main points of style - punctuation, spelling, use of initials and acronyms etc. - preferred for Journal papers.

6. Referees

The Editor may refer papers to independent referees for comment. If the referee recommends revisions to the draft, the author will be asked to make those revisions. Referees are anonymous. Minor editorial corrections, as for example to conform to the Journal's general style for spelling or notation, will be made by the Editor.

7. Proofs, Offprints

Printed proofs are sent to authors for correction before publication. Authors receive either a hard copy master of their paper or a Word for Windows version in either V2 or V6 on magnetic media.

8. Copyright

Copyright of papers published in the ICL Systems Journal rests with ICL unless specifically agreed otherwise before publication. Publications may be reproduced with the Editor's permission, which will normally be granted, and with due acknowledgement.

ICL Research & Advanced Technology
Lovelace Road
Bracknell
Berkshire RG12 8SN
UK