The **Systems** *Journal*

ICL

Special Feature — ICL's Exchange Server

Volume 13 Issue 2 *Spring 1999 Edition*

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 Group Technical Directorate, ICL, Kidsgrove.

1999 subscription rates (including postage & packing):

	UK and Europe	Rest of World
Annual subscription	£72	$120
Single issues	£43	$72

ISSN 1364-310X

ICL Systems Journal

Volume 13 Issue 2

Contents

Editorial

This issue has papers drawn from a wide spectrum of activities: Project THOR, which draws upon the successful partnership with Microsoft; *Monterey*, which complements the paper in the last issue on the very large commercial Internet service, *beeb.com*, developed by ICL for the BBC; a paper on the National Grid for Learning, a potential new market for ICL and one in which success is dependent on close interaction with Government public policy; Cochise, which enables existing transaction processing systems to operate within the web browser world of the Internet. There is also a paper on system management, which continues the series on the Millennium product range.

June, 1998 was the fiftieth anniversary of the first running of an electronically stored computer program at the University of Manchester. The results of this pioneering work were exploited by Ferranti in the production and sale of some of the World's first commercial computers. The Ferranti computer department later became part of ICL and its legacy provided much of the engineering backbone for several generations of ICL systems. As part of the celebrations in 1998, a replica of the first machine, "Baby", was built by a team led by Chris Burton, one of ICL's retired engineers, who then very kindly accepted an invitation to write a paper on "Baby" and its legacy.

In this issue, there are the obituaries of two engineers who both made immense individual contributions to ICL product engineering. Sadly, they both died suddenly a few weeks after their respective retirements.

V.A.J. Maller

Front cover: High Performance Microsoft Exchange. See the paper, "Project THOR" in this issue.

Project THOR

Derek Ashcroft

ICL High Performance Systems, Manchester, UK.

Abstract

Project THOR was initiated as the result of an idea by Stephen Tickhill from Sales in ICL High Performance Systems (HPS) who sold the concept to a major ICL customer. His idea was to build a centralised Microsoft Exchange system to provide an integrated office applications and email service for fifty thousand users, something that had never been done before anywhere in the world.

Traditionally, Exchange is implemented using a number of small, distributed servers each supporting about 2,000 users. Resilience is achieved by using a standby server. For this customer, this approach would have meant over 50 servers around the country being installed, configured, backed-up, administered and maintained. Clearly, a single centralised set-up has major benefits in terms of cost, security and manageability.

Naturally, the customer wanted to see a suitable demonstration of the concept prior to agreeing to its purchase. After some discussion it was agreed that a proof-of-concept demonstration of 12,000 users showing full resilience, and backup/restore capability would be acceptable.

The Objectives

- Demonstrate a system running 12,000 Microsoft Exchange users (***Loadsim*** 'medium' profile)
- Resilience to hardware failure, including complete server failure
- Improved back-up and recovery capability
- Reduced management overhead
- Automated management.

The Challenges

- Provide fully resilient system
- 6,000 users, 300 GB database per server
- Recover database within half a day from ***BCV***
- Recover database from tape within one day
- Automated back-up, service management and resilience
- Integrated system management.

The Approach

After researching other Exchange demonstrations we agreed that the test environment would be Microsoft's Loadsim with Microsoft Outlook as the Exchange client (customer's choice).

Loadsim is a simulation program, available from the Microsoft website, that can be tailored to simulate any user profile.

It was noticeable that all the previous large-scale Exchange demonstrations shown on the Microsoft website were on a single server, thus eliminating the interserver traffic. Our demonstration needed to use at least two servers so as to stimulate interserver traffic using MTA (Message Transfer Agent).

The main components of the solution are:

- **Xtraservers** from HPS, (8 Pentium Pro processors plus 2 GB memory each)
- ***Symmetrix*** disk system from EMC
- StorageTek ***Panther Library*** using ***DLT7000*** tape drives
- Forty PCs running loadsim to simulate 12,000 users (300 users per PC).

The Configuration: Clients

The first part of the exercise was to define the hardware configuration, taking into account the expected throughput of the system and the test environment.

The team "borrowed" forty PCs from the internal ICL office equipment "refreshment" services. These machines (233 Mhz, 96 MB) would support three hundred users each. They were arranged on specially constructed racking, thanks

to the West Gorton workshop. We purchased two 24 way 100 Mbit 3Com intelligent switches to connect the PCs to the servers to ensure that there were no network delays affecting the Exchange performance.

The Configuration: Servers

The number of *Xtra*servers we configured to support 12,000 users was a compromise between maximising the number of users per server (to reduce the number of servers needed) and minimising the database size per server in order to reduce the back-up time. The mailbox size per user as requested by the customer was 50 MB giving a total database size of 600 GB for 12,000 users. After conducting a number of experiments we decided to configure 6,000 users per server, giving two servers each with a database of 300 GB. The cache size for the server was initially 3 GB but this was changed to 2 GB at a later stage without adversely affecting performance.

After discussions with Bill Clark of EMC we configured three SCSI connections per *Xtra*server to the Symmetrix for database access, plus a further single SCSI interface for the *Xtra*server system and boot partition which was also relocated to the Symmetrix.

The Symmetrix type 3400 is an intelligent disk storage system which allows shared access from multiple independent hosts. There are 96 disks, 18 GB each, with up to 16 SCSI interfaces and 4 GB of memory.

The Configuration: Exchange

Microsoft Exchange has six main software components:

- Private information store (priv.edb) and Public information store (pub.edb) which contain the mail boxes and public folders respectively
- MTA which controls messages between servers
- Directory service which provides the addresses of recipients
- Transaction logs for both the Information store and the Directory services.

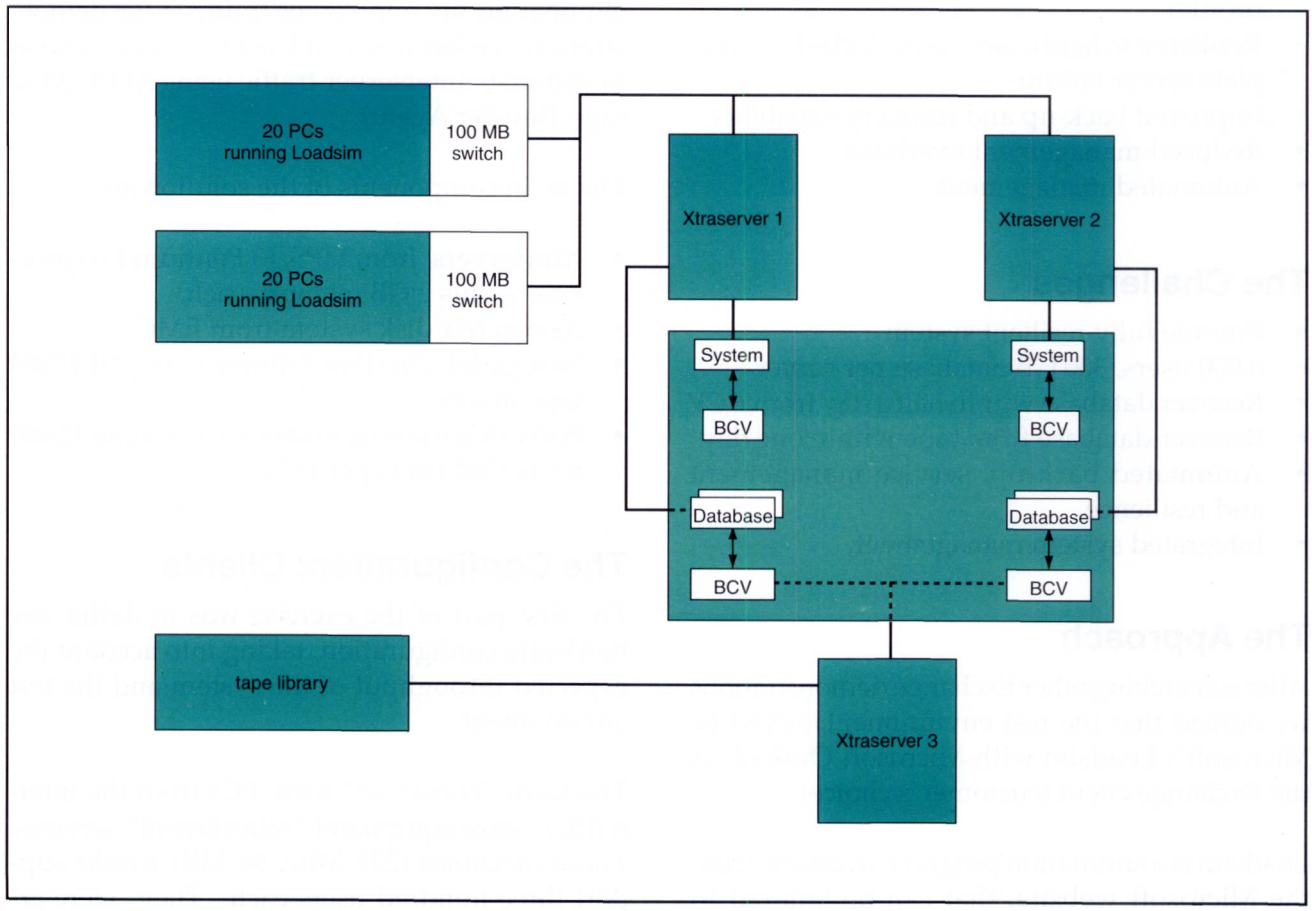


Figure 1: Hardware Configuration

The THOR system included all these elements with the exception that no public folders were configured within pub.edb.

Microsoft advise that the transaction logs should be placed on a separate drive for optimal performance. We arranged the virtual (NT) disks as two drives for each server (X: and V: for one server, Y: and W: for the other). The transaction logs were placed on the V: and W: drives, which were striped across 6 disks within the Symmetrix system. The rest of the Exchange components were placed on the X: and Y: drives, which were striped across 18 disks each.

The Security and Recovery Strategy

The standard method of database back-up for Exchange is an on-line back-up overnight. The on-line backup speed to a DLT tape is about 20 GB per hour with no concurrent workload.

With a database of 300 GB the back-up would have taken at least 15 hours and would have been running alongside users using the mail system throughout this period. Equally significant is the time to restore and recover the database, which is even longer due to the need to roll-forward transaction logs.

This clearly did not meet our customer's requirements, so we sat down to see if we could devise an approach which would be both faster and have less impact on the user service.

The Symmetrix system has a unique feature called the Business Continuance Volume (BCV). This is a 2nd mirror which can be joined to or detached from the data and 1st mirror under software control.

We used this feature to create a snapshot copy of the Exchange Database which could then be backed-up by a separate Xtraserver to the StorageTek tape library, thus eliminating any effect of back-up on the Exchange service.

Recovery can normally be based on the BCV snapshot, which makes it orders of magnitude faster than restoring from tape.

The Simulation

The loadsim workload simulator enables tailored Exchange solutions to be demonstrated and performance figures produced. Loadsim also has three standard user profiles—light, medium and heavy.

Our customer selected the medium profile as closest to his requirement, though recognising that predicting just how users will exploit the many sophisticated features of a product such as Exchange is not an exact science. The users will often have several different ways of performing a task available to them, each having different performance implications. A significant margin of safety was therefore required.

Loadsim simulation is based on an eight-hour run during which time each user performs a number of predetermined activities dependent on the profile selected. The activities for medium profile are:

- Send 4 items of mail per day to 3 recipients
- Include a distribution list for 30% of mail
- Process Inbox 12 times
- Browse mail 15 times
- Schedule 5 times
- Number of messages in Inbox 4
- Number of messages in deleted items folder 1
- Number of new folders (5 messages per new folder) 40

Performance data is obtained by running the system for at least an 8-hour period, collecting the data from all the simulation PCs and calculating the 95-percentile response time (i.e. the average response time for 95% of actions).

The normal guideline is that the response time should be below 1 second. This indicates that the system is performing well and is not struggling to cope with the workload.

Although we used the medium profile we did add a 750,000 byte attachment to 12% of the mail messages for one of the servers so that this would increase the database size to around 300 GB and demonstrate that size really does not matter.

Resilience

With a centralised, large server approach, resilience of the system to hardware breaks is more important than ever, and was a vital element of the demonstration.

The ***Xtra*server** has excellent resilience features: resilient power and fans, ***ECC*** memory, ***UPS***, and extensive hamming. Nonetheless, in a mission-critical system, contingency for server failure must be made: a fast replacement policy is essential, for example, clustering.

The Symmetrix system also has excellent features such as mirrored disks, resilient power supplies/fans, hamming recovery of cache and "phone home" when resilient components need replacing (most components may be "hot" replaced).

While clustering can be a good solution for a single server environment, it does involve a cost overhead. A "live" and a standby server is required for each Exchange service. This doubles the number of servers required. The 50,000 user Exchange solution used eight Exchange servers and, including a standby for all eight servers, was somewhat too expensive.

After a number of design reviews we concluded that the solution was to separate the system disk from the server and to find a mechanism for moving the system disk from a failed server to a single, shared standby server.

We examined a number of ways of achieving this, from hot disk replacement to Fibre channel switches and finally decided to use the Symmetrix BCV mechanism after an explanation from Bill Clark of its inner working.

BCVs as stated earlier are a 2nd mirror that can be controlled by software. The software includes a command line interface with a number of commands as listed below:

- **Establish**—establishes the 2nd mirror and hence copies data from the main data to the 2nd mirror (BCV)
- **Incremental Establish**—As above but the system has remembered the disk tracks that have changed and only overwrites these (much faster)
- **Split**—The BCV is split away from the main plex (the system needs to be quiescent when this command is issued (i.e. Exchange has to be stopped)
- **Restore**—The 2nd mirror is copied to the main data disk. This command is used to recover a corrupt database
- **Incremental restore**—As restore but overwrites only the tracks that have been modified since the last BCV split.

Use of BCVs and the Command Line Interface for database restore and ***N+1*** resilience enabled us to develop automated procedures which are essential to the operation and management of a live service. The high degree of automation we were able to achieve was also a key factor when we obtained support for the solution from the Microsoft Exchange development team in Redmond.

A further advantage of BCVs is that, when restoring a disk, the Symmetrix system is intelligent enough to enable access to the data **at the same time** as it performs the restore operation in the background. It uses pointers to direct application access to either the main plex or the BCV depending on how far the restoration has got. This effectively means that restoration is instant, since one does not have to wait for it to be complete.

The N+1 process uses the **restore** operation. The system disks for both the Exchange servers reside in the Symmetrix and include a BCV. When a server failure was simulated, the BCV of the 'failed' system was **restored** to the system disk for the N+1 server. This copies the contents of the system disk of the 'failed' server to the system disk of the N+1 server. As explained earlier, as soon as the restore starts, the copied system becomes visible to the N+1 server which then takes on the personality of the failed server.

The failed server when repaired becomes the new N+1 server.

Figures 2, 3 and 4 illustrate the N+1 Failover process.

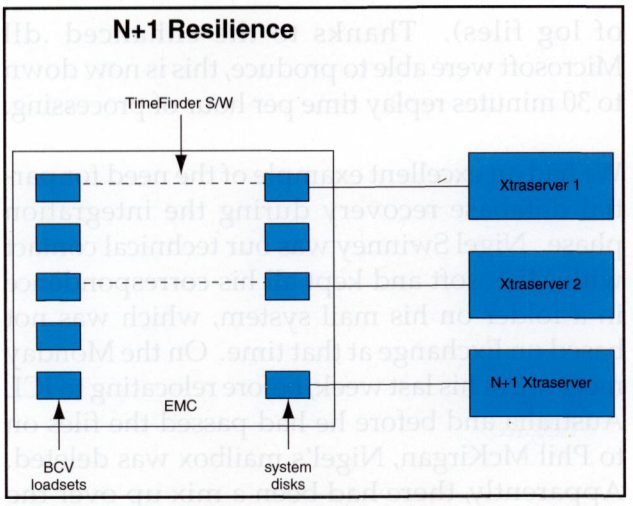


Figure 2: N + 1 Failover Process (a)

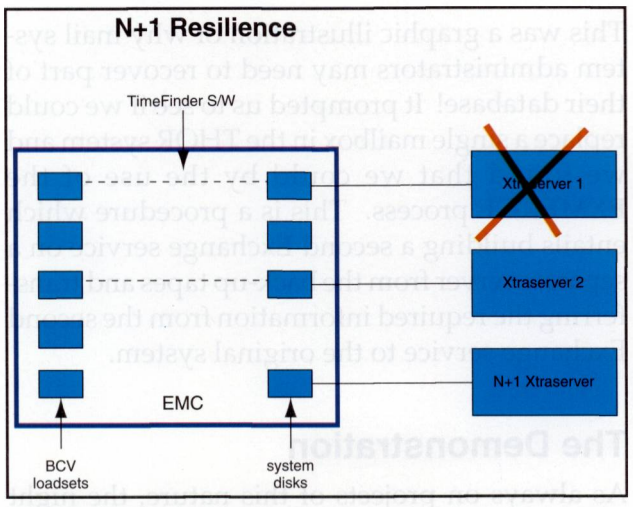


Figure 3: N + 1 Failover Process (b)

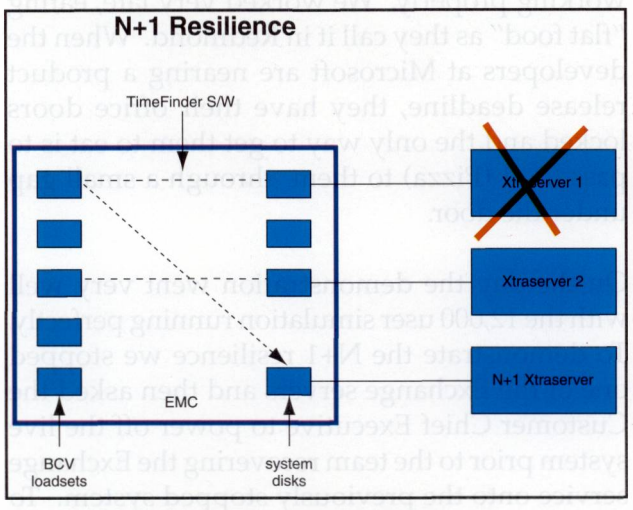


Figure 4: N + 1 Failover Process (c)

Backup and Recovery

Backup

Back-up is achieved by splitting the BCV from the main database, making it accessible by the archive server and backing up this snapshot to the shared StorageTek library.

This sounds simple but was in reality far from simple. When the BCV is connected to the main data disk it is considered part of the main disk and cannot be accessed by the archive server. When the BCV is split we need to enable the archive server to be able to access the BCV. At first we could only achieve this by re-booting the archive server but we subsequently removed the need to reboot by using a utility called "VCTL" which remounts the BCV disks attached to the archive servers after the splitting process.

It is also imperative that the backup software does not alter the disk signatures on the BCVs because this would alter the signature of the main system disks when a restore is conducted. To ensure the signatures are not altered we had to use a backup application running under NT (As Exchange), because NT guarantees not to alter any disk signatures generated by a fellow NT system.

The overall aim in designing our backup process was to reduce the user impact of the process and enable faster recovery, without losing any of the facilities available with the standard on-line backup utility. On-line backup performs an integrity check (checksum) of the data prior to backup. To achieve an equivalent level of integrity checking in our solution, we worked together with Microsoft to implement an off-line checksum check, which is run against the snapshot copy of the database prior to backing up to tape.

As mentioned earlier, a great deal of time and effort was expended in automating the backup procedures, and in integrating them with the control of the Exchange service, the Maestro batch scheduler and the Networker backup software. The result is a system which can operate completely autonomously, and which can be monitored using the standard MS Windows NT Event Viewer tool.

Recovery

The back-up approach also has major benefits for database recovery. The most remarkable of these is the speed of the database restore—10 minutes compared with the normal 5 hours or so. This is due to the fact that the latest back-up (the one which is normally needed) is retained on disk (on the BCV) and so no tapes are needed. Further details are given in the section, 'Results'.

The recovery process was designed with help from Microsoft/EMC and extended and improved by an extensive programme of testing and validation carried out both at our Manchester laboratories and at the Microsoft campus in Redmond, Washington State. This continued until we had a comprehensive working system.

The recovery process includes deleting the checkpoint file (identifies which transaction logs have been applied to the database), and applying the Information Store integrity check (isinteg patch). NB: these are non-standard activities which must only be considered in the most stringently controlled environment. Misuse of these facilities can cause serious damage to your database. Please do not try this at home!

During the integration period we were in constant contact with Microsoft as we discovered new issues, and Microsoft provided us with a number of hot fixes. These were associated with increased buffer settings, increased counters in the Information store and the JET database (including a set with the intriguing title of the "Squeaky Lobster" mod), optimising the settings of Exchange parameters, reducing the log replay times (from about 90 seconds to 30 seconds), integrity checking, transaction log checking, audit trail, event logging, etc.. Many of the mods identified are now included in Exchange service pack 2.

In the event of a database recovery being needed, the latest back-up, usually taken overnight, needs to be restored from back-up, and then modified by replaying the stored transaction log files.

We found that 6000 medium-profile users generated about 70 transaction log files per hour, hence the standard log replay time of 90 seconds would have meant about 1.5 hours to recover every hour of processing that the data base had run since the last backup (e.g. 12 hours to replay 8 hours-worth of log files). Thanks to the enhanced .dll Microsoft were able to produce, this is now down to 30 minutes replay time per hour of processing.

We had an excellent example of the need for partial database recovery during the integration phase. Nigel Swinney was our technical contact with Microsoft and kept all his correspondence in a folder on his mail system, which was not based on Exchange at that time. On the Monday morning of his last week, before relocating to ICL Australia and before he had passed the files on to Phil McKirgan, Nigel's mailbox was deleted. Apparently, there had been a mix up over the termination date and the mailbox was never recovered.

This was a graphic illustration of why mail system administrators may need to recover part of their database! It prompted us to see if we could replace a single mailbox in the THOR system and we found that we could by the use of the **EXMERGE** process. This is a procedure which entails building a second Exchange service on a separate server from the back-up tapes and transferring the required information from the second Exchange service to the original system.

The Demonstration

As always on projects of this nature, the night before the demonstration to the customer we were making absolutely sure everything was working properly. We worked very late, eating "flat food" as they call it in Redmond. When the developers at Microsoft are nearing a product release deadline, they have their office doors locked and the only way to get them to eat is to pass food (Pizza) to them through a small gap under the door.

On the day the demonstration went very well with the 12,000 user simulation running perfectly. To demonstrate the N+1 resilience we stopped one of the Exchange servers and then asked the Customer Chief Executive to power off the live system prior to the team recovering the Exchange service onto the previously stopped system. To our immense relief the process worked flawlessly, and the customer duly bought the solution.

The picture in Figure 5 shows the rehearsal on the day of the demonstration.

ITEM	THOR Metric
95th percentile response time	0.83 seconds
Total I/O's per second	800 (peaked at 1000 during login)
Average CPU usage	40–45%
Exchange recovery time:	
Restore from BCV	10 minutes
Log recovery	30 seconds (average) per log file
N +1 recovery time	Under 30 minutes

Table 1: Exchange Performance for 12,000 users

Figure 5: Demonstration Rehearsal

The Results

The performance figures from the Exchange run for 12K users on two Xtraservers are shown in Table 1.

Roadshow

HPS marketing then decided they wanted a roadshow demonstration of THOR and a forty-foot exhibition trailer, accompanied by a fourteen tonne support lorry and two transit vans, started a tour of the UK and mainland Europe. More than twenty venues hosted the Roadshow including Microsoft offices in Reading, Stockholm and Paris. Other venues included Belfast, Copenhagen, Dublin, Edinburgh and Maarssen.

The Roadshow events included an hour-long presentation featuring updates on the Microsoft Global Alliance, ***e*.workplace**, ***Infra*Care**, ICLNet (the ATM network) and, of course, ***Exchange***.

Speakers from both Microsoft and EMC also supported the customer sessions.

Following the presentation, attendees were able to see **the exchange** demonstration of a 'live' 12,000-user centralised Microsoft Exchange solution running on the ICL ***Trimetra Xtra*server** in the specially adapted exhibition trailer.

More than 3,500 people attended the Roadshow including ICL customers, client managers and staff as well as customers from our partner organisations, including Microsoft and EMC. Customer attendees included BT, Marks & Spencer, AA, British Gas and EDS. In addition, in excess of 400 Microsoft staff attended at the Microsoft venues, including a number of business managers, liaison managers and Exchange product managers.

Customer Configuration

When the system was scaled to 50,000 users we used the largest Symmetrix model—the 3700. This supports up to 128 disks each of 46 GB (unformatted), 32 SCSI connections and up to 8 GB of memory. We connected four Exchange ***Xtra*servers** to one 3700 Symmetrix system with **6250 users per *Xtra*server**. One additional ***Xtra*server** acted as the N+1 server plus a further 2 ***Xtra*servers** for backup from the BCVs. Using two such configurations, 50,000 users could be supported.

The resilience and recovery achieved in THOR are compared with a typical Exchange system in Table 2 and a diagram of the 50,000 user configuration is shown in Figure 6.

Item	Standard Solution	THOR Solution
Size of database	50–100 Gb maximum	300 Gb
Restore from latest backup	5 hours plus	10 minutes
Transaction Log recovery (average per log file)	2 minutes	30 seconds
Disc failure	5 hours plus	No action needed
Failed Server recovery	8 hours	Under 30 minutes
Backup speed	On line max 20Gb/hr	Off line max 30 Gb/hr

Table 2: Comparison between THOR and a standard Exchange system

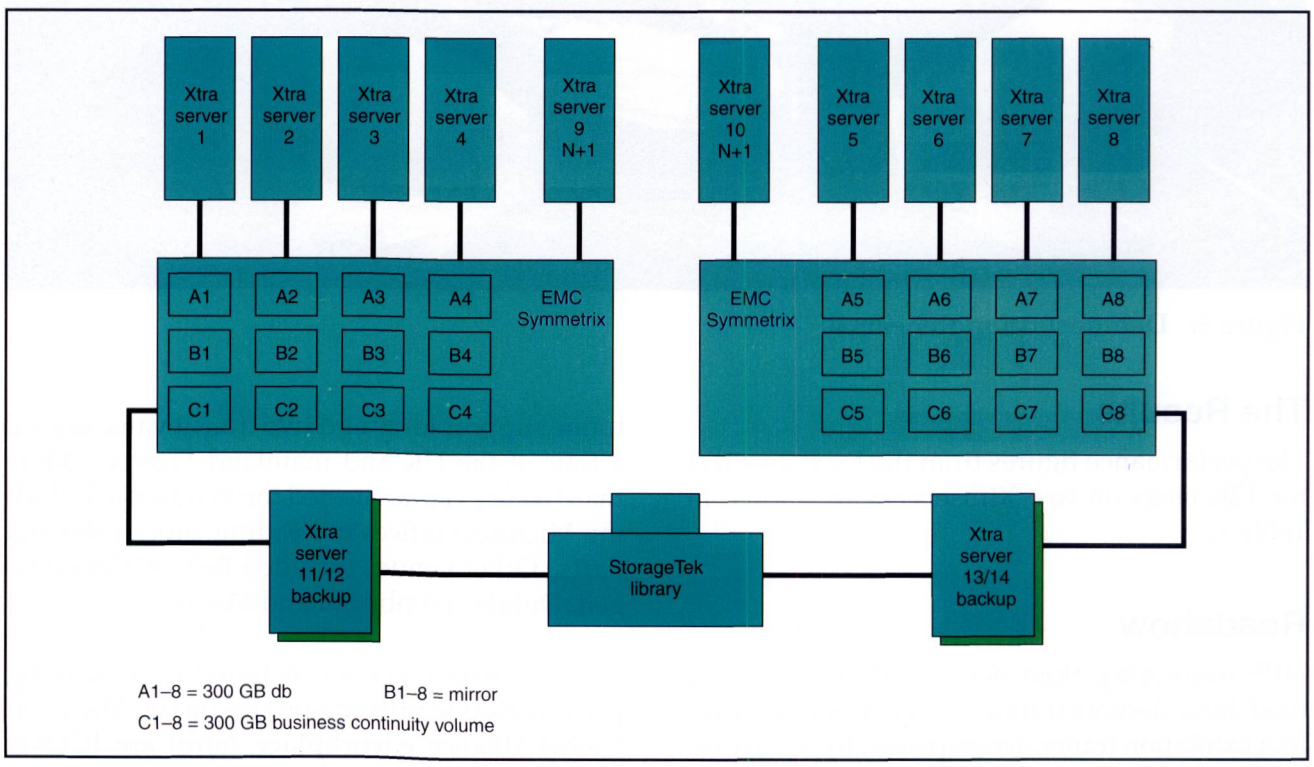


Figure 6: Exchange 50,000 user configuration

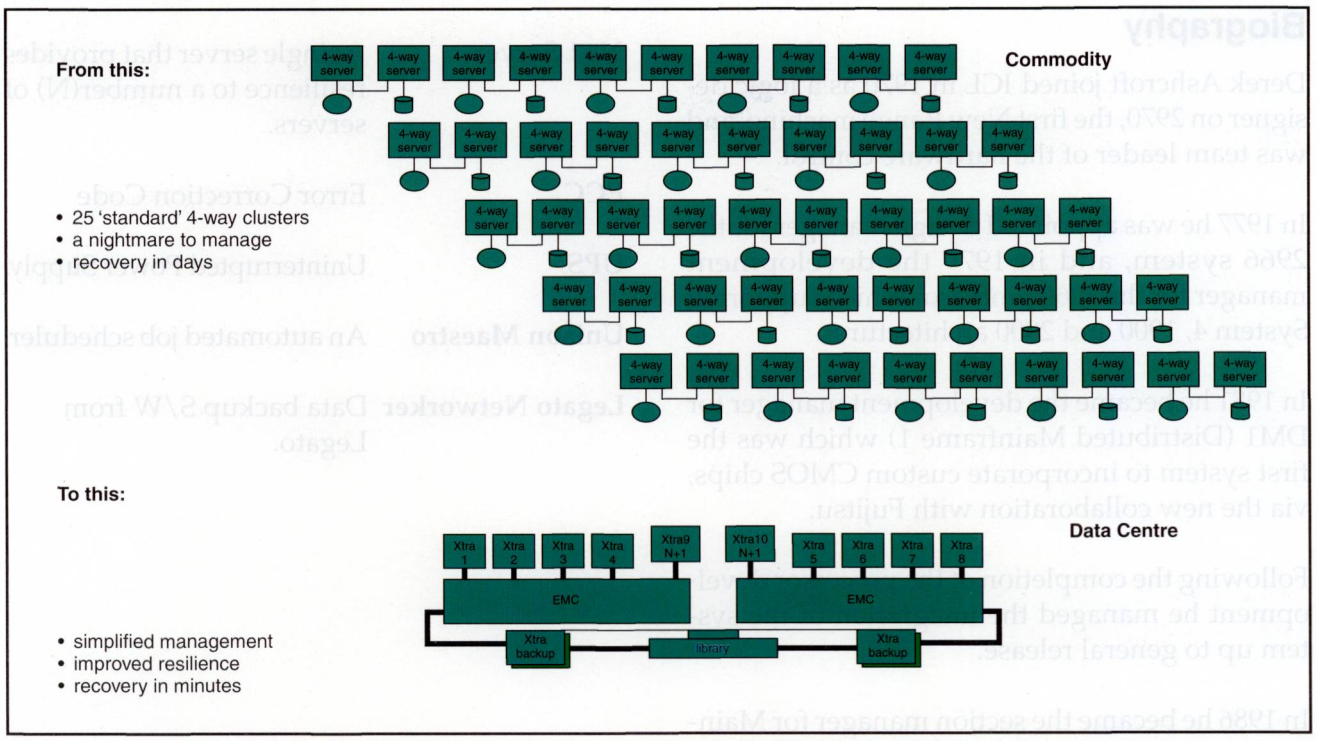


Figure 7: Comparison of centralised and decentralised solutions

Conclusions

This project was designed to demonstrate the viability of large-scale, centralised Exchange systems. We believe it has done that convincingly.

It met all its objectives—economies of scale, resilience, improved back-up and recovery, reduced cost of ownership, simplified management and automation.

The demonstration showed 12,000 users with subsecond response times and proved the viability of centralised systems for more than 50,000 users.

It confirmed that Exchange databases of 300 Gigabytes can be effectively managed by the use of an innovative back-up and recovery approach.

The system integrates Unison Maestro for Automation, Legato Networker for tape management and BMC Patrol for event management.

Figure 7 illustrates the centralised solution as compared to the traditional solution and demonstrates very graphically the benefits of the THOR project approach.

Following the THOR demonstration we have simulated a 25k and 50k centralised Exchange system (i.e. one organisation, one site, with 4(25k) and 8(50k) servers.

The 50k user simulation used 160 PCs each 350 Mhz with 128 Mb store eight servers and two 3700 symmetrix disk systems.

Acknowledgements

THOR was very much a team effort and the author would like to acknowledge the contributions from all the team members, particularly Bill Clark from EMC, Peter Nilsson from Microsoft, and the ICL team of Alan Hirst, Steve Rowe and Derek Ashcroft from High Performance Systems (HPS) development, Phil McKirgan and Nigel Swinney from HPS Sales support, Geoff Whitaker from the NT Centre of Excellence and Chris Weber from the Solution Centre.

The author wishes to give his special thanks to Phil McKirgan for his help and advice in preparing this paper.

Biography

Derek Ashcroft joined ICL in 1970 as a logic designer on 2970, the first New Range machine, and was team leader of the hardware control.

In 1977 he was appointed design manager for the 2966 system, and in 1979 the development manager for the 2966 processor which supported System 4, 1900 and 2900 architectures.

In 1981 he became the development manager for DM1 (Distributed Mainframe 1) which was the first system to incorporate custom CMOS chips, via the new collaboration with Fujitsu.

Following the completion of the processor development he managed the integration of the system up to general release.

In 1986 he became the section manager for Mainframe peripheral development for the next nine years and then in 1995, following the creation of High Performance Systems, he became a Project Manager.

Glossary

Loadsim	Microsofts Exchange client simulator.
Symmetrix	Name given to the range of EMC disk farms) 3300 (64 - 18 GB Disks), 3400 (64 -18 GB disks) and 3700 (124 - 46 GB disks).
Panther Library	Automated tape library from StorageTek.
Xtraserver	Intel based hardware processor from ICL's HPS.
SCSI	Small Computer Systems interface.
DLT7000	Digital Linear Tape (35 GB capacity).
BCV	Business Continuance Volume, a second mirror of the disk data that can be attached or split from the main data by S/W.
N+1 Server	A single server that provides resilience to a number(N) of servers.
ECC	Error Correction Code
UPS	Uninterrupted Power Supply.
Unison Maestro	An automated job scheduler.
Legato Networker	Data backup S/W from Legato.

Monterey
A Web Content Production System

John Edwards, Hugh Steele

ICL/BBC, London, UK

Abstract

Monterey is a Web based system for the rapid production and deployment of content to the Internet. It has been developed specifically for the BBC's commercial Internet service, *'beeb.com'*. This is a three-year partnership between ICL and BBC Worldwide and is focused on delivering a return on investment by harnessing the revenue opportunities from advertising, e-commerce and syndication built on key BBC brands, content and personalities.

This paper describes how the *Monterey* Content Production System (CPS) enables BBC editors, writers and designers to produce and publish stories, images and data on the WEB.

Introduction

This paper outlines:

- How the *beeb.com* project began
- How *beeb.com* has become one of the leading entertainment sites in the UK
- The challenges and the unique problems of the production environment
- The solutions and the innovations to overcome the challenges
- A detailed overview of the system and its architecture
- Future developments
- Summary and conclusions.

Section 2 provides background for those unfamiliar with the web and the *beeb.com* project.

Sections 3 to 9 are aimed at technical readers and describe the system and its architecture in detail.

Section 10 provides an overall summary and conclusions.

The *beeb.com* project

beeb.com is the result of a partnership (started in 1996) between ICL and BBC Worldwide. The service is built around favourite BBC programmes, magazines and personalities, delivering a wide range of content that focuses on comedy, motoring, sport, travel, TV & Film, and music. *beeb.com* users can also communicate with celebrities in on-line chat events, talk to each other in chat rooms, play games, participate in competitions and make on-line purchases at the BBC shop.

The BBC was quick to recognise the emergence of the Web as an additional New Media route to its viewing and listening audience. But to do this successfully, a technology partner was needed with an organisation to match the values of the world's greatest broadcasting brand and with a shared vision of the future of on-line media.

The BBC found this partner in ICL, which was equally keen to develop a new media culture, to learn a new way of "doing the Internet". The collaborative deal to create *beeb.com* was signed in August 1996. Since then, the pioneering relationship between "luvvies and techies" has proved successful and *beeb.com* has quickly become one of the UK's top entertainment web sites.

One of the powers behind *beeb's* creation was an ICL Director, John Davison, who stated, "Why is *beeb* so successful? Because the BBC is really good at creating content that grabs the audiences, engages them, invites emotional responses. The selection of sites on the Internet is infinite, and successful sites are the ones that provide people with compelling and exceptional on-line experiences. Success in the on-line world is not only about technology—it requires understanding of both technology and the creative process, in other

words, the coming together of arts and science, on a screen. And this is one of the most important lessons that ICL has learnt by working together with the BBC."

ICL's role as a technology partner in *beeb.com* covers three wide areas, ranging from setting the foundations, through developing the Content Production System, to revenue-related business systems.

At the foundation level, ICL built the network between all locations and the Internet, as well as the work environment at *beeb.com*. The business is located at the Television Centre in London, close to broadcasters and TV production teams. *beeb.com*'s continually changing content is hosted at ICL's Bracknell and Kidsgrove server farms, and back-up and recovery services come as part of the package.

The Content Production Centre at *beeb.com* is a good example of ICL's expertise and understanding of the media industry. In creating the system, ICL both integrated industry standard packages and developed bespoke applications when off-the shelf products were not available. The result is a unique Content Production system which links journalists, designers, editors and engineers in a dynamic network. This network enables each to use a range of web, audio and animation tools not only to create the content of the site but also to continually refresh the live web service.

Delivering great content in a fantastic-looking package is important—but clearly not enough to succeed in the long run. At *beeb.com*, these include on-line advertising management, automated measurement and reporting systems, communication tools and electronic protection of business assets.

In summary the objectives of this partnership are to:

- Develop new commercial, editorial, production and technical skills
- Create a base of excellence for each company
- Produce re-usable technology assets and production systems
- Earn revenue through a joint business.

beeb.com is an 'entertainment and leisure' service that combines news, features, interviews etc. with extensive information databases. It was targeted originally towards a young male audience at work in the UK, though home usage has grown considerably and new brands such as Gardeners World and Homes and Leisure are being added to support this growing audience.

beeb.com functions as an umbrella brand for the BBC's commercial on-line developments, and most importantly it earns revenue from advertising, subscriptions, transactions and content syndication.

Web development culture

The development and production culture at *beeb.com* has evolved somewhat organically over time. A traditional IT development methodology was applied initially, but it became clear that an approach requiring a complete specification to be in place before development began was not feasible. The starting point for many of the developments was a vision or creative concept that had to evolve through a process of prototyping and consolidation.

The development and production process has to support the following:

- A highly creative and ideas led environment
- Constantly changing and evolving requirements
- High content and design churn (some with hourly updates)
- Content that is focused on graphic design and visual impact
- A magazine model built on features, news, regular 'columns' and listings
- A need to maximize page impressions and views per user visit and ensure a 'paid for' advertisement on every page
- Fast and easy access to large databases that are updated frequently and are searchable by the consumer.

Production process

The production system is at the heart of *beeb.com* and must handle and process information and content from numerous sources. The production teams write editorials that must be integrated with internally produced graphics and imagery and all placed within a consistently styled template and frameset. Space for content is limited,

as *beeb.com* navigation, (up to six) advertisements, and *Webzine* navigation and links must also be contained in an 800 by 600 frame!

Third party data (in a wide variety of unstructured formats) and live feeds may also be integrated with internally produced content. This requires custom-built tools to allow data to be filtered and transformed before entry into the production system or relational databases.

All these content sources must be aggregated into HTML files (automatically in some cases) that are delivered to the consumer on the web. This complex process requires a number of disparate systems to inter-operate while presenting a consistent interface to the production system user.

beeb.com is a large operation, so a production pipeline with well-defined roles and responsibilities is vital. This allows resource pools to be set up which are efficient and highly skilled in their area.

These are:

- Graphic design and image creation
- JavaScript programming
- Editorial writing and research
- Navigation and user interface design
- Database and application programming
- Support and managed services—Webmasters.

Stakeholders

Dyer [Dyer, 1998] identified four main types of user within the *beeb.com* Production Centre, specifically:

- Web Developers—responsible for site design and integration along with HTML and Javascript coding
- Applications programmers—responsible for database design, object design and implementation in SQL and Java
- Editors and journalists—responsible for *Webzine* management and editorial writing.
- Multimedia researchers—responsible for daily creation and entry of content.

Monterey has been designed specifically to address the requirements of these four stakeholder groups, plus two additional parties:

- Webmasters—responsible for maintaining e' environment, ensuring 24x7 cover and overseeing deployment of new applications
- Technical Infrastructure—to define network and system architecture and ensure that the infrastructure has the necessary capacity and resilience.

Challenges

The existing 'manual' production process at *beeb.com* grew with the organisation and skills were gained and developed progressively. In isolation, this was a reasonably efficient process in that it capitalized on existing skills and freely available tools. However it meant that *beeb.com* was operating as essentially independent web production teams—one per *Webzine*. Each production team was fairly happy with this, but as the site's content base grew and production schedules tightened it became increasingly less efficient to have editorial staff and writers modifying HTML code and using FTP to deploy this directly to web servers.

From the business viewpoint this was producing no economies of scale and was rapidly becoming a concern for the centralized development, support and infrastructure functions. The need to re-purpose content for external partners and re-use content between sites was also a growing requirement.

In summary the challenges faced in developing a production system were to:

- Support, not change, the established working practices
- Underpin existing skills and tools—specifically HTML authoring and FTP utilities
- Significantly improve the productivity of the production teams and reduce dependency on technical (programming) resources
- Provide an easy to use (and install) interface
- Provide a central repository for content objects such as data, templates and applications
- Track and audit all content entry and deployment
- Ensure fast delivery of dynamic content from databases
- Deliver content in multiple formats—DHTML is becoming well established and WebTV is on the way.

The HTML paradox!

HTML has galvanised the IT industry by providing a non-proprietary standard for presenting text and images on any platform. With minimal effort, most people are able to grasp the basics of the language and produce their own content using simple tools.

However, it is the simplicity and monolithic nature of HTML that causes problems when the scale of its production and deployment process goes beyond individual authors.

A single monolithic HTML file used to define an increasingly complex page does not yield to economies of scale or a production pipeline involving multiple teams of specific skills. It also confounds re-use as the lack of modularity in HTML source requires code to be continually cut and pasted.

The many functions of HTML

The tags defined by the language perform a wide range of functions with only a small set specifically containing content. A page of HTML must not only hold the text and images, but also defines page layout, navigation, server directives, data input formats and even code (scripts). Examples of the types of tags are:

- Text: **<FONT>, <B>, <HR>**
- Layout: **<TABLE>, <FRAMESET>**
- Structure: **<HEAD>, <TITLE>, <BODY>**
- Data entry: **<FORM>**
- Navigation: **<A>**
- Code: **<APPLET>, <SCRIPT>**
- System directives and additional data: **<META>**.

This produces a daunting and often confusing task for those who are simply trying to edit a few lines or add a new picture. It has required production staff, whose main function is as writers or editors, to become proficient in HTML in order to get their output into the necessary web pages.

In the industry, this has created a new breed of 'all-rounders' who have come to the fore with the combined skills of graphic design, typesetting, programming and writing who can single-handedly create web pages. These individuals are rare, expensive and most suited to smaller web development teams.

With individual teams focused on specific aspects of the final HTML output, the emphasis shifts to the integration of the HTML components each is producing. This does not need to be a big bang and can be approached in an incremental way allowing the pages to build up in much the same way as an automotive production line. A standard 'chassis' for pages can be built that contains the common elements that the entire sub-group of pages requires: global navigation, advertisement serving code, overall page layout (frameset and background) on to which the page specific components are added. This chassis is known as a template and the process allows the components to be developed in parallel (and off site) if necessary. Most importantly, templates separate layout from content and protect the page layout when users enter text.

The *Monterey* Production Process

Figure 1 demonstrates how the initial site concept begins with a site map and navigation diagram, which is then populated with individual page designs mocked-up in Adobe PhotoShop. These mock-ups are used to visualise the graphic design concept, navigation paths and user interfaces to application functionality.

When reviewed and approved the designs are passed to HTML coders who will prototype the pages to produce a working framework of the site. These pages will be derived from base templates which include the standard HTML for global navigation, meta-tags and ad server calls. Any custom JavaScript functions will then be included along with overall page layout and background tags. Up to this point no page specific content will have been added, as what exists is an abstraction of the common elements of the site's 'look and feel'.

The next stage is to mark up the points within the HTML where content will be placed. This involves the addition of *Monterey* CPML (Content Production Mark-up Language) tags that instruct the *Monterey* Page Renderer on where to find the content specific to that page. This may come from two sources: components stored within the *Monterey* repository and from an ex-

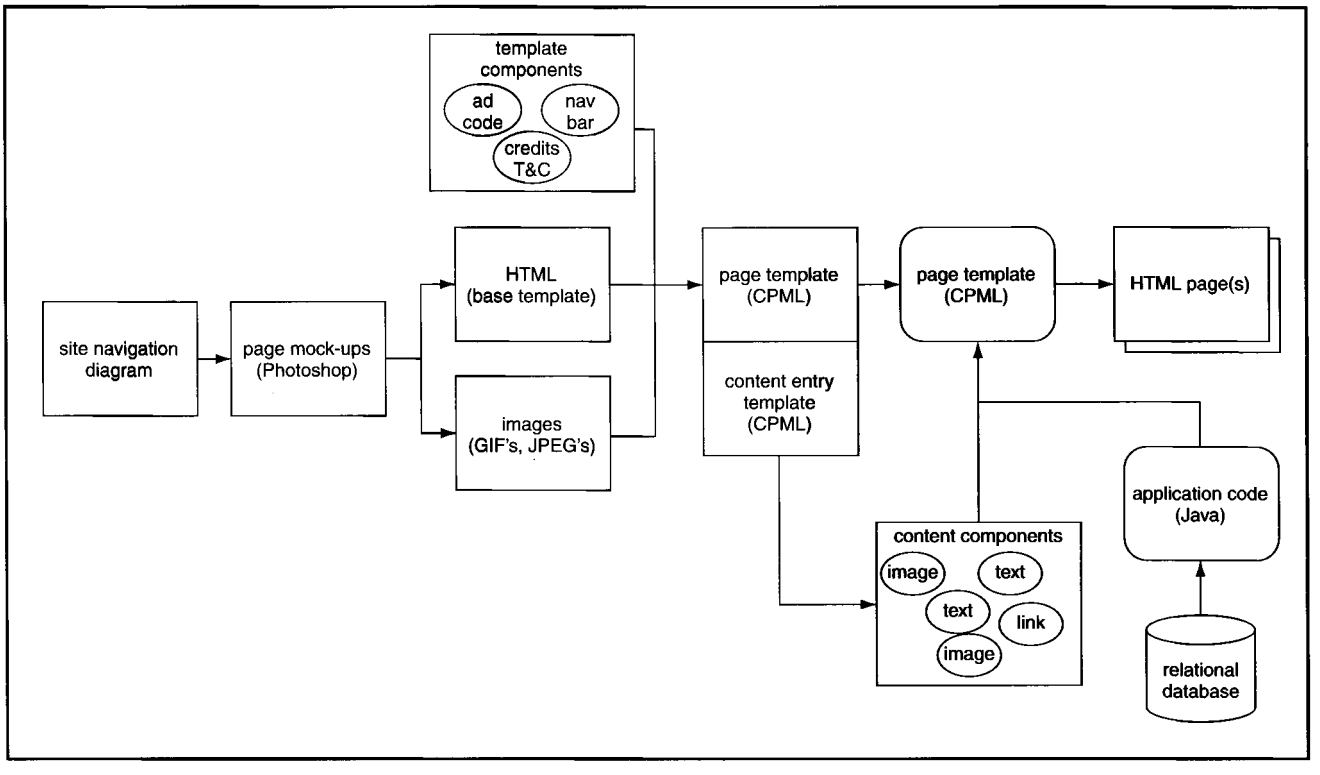


Figure 1: HTML Page creation process

ternally defined relational database schema. CPML tags and the Page Renderer are described in detail later.

Content components can be created directly in the *Monterey* repository using the page template in 'edit' mode or through a separate content entry template. The components can also be created externally using appropriate text and image tools and transferred into the repository using an FTP client. Components may be referenced by multiple templates, and in different websites, if they share the same repository!

Solutions and Innovations

As with any IT system, there were some significant pre-conceptions, user expectations to meet and resistance to overcome in successfully delivering a computer based production system that could satisfy all of the many 'stakeholders'.

The business required a system that would significantly improve the efficiency of the production processes and teams. It must allow them to focus purely on producing up-to-date re-usable content without the distractions of everything else that makes up a page. Advertising, navigation, deployment etc. should become controlled, centralized functions.

The main technical issues to be addressed are summarised below:

- Use a database to hold all content, code and data
- Make this database look like a web server file hierarchy to the user by creating a 'Virtual File System' (*VFS*)
- Use a web browser based interface to avoid constant switching between applications to preview work in progress
- Use templates to eliminate HTML code duplication and re-use UI designs
- Use the web-server cache to hold pages generated from a database and use these again for the next identical request
- Multi-publish to separate content from presentation which allows new formats to be simultaneously supported, such as DHTML and WebTV.

These issues and their implications are expanded and discussed in detail in the following sections.

Design overview

Re-use saves time and money and maximizes the revenue potential for a given expenditure of effort. It helps to ensure that tasks are performed less often and more efficiently. Re-use can be achieved across a number of "delivery vehicles",

including people, deliverables, tools and processes [Benner, 1997].

The *Monterey* technical design has endeavoured to apply the re-use principle in its broadest context, across all "vehicles" and stakeholder groups. The range covers tangible assets such as content, design, applications and infrastructure components, as well as the more abstract "people" benefits gained from having a consistent interface. The design encompasses existing assets, as well as providing a framework for the production of new material.

Skills (or "people") re-use has been achieved by considering each *beeb.com* stakeholder group individually and tailoring the system to make best use of their skills (see earlier section on Stakeholders).

Re-use of "deliverables" (e.g. work products) is generally achieved by packaging, or re-packaging, assets to produce re-usable components. Unsurprisingly, the decision to re-use involves a trade-off, as extra effort must be expended to make an asset re-usable. *Monterey's* approach to components is covered later in the section on Web Components.

Duplication of tools and processes has been largely avoided by designing a system that addresses the key requirements of the organisation and has potential for re-use externally.

Stakeholders

Naturally, the overriding goal has been to achieve maximum re-use with the minimum extra effort. However, experience at *beeb.com* has shown that the best way to achieve re-use is specific to each user area (stakeholder) of the organisation. The application of these ideas is therefore best understood from the point of view of each stakeholder.

Content producers

This group comprises editors, journalists and multimedia researchers, all of whom are responsible for creating publishable content. Re-use within this group is achieved principally by:

- Separating content components, from each other and from presentational elements
- Avoiding the need for editorial staff to learn new skills; i.e. HTML.
- Providing a consistent content production process allowing staff to move easily between different webzines and cross-pollinate skills across the organisation.

Web Developers

Web developers are responsible for implementing the design and high-level logic for the site. Conventional website production techniques confound re-use either by failing to support the separation and componentisation of design and content, or by requiring Web Developers to become programmers and database administrators.

- *Monterey* allows Web Developers to become more efficient by providing a framework for the re-use of template components.
- Web Developers only need to learn one language, CPML.
- CPML is designed to fit as closely as possible with first generation techniques, such as HTML, so Web Developers can re-use much of their existing expertise.
- Similarities with HTML also streamline the template creation process and aid re-use of existing Web-based assets.

Application Programmers

The *Monterey* systems architecture has been designed to avoid unnecessary duplication of applications programming effort.

- The system is developed around a Java component architecture. Well-designed components are inherently re-usable and the use of Java permits cross-platform re-use.
- The consistent applications architecture promotes skills re-use.
- The use of standard development practices extends the principle of knowledge re-use to potential recruits as well as current staff.
- Third party designs or implementations are used wherever applicable.

Technical Infrastructure and Support

beeb.com initially had to support a wide range of technology, from both the clients and the servers. Previously disparate areas of the system are now treated homogeneously, where possible (e.g. template and content components).

- The use of a browser-based interface allows infrastructure to be re-used for both live web

servers and content production systems (see later section on "First class interface").
- A three-tiered architecture maximizes the use of server architecture and reduces (thin) client support issues to a minimum.
- As the system supports the entire content production process, end-to-end, Webmasters only have to learn, deploy and support a single infrastructure.
- *Monterey* was designed specifically to re-use the existing server farm architecture.

Web Components

This section focuses on the re-usable deliverables created by the Web Development and editorial teams.

Applications components and Infrastructure are covered in the section on Technical Architecture.

Overview

Template and content components are different in terms of origin and use, but are intrinsically very similar.

Template components are produced and assembled by Web Developers according to original design requirements, whereas content components are created by editorial staff or delivered by automatic feeds. The two component types are then combined by the template engine to create publish-ready media; e.g., web pages.

From an operational perspective, these differences are marked, however in terms of system architecture both component types may be treated identically. This observation has meant that both component types can benefit from the same organisational and persistence mechanisms and a consistent user interface.

Template Components

Monterey promotes design re-use through template components and allows common presentational and logical features of the *beeb.com* site to be factored out as separate packages. This approach has the effect of "normalizing" the design elements of a site (*viz.* removing duplication) and ensures that the impact of a design change is minimized. Template components can then be combined to form new templates (Figure 2).

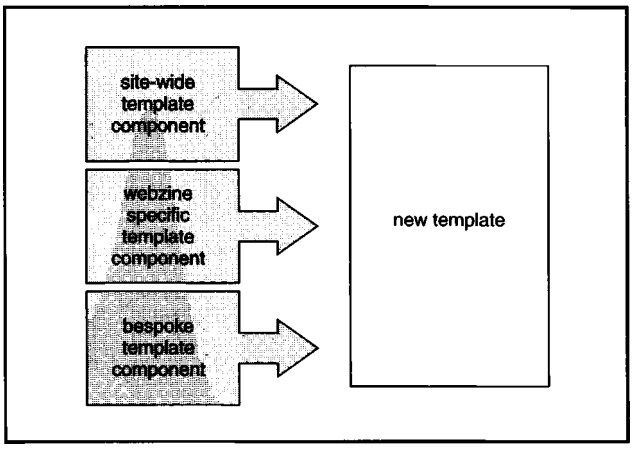


Figure 2: Combining template components to create a new template

One example of this technique is the "Terms and Conditions" (*T&C*) notice found at the bottom of every page on the *beeb.com* website. With first-generation web site production techniques (i.e. flat pages), a copy of the *T&C* HTML code would need to be present on every page. If the design of the *T&C* notice were to change, then every single page on the *beeb.com* site would need to be updated to reflect this modification. With a site comprising tens of thousands of distinct pages, this is not a trivial operation.

beeb.com templates within *Monterey*, however, simply contain a reference to the *T&C* template fragment. Only this fragment needs to be changed in order to upgrade the entire web site.

Content Components

Content is expensive and time-consuming to produce:

- In-house content must be researched, written and typed into the system
- Feeds-based content must be purchased, pre-processed and loaded.

Either way, it makes sense to extract maximum value from content via re-use. The potential for re-use of a given set of content can be greatly enhanced by re-presenting it in a variety of design contexts (re-decoration), or by re-combining subsets of content to produce useful new groupings (re-aggregation).

Monterey enables content re-use by separating data from presentation and supports content re-use through various aspects of CPML. It also provides the facility to repackage content auto-

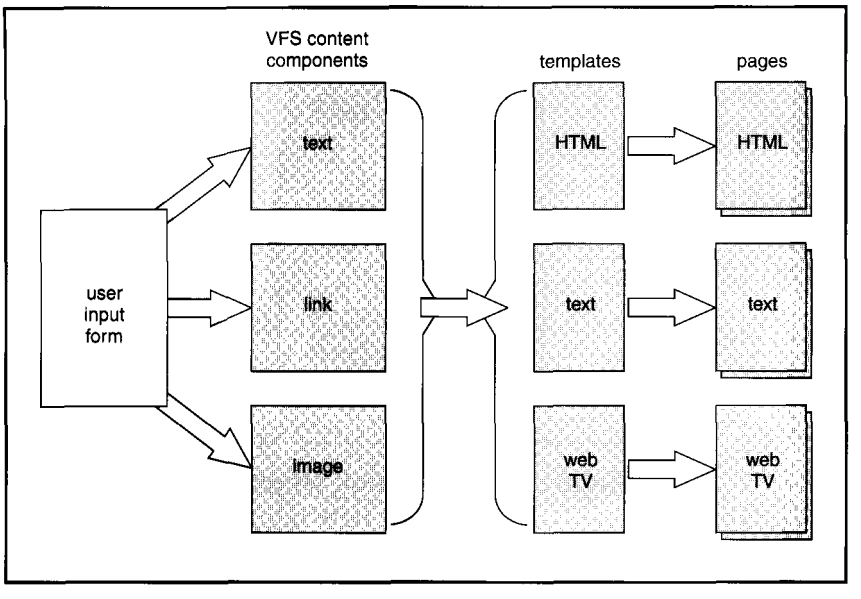


Figure 3: Multipublish

matically, without the need for duplication. In addition to supporting design normalization through template components, *Monterey* also promotes content re-use via Multipublish.

The HTML domain alone provides a wide range of possibilities for repackaging content. For example, the production team may want to provide browser-specific versions of a page, which can take advantage of proprietary tags and features. Alternatively, they may want to produce a frameless version of a page, or maybe a "printer-friendly" alternative.

Multipublish functionality is not restricted solely to HTML. By separating data from presentation, it is possible to re-use content in a wide range of formats, for example textual email, or Web TV (see Figure 3).

In addition to basic content/presentation separation, CPML also provides explicit support for multipublish allowing template designers to indicate that whenever a given template is used to produce a page, then further pages (based on other templates) should be generated automatically. For example, whenever a standard page is created, it should also create (say) an Internet Explorer specific version.

In its simplest form, Multipublish simply redecorates a given set of content with a different template. Multipublish is also a useful way to re-aggregate data. For example, a template may collect and display a list of "Title" and "Abstract" components from recent news stories.

Web Production Techniques

A key aim of the *Monterey* system is to streamline the production and maintenance of a web site, a goal it shares with other template-based content production systems. This section briefly examines the *Monterey* system in the context of alternative web production techniques, although a detailed appraisal of alternative approaches is beyond the scope of this paper.

First-generation web production techniques based on static HTML pages allow the initial page of content to be created very quickly. However, subsequent repeats of a design with alternative content, or vice versa, take almost as long as the first, as each page must be created in its entirety.

Template-based systems separate content from presentation, which promotes re-use and accelerates the creation of subsequent pages by allowing multiple sets of content to use the same template. However, a significant amount of effort may be required before this stage is reached, including:

- Converting the initial page into a template
- Allocating content storage
- Producing an interface for creating and updating content
- Specifying rules for content retrieval.

These up-front costs, often requiring the intervention of skilled IT staff, can make templates an unattractive option, especially for "small-scale" re-use, where the template is only used to create a few pages.

Scripts

Script-based approaches to template production combine program code (e.g. VBScript) with presentation (e.g. HTML) to generate each page; the scripting language being responsible for retrieving, processing and displaying individual content components, often from a relational database.

In some cases (e.g. Microsoft ASP), templates may be manually created from HTML by identifying and removing the content areas of the page, then replacing them with a scripting language to retrieve and display the content. Alternatively, fragments of HTML may be embedded into a program as "print-statements", a strategy commonly adopted by early CGI developers. In either case, these are skilled tasks requiring the involvement of a programmer and possibly a database administrator. There is also the disadvantage that, although content has been separated, business logic remains bound up with presentation. Additionally, a system for creating, storing and updating the required content elements must already be in place.

Nevertheless, script-based approaches can be extremely powerful, especially in situations where the presentation and data schemas are largely fixed.

Stylesheets

Stylesheet approaches to templating, such as CSS (Cascading Stylesheets) and XSL (Extensible Stylesheet Language), take a document containing content (XML/HTML and XML respectively) and add presentational elements according to a set of rules.

The source document may be read from a static file, or generated dynamically by a separate script. Unlike a simple script-based approach, this has the advantage of keeping the business logic involved in content aggregation separate from the presentation.

A further advantage of stylesheets is that modern web browsers have stylesheet functionality built-in, allowing the rendering process to take place on the client, thus reducing the load on the server. However, browser support for stylesheets is still patchy (especially for XSL), so rendering must currently occur on the server if correct output is to be guaranteed.

Both XSL and CSS can be used to style XML documents (i.e. the content), but only XSL has the ability to transform the XML document as well, providing a far richer set of rendering options. However, while CSS is relatively easy to create, XSL templates can rapidly approach the complexity of a scripting language, which probably places them out of reach of most Web Developers, at least until effective WYSIWYG development tools begin to appear.

Stylesheets have the potential to become the primary content delivery method in future, but, for the time being, the lack of consistent browser support and graphical development tools is slowing their adoption. The development team is currently exploring the use of CPML as an XML-generator for XSL stylesheets in anticipation of suitable tools becoming available.

As with scripts, however, stylesheets fail to fully address the content production problem, largely due to the effort required in creating the initial templates and the need for an additional content maintenance infrastructure.

CPML advantages

By minimizing the *total* cost of producing templates, CPML can prove effective even for small template:page ratios, while retaining the economies-of-scale enjoyed by standard template-based approaches.

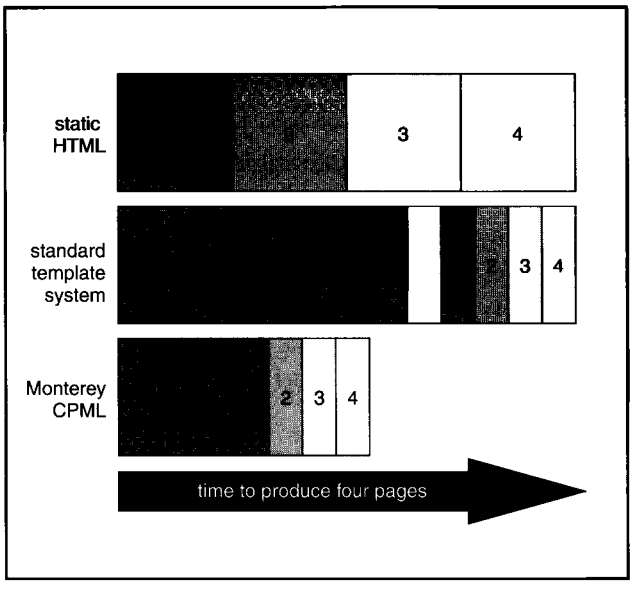


Figure 4: Fast initial content creation

CPML significantly reduces the effort required to produce the initial template (see Figure 4) by using an HTML-like approach to template directives and relying on an integrated datastore (see later section on Virtual File System) for most content store functionality.

When a site has been created as static HTML, a design change requires that all the pages based on the old design are changed to reflect the new

design. As content and design are mixed this is difficult, if not impossible, so a significant content investment may be wasted.

Template-based systems isolate content from design, which makes presentational changes much simpler. However, if scripting code additions were used to create the template, then updates can only be effected by skilled applications staff. Stylesheets can allow design updates to be made by Web Developers, but in either case changes or additions to content elements can often require skilled modifications to the core database schemas.

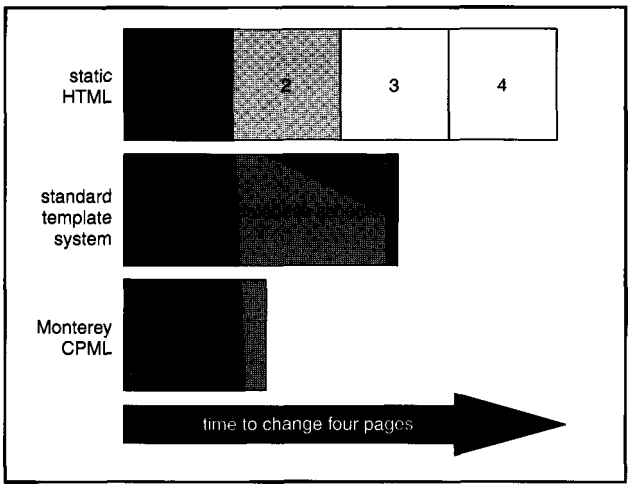


Figure 5: Fast design change

Monterey separates content and design, but also allows Web Developers to change content profiles simply by modifying a template. No expensive schema modifications are required, which can help to reduce the effort required to change a design (see Figure 5).

In most conventional template-based systems, converting an HTML design into a template requires certain key page components—those that will incorporate external data—to be removed and replaced with procedural script code.

Although HTML is fairly robust when rendering within different browser implementations, it is quite fragile when large sections are removed and replaced with non-native code. The absence of key page components makes it impossible to get a true impression of the page layout and the programmatic additions can render the page unreadable in a browser. Therefore, the conversion process is time consuming and awkward and the resulting web templates cannot be usefully viewed in a conventional browser.

The problem becomes even more significant when the templates need to be changed, as standard HTML tools can no longer easily edit them. Changes must be made "blind", and can only be checked by passing them through the CPML engine where they are combined with data.

CPML is designed specifically to ensure that templates continue to exist as viewable HTML documents. Rather than removing key page components to create a template of "holes", CPML allows dynamic elements to be marked in place. No content needs to be removed, so CPML templates can be viewed and edited using conventional HTML tools and published as finished pages. Numerous pages may be instantiated, which follow the design of the original template page, but allow alternative content. The template rendering code ensures that default text is replaced by external data where appropriate.

Technical architecture

Monterey uses a three-tier approach comprising a Data Server, Application Servers and two types of thin client, namely Web Browser and FTP (see Figure 6).

As a three-tier architecture, *Monterey* centralizes business logic at the Application Server layer. This approach has a number of advantages:

- Simplified system administration
- Support for heterogeneous thin clients
- Improved performance (fewer network calls per operation; core logic executes on high performance server)
- Better encapsulation of data
- Better security (method-level as opposed to data-level)
- Enhanced potential for server component reuse
- Potential for server-to-server communication
- Improved scalability and resilience using multiple servers.

Data Server

The Data Server provides a platform for all *Monterey* data. It supports the Virtual Filesystem

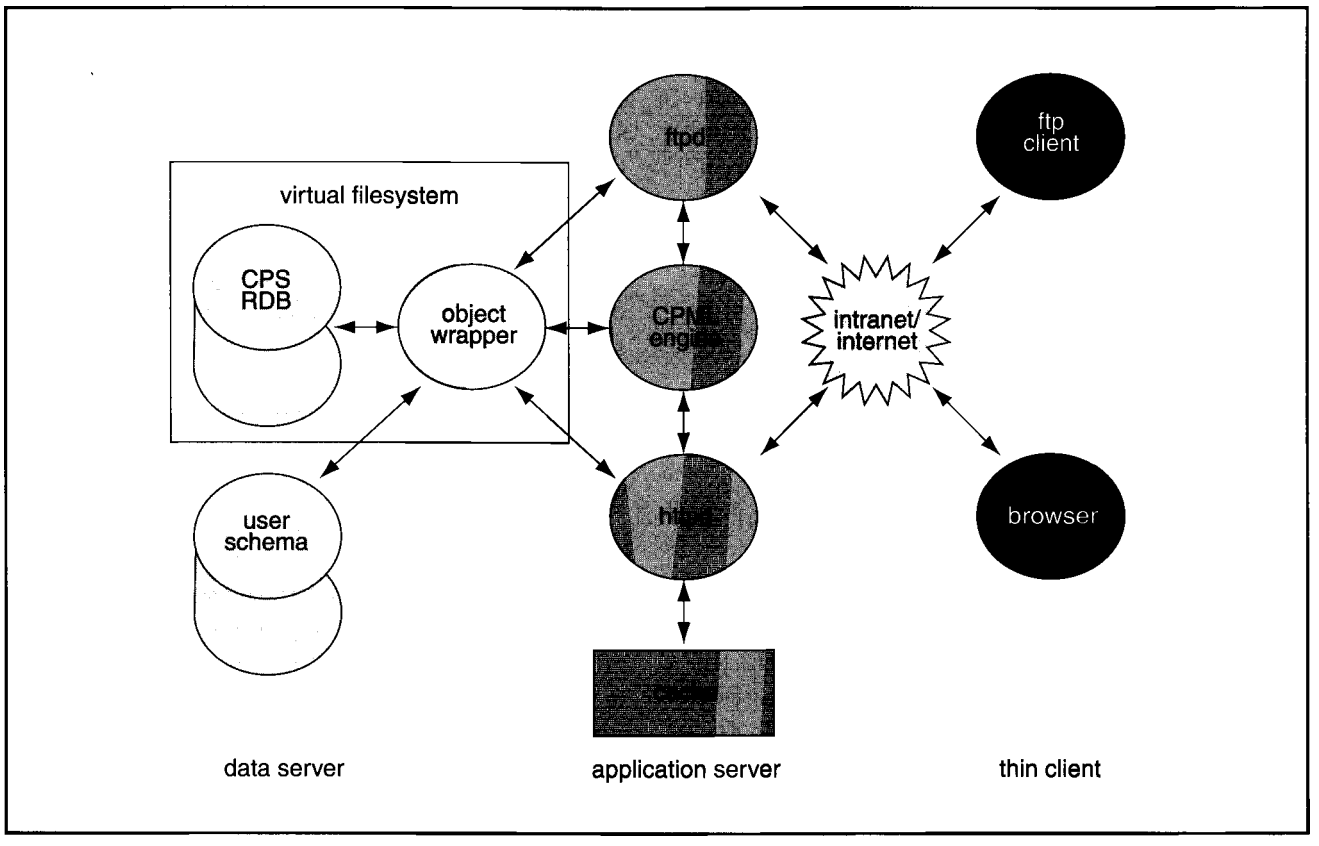


Figure 6: Technical Architecture

(VFS) as well as any user defined relational schemas.

The Data Server is actually two physical servers mirrored to provide resilience. Further details of *Monterey*'s high availability architecture are given in a later section.

Virtual Filesystem

The Virtual Filesystem (*VFS*) is a key component of *Monterey* and was implemented early on in the lifecycle of the system. It combines the best attributes of an industrial strength relational database (the system currently uses Oracle) with an organisational metaphor that is appropriate and familiar for web site design.

The *VFS* is implemented using a custom relational database schema. A Java object-relational wrapper encapsulates access to the *VFS*, plus any user-defined schemas.

The *VFS* emulates the hierarchical arrangement of a normal filesystem (i.e. directories and files), and provides the same basic facilities, for example:

- Create a file
- Create a directory
- Change a directory
- Read file contents
- Delete a file.

However, the *VFS* offers significant additional benefits over a traditional filesystem in terms of structure, functionality and performance.

Extending the hierarchy

An important structural benefit of the *VFS* is that it seamlessly extends the hierarchy both above and below the normal filesystem scope.

Table 1 shows the various layers managed by the *VFS*; those available in a normal filesystem are emboldened. The right-hand column shows how each level corresponds to elements of a standard URL.

Level of hierarchy	URL example
Protocol	http:
Subscriber	//www.beeb.com
Directories	/html/
Document	TodaysPage.html
Text Component	#FirstCaption

Table 1: Hierarchical components of the *VFS*

At the very top of the hierarchy, *Monterey* currently supports two thin client protocols; HTTP and FTP. Support for other protocols, for example SMB (SAMBA) may be added in the future. One level down, the *VFS* allows for multiple content subscriber groups. In the simplest case, each subscriber group may be an alternative website; other potential subscriber groups include ftp syndication or listserv groups. Managing all the subscribers within a single content repository promotes the sharing and re-use of content/design components.

Next come directories and documents, which are handled in exactly the same way as a normal file system.

Finally, at the other end of the scale, the *VFS* treats components within a document in exactly the same way as documents within a directory. Whereas the leaf node of a conventional file-system is the file, the *VFS* imposes no limits on granularity. It is therefore just as simple to refer to a specific text fragment within a page, as it is to refer to a page within a directory.

The homogenous treatment of the complete web hierarchy is apparent at the user interface too; just as a user may browse though directories and see pages, they can also navigate between different subscribers, or browse within pages and examine the composite text fragments.

Audit trail and point-in-time recovery

A key functional benefit of the *VFS* over a traditional file-system is the provision of full audit trail and point-in-time recovery facilities. Every update to an object is stored as a distinct new version and includes full details of who made the change and when.

This mechanism operates transparently and allows for all or part of the *VFS* to be rolled back to a specific point in time. Alternatively, content can be viewed as it was at a specific time, which fulfils a significant legal requirement of the BBC contract.

Explicit support for the time dimension of every component also allows for a very flexible publish-subscribe model, where different subscribers can view content at different stages in its lifecycle. For example, editorial staff producing content would expect to see documents as their most recent version, while the outside world sees a frozen snapshot of earlier versions. This model may be extended to allow, for example, premium subscription users to see more up-to-date content than the general Internet user.

It is also possible to generate reports showing the detailed history of an object or group of objects, or to enumerate all content changes made by a specific user.

The database schema is optimized to support the transparent audit trail with no discernible effect on performance.

Asset Management

A large content and design asset quickly becomes unmanageable if there is no easy way to locate and extract the components required for a given task. Component re-use is impossible if the components cannot be located and extracted.

This is especially significant for *Monterey*, where content and presentation have been separated and componentised; therefore an efficient mechanism is required to select and recombine the relevant components.

Traditional file-systems provide very basic querying mechanisms, which are not suitable for searching through high volumes of data, nor for complex restrictions such as joins. Examples of traditional "query" mechanisms include directory listings (e.g. "dir" and "ls") and simple hierarchical commands (e.g. "find"). These facilities are useful for occasional searches or small data volumes, but do not scale.

Relational databases, on the other hand, are specifically designed to handle complex queries quickly and efficiently. The internal *Monterey VFS* schema has been constructed to take advantage of this innate capability. However, the raw searching capabilities of the relational database are not exposed directly to users or Web Developers; instead they are encapsulated by the object relational wrapper and made available through the CPML template language and at the user interface. This approach hides the underlying VFS implementation enabling future upgrade.

The *VFS* also provides for multiple categorizations via a "shortcut" metaphor, which maxi-

mizes the potential for asset classification and tracking.

Transactions

Transactional behaviour is an essential aspect of most computing systems and Internet content production is no exception. For example, the mesh of hypertext links which binds a website together requires that groups of related pages are either all published or not published at all. If the process were allowed to fail half-way through, the web site would probably be left with many broken links. A transactional publishing process ensures that even when something does go wrong, the site is left in a consistent state. The atomic nature of a transaction also make it impossible for sections of the content to be changed while the publishing process is underway.

Conventional file-systems, the traditional web site substrate, are not naturally transactional in nature (although basic atomic transactions can be elicited with some programming effort).

Through its use of relational database technology, *Monterey* is inherently transactional for all operations, including publishing, copying, and deletion.

Other RDBMS features

The task of implementing audit trail, asset management and transaction functionality within the *VFS* was greatly simplified through the re-use of the Oracle RDBMS feature set.

The decision to use Relational Database technology has also provided a wealth of additional benefits which became available without additional programming effort, for example: Mirroring, Concurrency, Archiving, Backup and recovery.

Middleware components

Monterey middleware components currently include:

- HTTP Service
- FTP Service
- CPML Parsing Engine.

HTTP Service

The HTTP service provides a browser-based interface to *Monterey*. It accepts and processes browser-originated requests and generates HTML pages dynamically via the CPML Engine. The service is implemented as a Java servlet, which currently runs under JRun using Netscape Enterprise Server 3 as a "listener". The system will run without modification on any Web Server with servlet running capabilities, and has been tested using Microsoft Personal Web Server and Jrun on NT.

The HTTP Service (and hence the web browser interface) is the main access point for editorial staff to create and organise content. It is also used by Web Developers, to a lesser extent, for creating and organising template components.

The decision to use a browser interface has provided a number of advantages; some are the generic benefits of a three-tier architecture; some are specific to the context of web content production.

Multi-platform support

beeb.com is a multi-platform site. Designers generally use Macintosh computers, whereas production staff use a mixture of Mac and PC systems. Webmasters and Applications Developers also make heavy use of PC technology.

It was therefore imperative to choose an architecture which could make use of a cross-platform interface.

A browser-based interface was the obvious way to maximize the compatibility of the system across multiple operating systems and specifications.

Remote access and teleworking

Many of the staff at *beeb.com* require remote access to the system from time to time, be they support staff working out-of-hours or editorial staff on an "outside broadcast". A browser-based approach is naturally suited to remote operation and *Monterey* has been used successfully from overseas on a number of occasions.

Minimise rollout and maintenance costs

A fat client approach (for example, a program written in Visual Basic or Delphi) would require that every user have the program installed on his/her PC before they could start using it. This means that support staff have to be involved with the initial installation, plus any compatibility and upgrade issues.

Web Browsers, on the other hand, are ubiquitous, so any new user simply needs to be allocated a username and password and informed of the URL.

Minimize support costs

When users experience problems with a two-tier application, the difficulty is often due to the client configuration.

Major variations in client technology can make it difficult for support staff to diagnose and resolve a problem, especially from a remote location.

Monterey confines all critical processing to the server side. Therefore, any problems can be quickly isolated and rectified by support staff. The browser interface is largely invulnerable to client configuration problems.

"First class" interface

A web browser based interface is particularly significant given the nature of the content being produced. Specifically, the interface used to produce the content exists in the same medium as the content itself—it is a "first class" interface.

This has two main benefits. Firstly, it allows content to be previewed accurately within the interface. Secondly, it encourages a symbiotic relationship between the content production environment and the delivery environment, such that any functional enhancements to the content production system are of direct benefit to the delivery architecture, and vice versa.

This is of particular importance with regard to the advanced resilience and high availability of the production delivery service. Sharing a common infrastructure means that any performance and resilience enhancements made to the live service (e.g. load balancing, fault tolerance) will be of direct benefit to the content production system. If differing technologies had been used, then up to twice the effort would be required to guarantee high availability on both systems.

Similarly, if Applications Developers improve the content production interface via CPML enhancements, then these improvements are instantly available to live templates.

A "first class" interface confers benefits to the Web Developers as well, since their CPML expertise allows them to tailor the interface to their (or their users') preferences and to create bespoke interfaces for specific content areas. Figure 7 (next page) is an example screenshot from *Monterey* showing a user input form (left pane) adjacent to the finished presentation—both templates were easily designed and created by Web Developers.

In the longer term, it also provides the technical foundations for an architecture where the current site consumers (the Internet public) can become regulated content providers.

FTP Service

The FTP Service provides an alternative thin-client access route to the *Monterey VFS*. It is implemented as a Java socket server, which exposes the *VFS* to the outside world via the standard FTP protocol. This service is particularly suitable for Web Developers, who generally work with large volumes of template components and multimedia.

FTP clients are freely available for most (if not all) platforms and most Web Developers will already have one or more installed on their systems, thus minimizing deployment costs. The ability to use platform-specific FTP clients has also helped by conferring native client performance and functionality to the *Monterey VFS*, for example "drag-and-drop". Moreover, FTP client functionality is often built into third party web development tools, a fact that has enabled *Monterey* to become quickly an integral part of the Web Development process.

The FTP Server was relatively straightforward to specify and implement, as the FTP API is stable and well-defined and a range of Java FTP server implementations were available in the public domain.

The FTP Service does not replace the Web Browser interface, but is a useful alternative for certain Web Development tasks.

CPML Parsing Engine

Both the FTP Service and the HTTP service are able to access directly individual components within the *VFS*.

However, these services also access the *VFS* via templates, which allow multiple content and design components to be collected and assembled to produce finished documents.

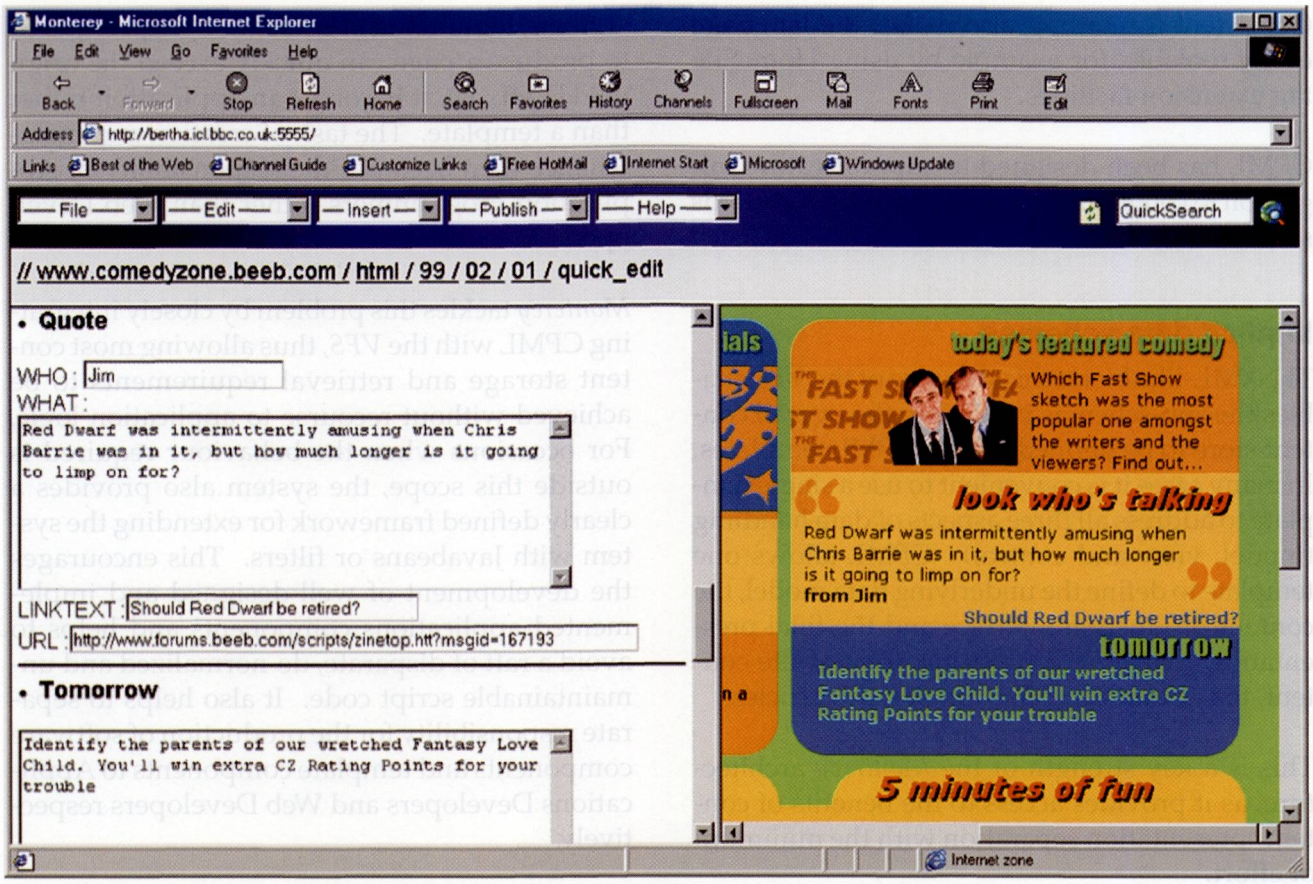


Figure 7: ***Monterey*** **screenshot demonstrating a "first class" interface**

Support for templates in *Monterey* is provided though a specially developed XML-based language known as CPML (Content Production Mark-up Language) which is processed by the CPML parsing engine.

The following section discusses the CPML template language and processing engine in more detail.

CPML Template Processing

In accordance with *Monterey*'s high-level design goals, CPML aims to:

- Support re-use of work products through the separation of design, content and logic components
- Support re-use of infrastructure and skills by defining the interface as well as the publishable product (see section on the "First Class" interface)
- Retain the simplicity of HTML.

The last point is probably the most important of all, as many template-based approaches fail because template creation and maintenance is difficult and time consuming.

CPML provides a standard set of elements, which permit Web Developers to carry out most templating tasks quickly and easily. The language is easily extensible using well-known Java idioms.

CPML is designed to offer significant benefits over first-generation template systems for most content-design ratios. The remainder of this section gives an overview of the key aspects of CPML.

Standards based

CPML uses the W3C XML standard syntax, an approach that has provided easy access to a wide range of third party specifications, designs and implementations. For example, it is possible to parse a CPML template using any XML compliant parser[1].

[1]Non XML compliant HTML code is fixed automatically before being presented to the parser.

Consistent formatting also makes the language easily toolable, for example by using HomeSite tag extension facilities.

CPML has been designed to support future migration to contemporary and future web standards such as CSS, XSL, XLINK, and XPOINTER.

Implicit data schemas

The XML-like hierarchical nature of the *VFS* enables the sub-schemas implicit in the overall content store to be defined within CPML templates. In many cases it is convenient to use a single template to address all three aspects of data handling (model, view and control). CPML allows one template to define the underlying data model, the content production interface and the final presentation format for a particular area of site content, for example a collection of news articles.

This is a key strength of the *Monterey* architecture, as it provides access to the benefits of content/presentation separation with the minimum of effort.

However, just because it is possible to define all three aspects with a single CPML template, does not mean that this is always the best approach. For example, one or more templates can be used for presentation, while other templates are designed specifically for creating and updating content. The data model itself is implicit in the union of these related templates, and could be defined or refined by an external feed if required. *Monterey* provides (via CPML) support for either approach, depending on the specific application.

Declarative, not programmatic

With its roots in XML, CPML enables Web Developers to specify their content and presentation needs declaratively.

In script-based template systems, such as Microsoft's ASP or Vignette StoryServer, templates are generally created by embedding a fully functional scripting language (e.g. VBScript, Perl, TCL) into HTML. While access to a full scripting language is compelling in the short term, it can quickly result in complex templates, many of which contain duplicate functionality. In such systems, the amount of embedded script required to produce a page can quickly exceed the original HTML; i.e. it becomes an application rather than a template. The task of creation and maintenance then becomes the responsibility of Application Programmers rather than Web Developers.

Monterey tackles this problem by closely integrating CPML with the *VFS*, thus allowing most content storage and retrieval requirements to be achieved without recourse to application logic. For occasions when the behaviour required is outside this scope, the system also provides a clearly defined framework for extending the system with Javabeans or filters. This encourages the development of well-designed and implemented applications components and helps to avoid a raft of disparate, de-normalized and unmaintainable script code. It also helps to separate responsibility for the production of software components and template components to Applications Developers and Web Developers respectively.

The bean approach is similar to the use of ActiveX components in ASPs, but has the additional advantage of cross-platform portability.

Quick to produce and maintain; compatible with existing resources

The overriding design goal of the CPML language has been to choose the option which best supports fast template creation and maintenance. The close proximity of the *Monterey* development group to the *beeb.com* production environment has allowed these design decisions to be based on real domain knowledge.

CPML is designed as a natural extension to HTML and as such it is easily understood and used by Web Developers. Wherever applicable, a similar syntax and semantic has been adopted. In some cases, standard HTML elements (e.g. `<IMG>`) gain additional functionality and *VFS* integration without intervention. This enables Web Development resources, both internal and external, to be re-deployed quickly as CPML template creators. CPML also allows *beeb.com*'s significant static HTML legacy to be re-purposed easily as CPML templates.

CPML element handler

The CPML element handler is based on the publicly available SAXON Java class library [Kay, 1998]. SAXON, in turn, is built on top of the standard SAX interface, which provides an event-driven API to XML documents.

SAXON provides many useful extensions to basic SAX functionality, including an element-type based event distributor. Through SAXON, the *Monterey* infrastructure is able to provide a clear framework for extension of the CPML language on a class-per-element basis. A new CPML element is simply a Java class, which specializes the CpsHandler class.

The following sections describe the main aspects of the CPML language, but are not intended as an exhaustive reference; for that readers should refer to the *Monterey CPML User Guide* [Wells, 1999].

Core elements

The **<CPSTEXT>** tag is at the heart of CPML and is responsible for binding a template component to other components in the underlying *VFS*. The idea behind **<CPSTEXT>** is simple; it is essentially the textual equivalent of the HTML **<IMG>** tag.

Depending on the context, a **<CPSTEXT>** element will be displayed as read-only (view mode) or editable (edit mode). If edit mode is enabled, then the contents of **<CPSTEXT>** elements can be changed from directly within the browser interface with updates being reflected in the *VFS* automatically. The context varies depending on how the element is being accessed (e.g. edit mode when changing a page from within the production interface, view mode when published) and may be changed dynamically using the **<CPSSHOW>** tag.

The contents of **<CPSTEXT>** elements are recursively parsed, allowing template components to be nested. This also permits editorial staff to create further **<CPSTEXT>** references within content on an *ad hoc* basis, although this is the exception rather than the rule. If further parsing is not required, then the **<CPSINC>** element can be used to force an unparsed verbatim inclusion.

```
<HTML>
<BODY>
<H1>Article Title</H1>
<HR>
The abstract of the article goes here.
</BODY>
</HTML>
```

Example 1: HTML before conversion

The above example (1) shows a simple HTML page that displays a title and an abstract separated by a horizontal line:

In the next example (2), the requirement is to allow editorial staff to edit the title and text of an article without having to edit the HTML file directly. The relevant sections of the HTML page (shown in italics) are marked-up using **<CPSTEXT>** elements to separate the variable content from static HTML presentation. In addition, the design of the plain HTML page has been augmented with standard template components that can be assumed to exist in the system already.

```
<HTML>
<CPSTEXT SRC="./header"/>
<BODY>
<H1><CPSTEXT SRC="*/title">Article Title</CPSTEXT></H1>
<HR>
<CPSTEXT SRC="*/abstract">
The abstract of the article goes here.
</CPSTEXT>
</HR>
<CPSTEXT SRC="/standard-components/footer"/>
</BODY>
</HTML>
```

Example 2: HTML with CPML extensions

For this very simple page, the ratio between HTML and "content components" is approximately equal. In a typical *beeb.com* HTML page, the ratio would probably be closer to 10:1, while the amount of CPML required to produce the template would remain the same.

By default, the **SRC** attribute operates relative to the template rather than the page. This design decision rests on the observation that the major-

ity of relative references in a typical HTML page (i.e. **IMG** elements) are concerned with common presentational components rather than instance specific content. Therefore, the Web Developer has only to change the content-specific elements to transform an HTML page into a CPML template. The first **<CPSTEXT>** element in Example 2 uses a template-relative path to include a standard header component from the same directory as the template.

The article title and abstract in this example, however, are content elements, specific to each published page. In these cases, the * symbol is used within the **SRC** attribute to indicate that the content path should be generated relative to the page rather than the template. For example, if a page based on the template is stored at location "`/articles/article1`" in the *VFS*, then the article title will be retrieved from location "`/articles/article1/title`".

The second example also illustrates how content elements can be defined non-destructively; the original title and abstract text remain in the template. This approach allows the template to continue to render correctly within standard web browsers and also provides default text *in situ*, which is displayed if the referenced item is not present in the *VFS*.

The **<IMG>** tag is part of the HTML specification and is not unique to CPML. However, within *Monterey* the **<IMG>** tag can be used automatically to create and edit multimedia data from within the browser interface, as opposed to just viewing it. The **<IMG>** tag follows the same rules for determining relative paths and view/edit contexts as the **<CPSTEXT>** tag. By parsing and processing the native HTML **<IMG>** element, the *Monterey* templating system further reduces the effort required to produce templates.

External class support

The **<CPSBEAN>** tag provides a framework for incorporating Java classes that follow the JavaBean design pattern for scalar mutators.

Bean fields are set (via introspection) according to **<CPSPARAM>** elements appearing within the bean tag. When the **<CPSBEAN>** tag is closed, the *doProcessing()* method is executed. The **DST** attribute allows the Web Developer to direct the results to an area of the *VFS*, where it is then available for the duration of the request. If the **DST** attribute is not present, then the output is inserted directly into the page.

The next example (3) demonstrates the use of **<CPSBEAN>** in both contexts.

```
<CPSBEAN NAME="com.beeb.beans.
  NewsFinderBean" DST="/todays_news">
 <CPSPARAM NAME="SearchBy">Date</
                                CPSPARAM>
 <CPSPARAM NAME="SearchValue">
  <CPSBEAN NAME="com.beeb.beans.
                              DateBean"/>
 </CPSPARAM>
</CPSBEAN>
```

Example 3 : CPSBEAN and CPSPARAM

The outer *NewsFinderBean* generates a list of news articles based on supplied parameters and places the result at location "`/todays_news`". Note that this dynamic dataset resides in memory, it is not physically stored in the database. Moreover, the dynamic data is only available in the context of the current request. The standard path notation allows request-level dynamic data to be accessed in exactly the same way as persistent data stored in the *VFS*, for example using **<CPSTEXT>** or **<CPSLOOP>**.

The inner *DateBean* is used in-line (i.e. without a **DST** attribute) to provide a parameter for the article search.

The **<CPSQUERY>** tag is a specialization of **<CPSBEAN>** and provides a user-defined mapping between an external relational database schema and the request-level *VFS*.

The **<CPSFILTER>** tag allows for the introduction of bespoke stream-based processing into the CPML language. New filters are created by specialising the standard java.io.FilterWriter class. The next example (4) shows a **<CPSFILTER>** tag being used to transform its contents to uppercase. **<CPSFILTER>** elements may be nested as required.

Iteration and conditional logic

<CPSLOOP> is used to iterate through objects at a specified node of the *VFS*. Some nodes of the *VFS* contain persistent objects, whereas others contain request level data, such as that created

by <CPSBEAN> or <CPSQUERY> elements. Like other CPML elements, <CPSLOOP> is unaffected by this distinction and treats entity and request level data identically.

```
<CPSFILTER NAME="com.beeb.filters.
                              Uppercase">
   Everything will be displayed in
   uppercase when rendered including
   the data from this CPSTEXT element:
   <CPSTEXT SRC="*/example"/>
</CPSFILTER>
```

Example 4: CPSFILTER

<CPSBREAK> provides a mechanism for applying simple pattern-action rules to each <CPSLOOP> iteration. If the pattern defined in the <CPSBREAK> element is matched, then the contents of the <CPSBREAK> element are executed; otherwise, they are ignored. Patterns include the iteration index (e.g. first row, an even row), element equality or inequality, or changes in specific values from one iteration to the next (similar to "breaks" in traditional database reporting languages).

In the next example (5), it is shown how <CPSLOOP> and <CPSBREAK> can be used to iterate through a node in the *VFS*, in this case a request level node generated by the <CPSBEAN> shown in Example 2.

```
<CPSLOOP SRC="/todays_news" DST="/x"
                    FROM="1" To="10">
 <CPSBREAK INDEX="1">
  <CPSTEXT SRC="/x/headline">headline
                              </CPSTEXT>
  <P>
   <B>
    <CPSTEXT SRC="/x/body">first
     article is shown in full</CPSTEXT>
   </B>
  </P>
 </CPSBREAK>

 <CPSBREAK INDEX="2+">
  <CPSTEXT SRC="/x/headline">headline
                              </CPSTEXT>
  <P>
   <CPSTEXT SRC="/x/abstract"/>latter
    articles show abstract</CPSTEXT>
  </P>
 </CPSBREAK>
</CPSLOOP>
```

Example 5: CPSLOOP and CPSBREAK

Basic conditional logic is supplied by <CPSIF>, <CPSTHEN> and <CPSELSE>. There is currently no general expression handler, but several useful equalities/inequalities are accommodated. Example 6 shows how these tags may be combined, in this case by displaying a welcome message, or a registration form, depending on whether or not the user has been recognized by the system.

```
<CPSIF EXISTS="/system/user">
 <CPSTHEN>
  Hello <CPSTEXT SRC="/system/user/
                          firstname"/>
 </CPSTHEN>
 <CPSELSE>
  <CPSINC SRC="/standard-components/
                          register"/>
 </CPSELSE>
</CPSIF>
```

Example 6: CPSIF, CPSTHEN and CPSELSE

Multipublish

The <CPSMULTI> tag provides explicit support for *Monterey*'s multipublish functionality as described in the earlier section on Content Components.

```
<CPSMULTI SRC="./IE4-version.html"
     DST="*/../IE4-version.html"/>
<CPSMULTI SRC="./NS4-version.html"
     DST="*/../WebTV-version.html"/>
```

Example 7: CPSMULTI

The two <CPSMULTI> elements shown in Example 7 will create two further pages whenever the template is used, based on Internet Explorer 4 and WebTV specific templates respectively.

Content syndication

The manifold content components stored within the *Monterey VFS* represents a valuable asset which can also be re-used externally. By allowing content components to be distributed automatically to third parties outside the *beeb.com* environment, content syndication is a natural extension of the internal content re-use philosophy. Content syndication can provide an alternative revenue stream and broadens the reach of *beeb.com* brands beyond the core websites.

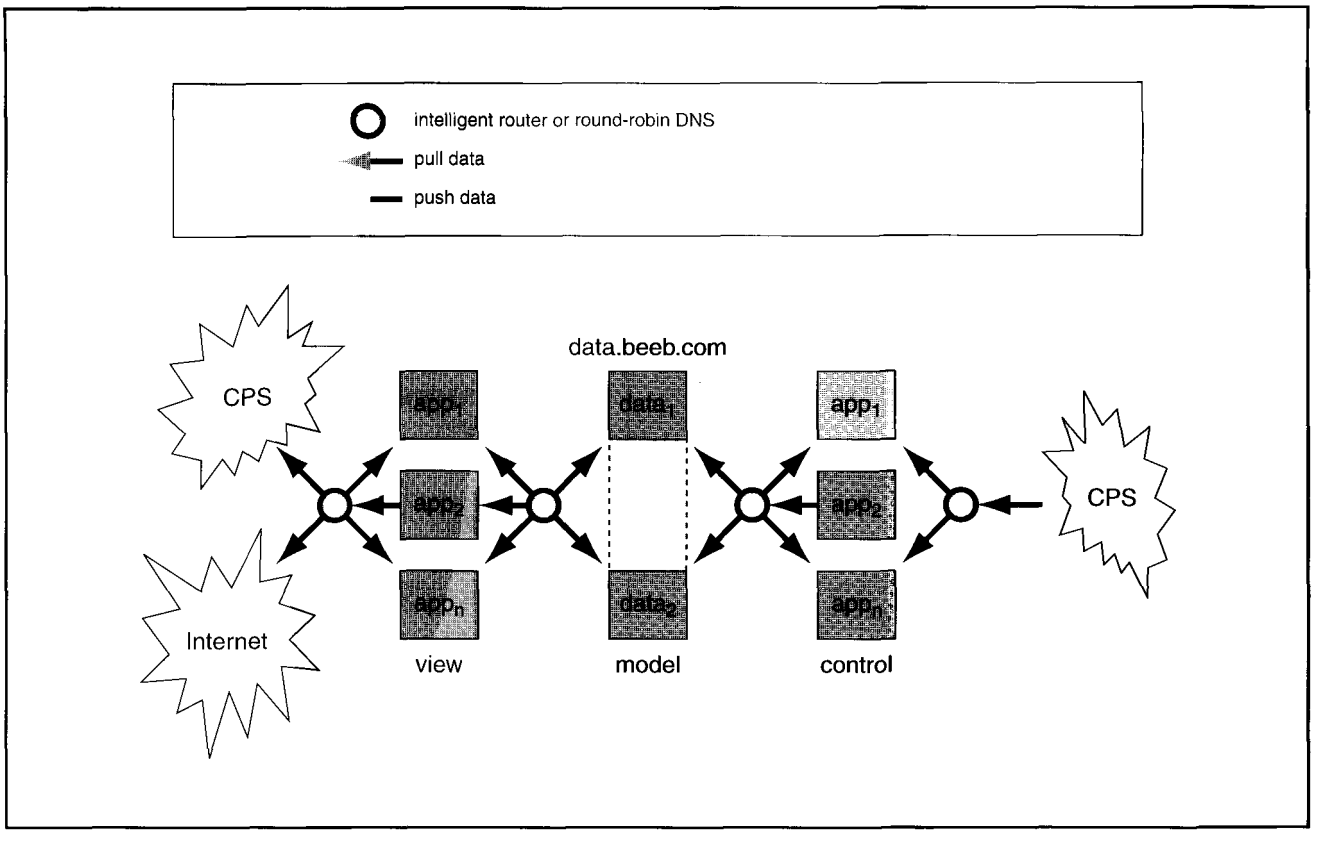


Figure 8: High Availability Architecture

The *Monterey* publish-subscribe mechanisms, in conjunction with *multipublish* facilities, provide a framework for supporting a wide range of subscriber types. Subscriber types currently supported include HTTP pull and FTP push.

Edwards Fig 8

High Availability Architecture

Monterey has been designed to take full advantage of the High Availability Architecture (Figure 8) that has been put in place at the *beeb.com* web server farm.

Data layer

Resilience at the data layer is provided via Oracle replication across two mirrored servers. A detailed discussion of this service is beyond the scope of this paper; suffice to say that the architecture guarantees an extremely reliable, high-performance platform for the *Monterey VFS*.

Application Server layer

Resilience at the application layer is achieved by using multiple application servers in conjunction with an intelligent router. Thin client requests are distributed across application servers by an intelligent router [Dyer, 98]. The router may use a range of load-balancing algorithms, for example "round robin" or packet latency.

Application servers communicate with the Data Server using a variety of protocols, including JDBC and HTTP.

The *Monterey* design also fits in very closely with the high availability architecture put in place for the *beeb.com* Internet presence. The application layer operates mainly in pull-mode, which allows applications servers to be brought on- and off-line (for load balancing and maintenance) without incurring service outage or synchronisation problems.

Intelligent cache

In addition to static web pages, most modern web sites have a large volume of rapidly changing dynamic content, for example search results, personalization and news.

Every time a page is built dynamically, a program must be executed on the web server. Usually the program runs a query to retrieve some information from a database, then decorates this content with HTML and finally sends the completed page back to the client web browser. Creating pages dy-

namically (via CGI, or servlets, etc.) places far more strain on the web server than serving static pages, which use highly optimized code to read and serve pages directly from file-system or memory. In fact, the performance characteristics of static versus dynamic page generation may differ by at least an order of magnitude, sometimes much more. This characteristic naturally reflects on the number of concurrent requests a given hardware platform can support.

Sometimes, every dynamically generated page is truly unique, for example, search results. In such cases, the only way to improve performance is to upgrade hardware or fine tune the algorithms and architecture employed. However, many apparently unique dynamic pages are nowhere near as singular as they look.

One example is the *beeb.com Radio Times* active channel page that displays "now and next" schedules for a range of channels and regions, based on the current time. On the face of it, this would appear to be a fairly dynamic page. However, the schedule timings only operate on a granularity of five minutes, whereas the page may be requested many thousands of times in that period. Therefore, a computationally expensive program is running thousands of times in a five-minute period, when it only needs to run once. This observation is of particular significance to *Monterey*, as all content is effectively dynamic.

The obvious solution to this problem is to use a *cache*. A cache is a fast temporary store for frequently used data. In the case of the *Radio Times* problem, a cache would be used to store the result of the first request to the dynamic page generator; subsequent requests would then be served from the cache. Once the cached results become invalid (i.e. at the beginning of a new five minute period) the contents of the cache would be retired and the dynamic page generator re-run. The fastest cache implementation would be to store the pages directly in memory. However, a cache is beneficial provided it can operate more efficiently than that which it replaces, so a file based cache would also be effective.

Proxy web servers provide some basic cache functionality and are extremely useful in reducing the amount of end-to-end traffic required to service a request. However, proxy servers only work well with static requests since they can only retire content (that is, remove it from the cache) using a fairly basic set of rules, such as age or absolute time. More complex expiry requirements, such as those described in the Radio Times example, would not be possible.

The dynamic caching architecture is similar to a proxy server in that it can prevent the end-to-end generation and transmission of content for each and every request. However, unlike a proxy server, each application server running the intelligent cache is able to share some of the processing load.

A proxy server is only able to match requests with previously cached content if the request pattern matches fairly closely, if not exactly, with the cache key. By sharing processing load, and providing access to back-end services, the server architecture is able to use request signature derivatives to key into cache content. For example, in the case of a *Radio Times* TV listing, the server uses the postcode cookie to determine the television region, then uses this as a cache key.

Standards

The considered use of standards-based technologies within *Monterey* promotes the re-use of third party design decisions and implementations.

The use of standards offsets the risk inherent in building (rather than buying) an application, by promoting inter-operability both present and future. It also provides free off-the-shelf design decisions and, in the case of Java, a wide-range of ready-made implementations as well.

Some contemporary web standards, for example XSL and XLINK were considered to be too unstable to incorporate into the first version of the *Monterey* system. However, the underlying architecture is capable of incorporating these technologies once they become stable.

The current range of XML-based technologies—some still just at draft status—may indeed eventually provide a solution , but they are still in flux and are not yet capable of providing an integrated content production environment.

Pilot work is currently underway which uses CPML templates as XML-aggregators, for subsequent rendering via an XSL stylesheet processor.

Technology	Benefits
Virtual Filesystem	Provides an enhanced filesystem metaphor based on enterprise-class storage (Oracle8)
CPML	Promotes component re-use, while retaining much of HTML's simplicity
Standard thin client services	Supports remote, cross-platform operation with reduced support costs and better scalability
Open standards	Reduces ongoing development effort and promotes interoperability, both now and in the future
First-class interface	Leverages infrastructure and development investment across the organisation

Table 2: Technology Summary

This will allow *Monterey* to take full advantage of stylesheet technology, while retaining the benefits of a cross-platform, component based architecture with integrated content storage facilities.

Technology summary

Table 2 lists the principal technologies used in *Monterey*, together with the benefits gained.

Summary and Future Developments

Monterey has been developed to satisfy the production requirements of an entertainment web site. It is a third generation system at *beeb.com* and draws on the extensive experience of the Production and Applications Programming teams. The system now at version 2 has been in development since February 1998 by two members of the Applications Programming Team.

DSDM has been applied in the latter stages of the project to ensure tight integration of system features with business and the users needs. A regular roll-out of releases to coincide with Webzine launches has ensured the system has delivered early benefit to the production teams and has kept track of business and the users' changing requirements.

Version 2 is currently being rolled-out across the entire *beeb.com* production process.

The architectural approach of *Monterey* has been to avoid re-inventing any wheels by being highly modular and seeking to integrate and add value to third party systems. Public domain standards have been used with proprietary interfaces being used only when necessary. This has reduced the overall time and cost of development and ensured that the system can evolve apace as the industry innovates. This should allow *Monterey* to grow easily and continue to meet the demands of *beeb.com* without becoming a legacy system!

We are still in the early days of the medium and there currently are only a few commercial products in this area. These products only address highly structured news style production and do not have the flexibility for the diverse needs of an entertainment site.

Monterey is increasing the productivity of *beeb.com* production staff while driving down the cost to produce each web page and reducing time to market.

The next version of *Monterey* is now being developed. The main areas of work are:

- Content Syndication—possibly based on the W3C ICE (Internet Content Exchange) submission
- Extended access control and security to add workflow and allow contributors external to *beeb.com* to use the system
- Support for 'high availability' in the *Monterey* application itself and in the deployment of content to multiple content servers

- Ongoing performance and usability improvements based on a rapidly growing user population.

Future developments may include:

- A new user interface built completely in *Monterey* templates
- Full support for XML (eXtended Mark-up Language) and XSL (XML style sheets)
- Migration of the Virtual File System to the Oracle IFS (Internet File System) or an XML data server.

Acknowledgements

The *Monterey* Development team consisted of:

John Edwards	Senior Applications Architect
Phillip Wells	Systems Engineer
Hugh Steele	Applications Programming Manager

Any reader seeking further information on Monterey should contact either Hugh Steele or Martin Goodier at ICL, Bracknell.

Bibliography

BENNER, K., "Tradeoffs in Packaging Reusable Assets", The proceedings of the Sixth Annual Workshop on Software Reuse, 1994. ftp://gandalf.umcs.maine.edu/pub/WISR/wisr7/proceedings/txt/benner.txt

DYER, N., "An Architecture for Commercial Online Internet Services," ICL Systems Journal, Autumn 1998.

KAY, M., "SAXON", http://home.iclweb.com/icl2/mhkay/saxon.html

ORFALI, R., HARKEY, D., "Client/Server Programming with JAVA and CORBA Second Edition", Wiley Computer Publishing, 1998.

WELLS, P., "Templating Language User Guide", ICL */beeb.com* Internal Document, 1999.

Additional Material may be obtained from the following Web sites:

HyperText Markup Language, http://www.w3.org/MarkUp/

Extensible Markup Language, http://www.w3.org/XML/

Web Style Sheets, http://www.w3.org/Style/

Cascading Style Sheets (CSS), http://www.w3.org/Style/CSS/
Extensible Stylesheet Language (XSL), http://www.w3.org/Style/XSL/

Active Server Pages (ASP), http://msdn.microsoft.com/workshop//server/asp/aspfeat.asp

Java Servlet API, http://www.javasoft.com/products/servlet/2.1/

Biographies

John Edwards
John Edwards joined ICL in 1997 and is currently the Senior Applications Architect of 'beeb.com', based at the BBC Television Centre.

Since 1991, he has followed a variety of technical roles at financial and media institutions, including BZW, Reuters and Robert Fleming. He took a one-year sabbatical in 1995 and graduated from De Montfort University with an MSc in Human Computer Systems and AI.

Hugh Steele
Hugh Steele joined ICL (and beeb.com) in December 1996 as Applications Programming Manager. He had an extensive background in multimedia and software development before joining ICL. Former positions were:

- Project Leader for Midi systems development with Voyetra Technologies based in New York
- Technical Director of Visual Data Systems, a London based interactive video tools and systems house
- Development Manager at Galileo, an international travel reservations systems company
- Product Manager at Virtuality Ltd. responsible for VRML and VR systems development (in partnership with IBM and Philips).

Cochise-2: An Integration Solution Kit for TPMS Applications

Michael Royster

ICL, Manchester, UK

Abstract

Over the past 20 years huge investments have been made by organisations in the development of applications based on ICL's VME mainframe operating system and associated Transaction Processing Management System (TPMS). Having made these investments organisations are now looking to leverage this when building new business solutions. Today these solutions are typically built on, or utilize, Microsoft technology. Cochise-2 provides a solution toolkit to allow a TPMS application to be accessed easily and exploited from within a Microsoft Windows based environment.

This paper provides an overview of Cochise-2 and includes a description of the problems it is addressing, application scenarios and deployment. The Cochise programming model is also introduced.

Cochise-2 was highly recommended in its entry to the 1998 ICL Innovation awards.

Introduction

Organisations world-wide are looking at better ways to leverage major existing applications and to incorporate new technology. Replacing or re-engineering major applications are not viewed as cost effective and integration has been proved as the best strategy for addressing this problem. Microsoft is firmly established as the world's leading software supplier and is continually providing pervasive new technologies which organisations wish to exploit. ICL's VME customers are no exception to this global trend and they too are looking for ways of leveraging their existing investments in VME when building today's business solutions.

The majority of VME mainframe applications which were developed using TPMS provided a complete environment within the VME operating system for managing transaction processing from on-line terminals. Cochise-2 provides the ability to access TPMS applications and incorporate them in today's business solutions built upon Windows DNA, Microsoft's architectural framework for building modern, scalable, multi-tier distributed computing solutions.

Using Cochise-2 from within the Windows DNA framework, it is possible to access a VME application in a manner identical to accessing any other data or application through the unified access mechanism of the Component Object Model (COM). Cochise-2 requires no changes to the existing VME TPMS application and allows customers to consolidate their development skills in the Microsoft Windows skill-set. For example, a developer with the relevant skills to access a database could now integrate with a TPMS application with no further training.

Cochise-2 was developed from the original Cochise product that provided Web Browser access to TPMS applications over the Intranet/Internet using Java based technology. The original Cochise Java source code underwent very limited changes during the development of Cochise-2; the development essentially consisting of providing a sophisticated COM wrapper around existing functionality.

Business Scenarios

Business Problems

In today's business environment just a few of the problems faced by IT departments are:

Software is providing competitive advantages

- need to develop systems quickly and incrementally
- need to protect business assets (knowledge)

Requirements change rapidly

- need applications which can be modified incrementally
- need code which can be reused easily

New applications are demanding

- highly distributed (WAN Internet)
- potentially millions of (unknown) users
- connections are non permanent, low speed
- connect multiple data sources

Developers are expensive

- demand exceeds supply
- need to leverage existing skills
- need to focus on business problem not plumbing

Computing environment is heterogeneous

- existing hardware/software investments must be leveraged.

The Solution Framework

Microsoft's Windows DNA provides a generic framework which can be used to address these problems and provides the flexibility to build or buy software solutions which can be integrated easily with existing PC and mainframe computing investments.

Cochise-2 is fully aligned with Windows DNA (Distributed interNet Applications Architecture) and provides the component within the framework for the integration of existing VME TPMS mainframe applications in today's business solutions. Using Cochise to integrate VME TPMS applications requires no changes to the mainframe application and allows you to leverage your existing VME hardware and software investments.

Windows DNA

The Microsoft Windows Distributed interNet Applications Architecture is an architectural framework for building modern, scalable, multi-tier distributed computing solutions which can be delivered over any network. Windows DNA provides a unified architecture which integrates the worlds of client/server and Web-based application development (see Figure 1).

Windows DNA addresses requirements at all tiers of modern distributed applications: user interface and navigation, business process, and storage. Like the familiar PC environment, Windows

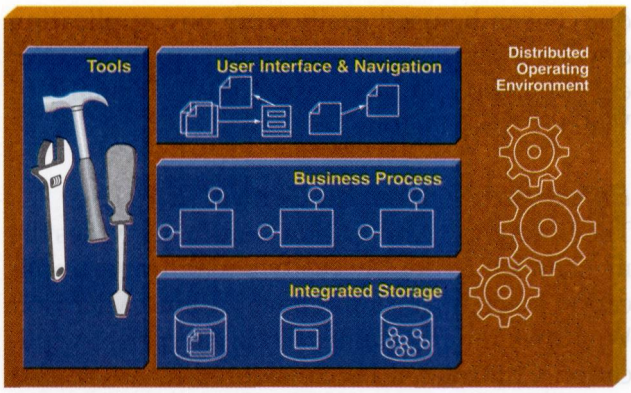


Figure 1: Windows DNA Architecture

DNA enables developers to build tightly integrated applications by accessing a rich set of application services in the Windows platform using a wide range of familiar tools. These services are exposed in a unified way through the Component Object Model (COM).

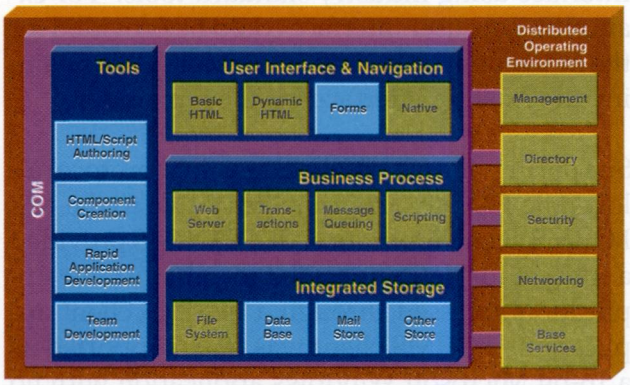


Figure 2: Windows DNA Services

Windows DNA provides a road-map for creating successful solutions which build on existing computing investments and take them into the future. Using Windows DNA, any developer will be able to build or extend existing applications to combine the power and richness of the PC, the robustness of client/server computing and the universal reach and global communications capabilities of the Internet (see Figure 2).

The Cochise integration solution kit comprises the following components.

Enterprise Application Objects (EAO)

Enterprise Application Objects (EAO) allow you to develop applications which access and manipulate TPMS applications running on a mainframe. The Enterprise Application Objects provide an object representation of a mainframe application allowing it to be driven programmatically. As Microsoft COM interfaces, the EAO

objects can be used easily from any application development environment that supports COM. e.g. Microsoft Office, Visual Basic, Visual C++, Visual J++ and Delphi.

Enterprise Application Control (EAC)

The Enterprise Application Control provides the mechanism to deploy easily a mainframe application's existing user interface within a Microsoft Windows environment. The Enterprise Application Control can be considered as a terminal emulator ActiveX control, supporting the ICL 7561 and FORMS terminal protocols.

The EAC is built on top of EAO, and provides a highly flexible architecture. Using the two components in conjunction, the presentation of an application can be incrementally migrated from its original character based interface to a new user interface using industry standard tools. For example a new user interface could be provided for the most commonly used TP screens while retaining the existing interface for the remainder.

Supporting the ICL FORMS terminal protocol allows the EAC to act as an alternative to the ICL FORMS client application. ICL FORMS provided a tool set for defining a GUI to a mainframe application accessed through a FORMS client application. FORMS client was only supported as a 16-bit Microsoft Windows application with no plans to offer 32-bit support. The EAC and Cochise-2 provide the strategic 32-bit replacement of the FORMS client.

Cochise HTML Generator

The Cochise HTML generator provides a mechanism to provide easy access to a TPMS application over the Internet/Intranet using a standard Web Browser. The TPMS application is simply presented to the user as a series of HTML forms.

The HTML generator uses standard Microsoft Active Server Pages to generate the HTML and process the HTTP requests. Like the EAC the HTML generator is built upon the foundation of the EAO library. In essence the current mainframe screen, as held by EAO, is converted to HTML and delivered to the browser. After the user has completed the form the subsequent HTTP request is then mapped to the EAO representation and the data passed back to the mainframe. As was the case with the EAC, using this architecture you can incrementally migrate the presentation of an application from its original character based interface to a richer web based interface using industry standard tools.

Cochise Java Applet

The Cochise Java applet provides high performance Web Browser access to TPMS applications over the Internet or corporate Intranet. It enables organisations to exploit the internet for business trading, for example, opening up core applications to external 'customers', whilst helping to reduce the internal cost of ownership of desktop computing. The Cochise Java Applet is basically a Java terminal emulator supporting the ICL 7561 and ICL FORMS terminal protocols and has the capability as provided by Cochise release 1.

Cochise Server

The Cochise Server manages the connection with the mainframe and provides a concentrator capability for Cochise clients; i.e., users of EAO, EAC, or the Cochise Java applet. In addition to this 3-tier architecture, you can deploy the Cochise Server in a traditional 2-tier model by installing it on the same platform as the EAO or EAC client. The Cochise server also incorporates an administration facility for configuring and managing the server.

Figure 3 illustrates how the respective components forming the Cochise Solution Kit relate to each other and the Windows DNA architecture. The figure shows the user interface linking directly to EAO for illustration purposes. However in a typical application architecture the user interface components would access business objects which in turn utilize EAO.

Cochise Solutions

The following sections outline some of the potential business solutions in which Cochise-2 could be used.

Application Re-engineering

Using Cochise's Enterprise Application Objects (EAO), core applications can be retained on the mainframe, with all the benefits this brings with respect to availability, security and reliability, and yet be able to exploit the benefits of the Microsoft Windows interface for data presentation. With no modifications to the mainframe application, the user interface can be re-engineered using the industry standard development tool of choice,

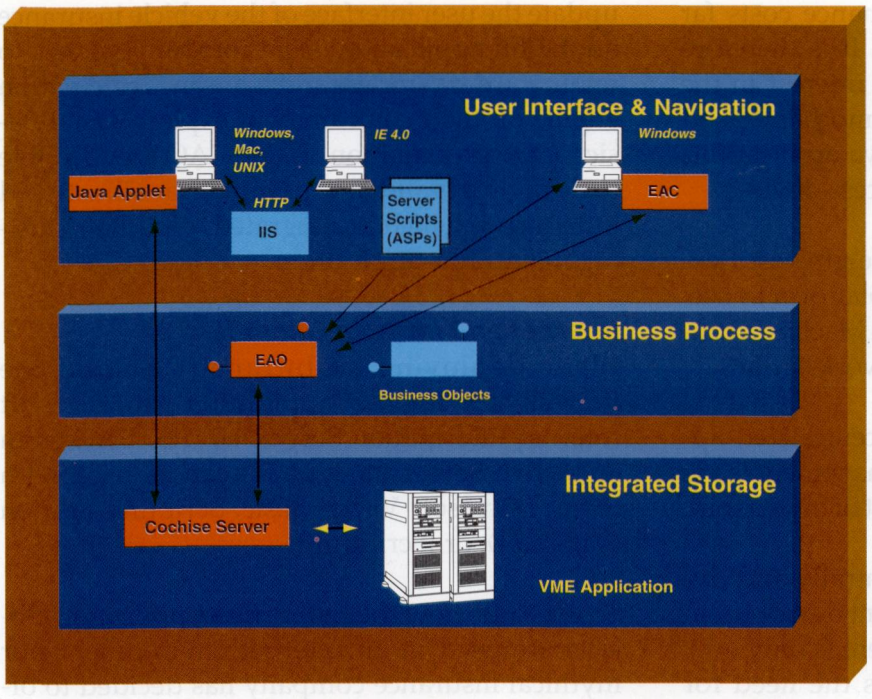


Figure 3: Cochise Integration Solution Kit relationship to DNA

such as Microsoft Visual Basic, Visual C++, and Borland Delphi.

Unlike many solutions Cochise does not require "big bang" application re-engineering but allows the user to re-engineer applications incrementally. Using the Enterprise Application Control (EAC), in conjunction with EAO, an application's presentation can be migrated incrementally. For example, a new user interface could be provided for the most commonly used TP screens while retaining the existing interface for the remainder.

Enabling Global Reach

Cochise provides the capability of creating solutions that fully exploit the global reach and "on demand" communication capabilities of the Internet, while preserving existing investments in applications and data.

Using EAO in conjunction with Microsoft's Active Server Pages (ASP) an application's reach can be extended via the Internet. For example, external, and potentially unknown, users can take advantage of the functionality of an existing mainframe application through a modern browser interface.

Building a Component Architecture

Many organisations are now seeing the benefits of adopting a component architecture and assembling their business applications from component parts. Utilizing Microsoft's component backbone, COM, Cochise's EAO allows users to encapsulate existing application's functionality into business objects. These business objects can then be assembled into flexible business solutions that can easily be adapted to rapidly changing requirements.

Although Cochise does not support transaction co-ordination with TPMS, business objects built using EAO within Microsoft's Transaction Server (MTS) can still be deployed to take advantage of MTS's "roles", component packaging and pooling of database connections.

Productivity Improvements

Using the power of COM and the simplicity of the EAO object model, end-user productivity improvements can easily be made.

Repetitive end-user interactions with a mainframe application can be automated using any scripting environment which supports COM, for example, Visual Basic for Applications (VBA), supported by Microsoft Office, or the Windows Scripting Host.

Terminal Emulator Replacement

Cochise-2 removes the need to deploy existing ICL 7561 (or ICL FORMS Client) terminal emulators across an organisation. Using the Enterprise Application Control (EAC), mainframe access capability can be embedded within other "line of business" applications. For example, users, who typically use Microsoft Excel as their main business application, can gain mainframe access without having to leave Excel to invoke a separate terminal emulator application.

Alternatively, the Cochise Java Applet could be deployed as a Terminal emulator replacement.

Web Browser Access to enterprise applications

Cochise-2 also provides high performance Web Browser access to enterprise applications over the internet or corporate intranet.

Typical PC support and maintenance costs far outweigh initial purchase costs. It is therefore not surprising that there is such interest in the new client/server computing models based around platform independent Java applications and Network Computer (NC) access devices.

Written in Java, Cochise gives the greatest flexibility for choosing which client platform to use, from ultra thin NCs to Java Virtual Machines running on Windows PCs or UNIX workstations.

Combining 'thin clients' with a centralized approach to administration and management leads to significant reductions in the cost of ownership.

The client Java applet is held centrally and dynamically downloaded to the client device upon connection. This ensures that there is only ever one version in use and removes the need for costly software distribution and management.

Application Scenarios

The examples below show how an existing "Green Screen" TP application can be left completely unchanged and incorporated in a new business solution utilizing the Enterprise Application Objects. A number of application architectures are illustrated:

- Microsoft Windows Client
- Internet and Intranet Client/Server
- Component Architecture
- Microsoft Office Integration.

In illustrating how EAO can be used in each of these application environments a mythical insurance company's motor vehicle insurance quotation application will be used. This standard TPMS application provides insurance quotations where a user supplies, through a series of data entry screens, details of the driver and vehicle to be insured and the output is a quotation for insurance. Figure 4 illustrates one of the "green screen" data entry forms capturing details of the vehicle to be insured.

Microsoft Windows Client

The following illustrates how the Enterprise Application Objects can be used in the environment of our mythical insurance company, *Ambassador Insurance*. *Ambassador Insurance* has decided to update the user interface of the vehicle insurance quotation application. The company wishes to retain the data storage and insurance calculation engine on the mainframe but exploit Visual Basic for the presentation. Using EAO they are able to develop a brand new user interface illustrated in Figure 5, which is integrated with their existing mainframe application.

Internet and Intranet Client/Server

When deploying an Intranet ("behind the firewall") or Internet application via Microsoft Internet Information Server 4.0 (IIS) EAO from an Active Server Page (ASP) can be used with either Microsoft Visual Basic, Scripting Edition (VBScript), or JScript(tm).

To illustrate how the Enterprise Application Objects are used in an Internet environment our mythical insurance company has decided to offer a motor insurance quotation service on the Internet. Using a series of Active Server Pages that utilize EAO, our insurance company is able to build its Internet quotation service. Figure 6 illustrates the view of the service the customer would see.

Component Architecture

To illustrate how EAO can be used to build business objects to underpin a component architecture, our mythical insurance company could develop a higher level Insurance Quotation object. This same object could then be used in both the Visual Basic application and within the Active Server Pages.

Microsoft Office Integration

EAO can be driven from Visual Basic for Applications (VBA), the standard macro language of Microsoft Office, allowing simple integration between a Microsoft Office application and a mainframe application. For example, just a few of the potential benefits of the integration for our mythical insurance company are:

- Populating a Microsoft Word document with data from a mainframe application, for example, generating quotation confirmation letters
- Invoking the mainframe application based on a cell changing in a spreadsheet. For example, our insurance company uses an Excel spreadsheet to compare its prices with that of its competitors and, using triggers in the

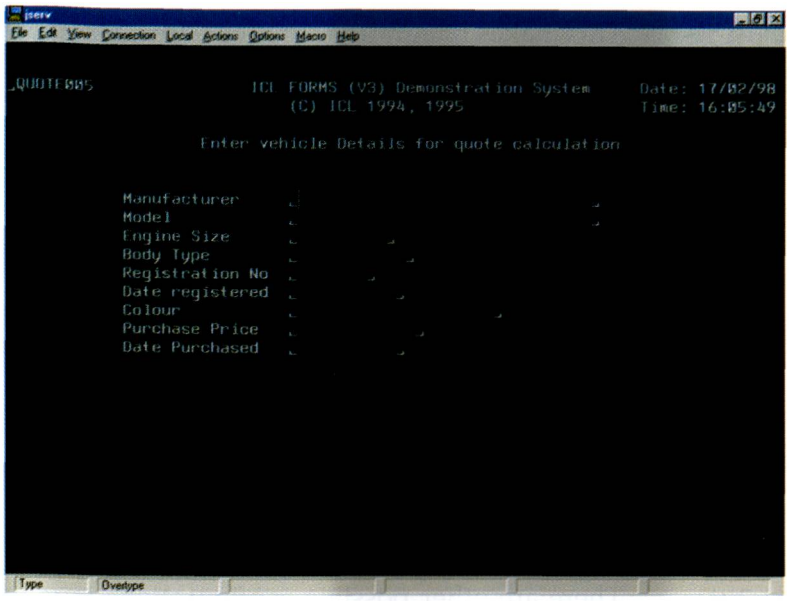


Figure 4: Existing Application Data Entry Screen (QUOTE005)

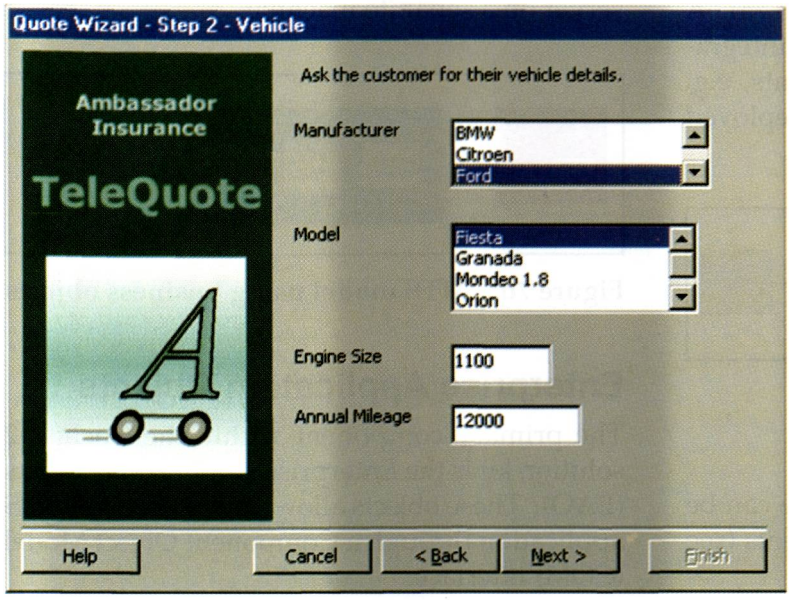


Figure 5: New Quotation Application

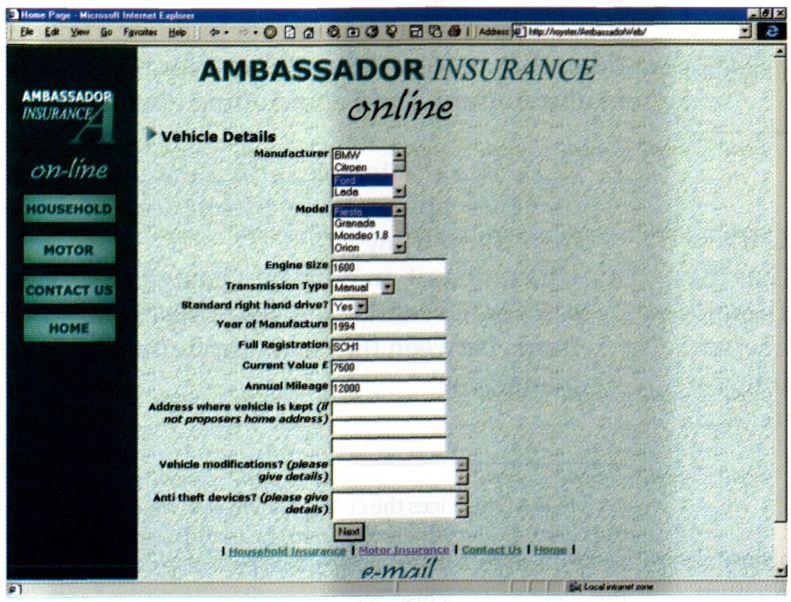


Figure 6: New Internet Application

spreadsheet, is automatically able to adjust its prices held in the mainframe application.

Flexible Deployment

Cochise provides a flexible deployment model and can be deployed in either a 3-tier or traditional 2-tier architecture. The choice of deployment model depends on a number of factors, including:

- The number of supported users
- The client and server hardware specifications
- Integration requirements.

2-tier

Cochise can be deployed in a traditional 2-tier architecture with EAO/EAC and Cochise Server installed on the client system (see Figure 7a). A disadvantage of this deployment is that integration is difficult because any components, e.g. postcode checking software, must be deployed on each client machine.

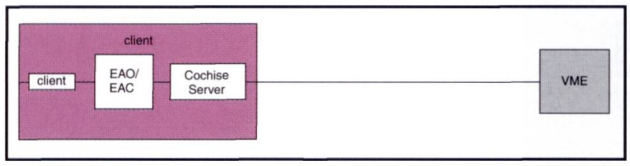


Figure 7a: 2-tier architecture

3-tier

Through the flexibility of COM Cochise can be deployed in a variety of 3-tier deployments (see Figure 7b).

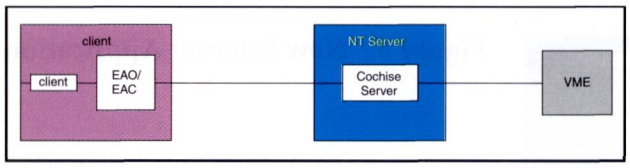


Figure 7b: 2-tier architecture

EAO/EAC could be on the client and the Cochise server kept on a server. This deployment has the advantage that the EAO uses the efficient changed field protocol when talking to the server. However, this architecture also makes integration more difficult as the EAO is installed on each client.

Using DCOM

Another alternative is to move the EAO object to the server and drive it directly from the client. This makes integration simpler. However, there would be an increased cost in the number of DCOM calls between the client and server due to the simple nature of the interface (Figure 7c). One final deployment is the MTS model using business objects (see Figure 7d). This reduces the COM traffic between the client and the server.

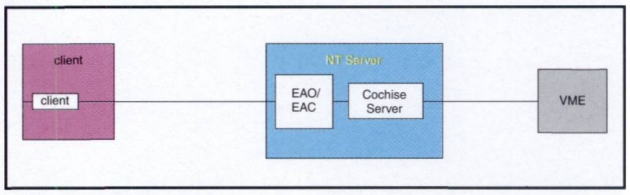


Figure 7c: DCOM

There are other possible configurations involving Business objects in MTS using a Cochise server on a separate machine.

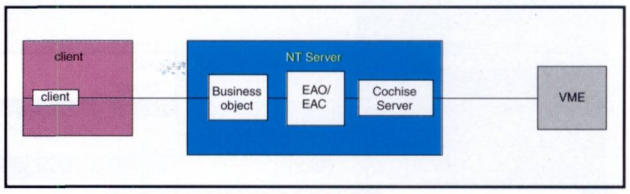


Figure 7d: MTS model using business objects

Enterprise Application Objects

The primary component within the Cochise-2 solution kit is the Enterprise Application Objects (EAO). These objects allow you to drive a TPMS application through a Component Object Model (COM) interface.

EAO Features

EAO supports key features for building applications that require integration with an existing mainframe application, including the following:

- COM based interface, providing integration of TPMS applications within any environment supporting COM
- Utilization of an efficient Change Field Protocol based connection, reducing the network traffic between the client application and the mainframe[1]

[1] For FORMS services the changed field protocol is used from the client all the way through to the mainframe, in the case of 7561 services the change field protocol is used between the client and Cochise Server.

- Supports validation of data at the client using validation rules defined by the mainframe application, reducing mainframe interactions
- Supports Synchronous and Asynchronous event based programming models.

EAO Object Model

The EAO object model is similar in style to Microsoft(r) ActiveX(tm) Data Objects (ADO). This similarity is deliberate to allow any user familiar with ADO to exploit EAO quickly. The only externally creatable object is the Connection object, all the other objects are accessed through the hierarchy (see Figure 8).

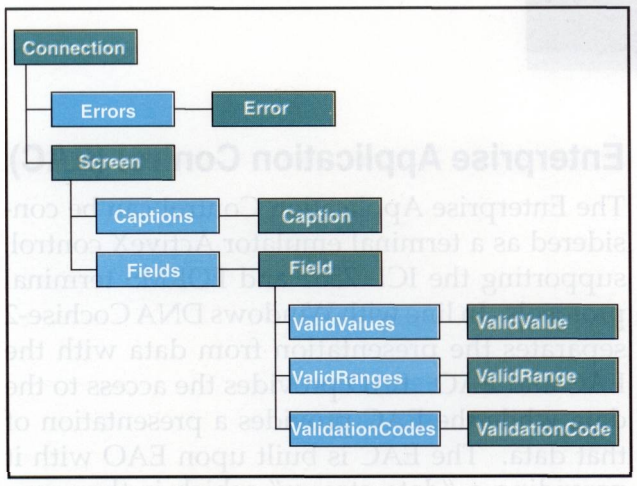


Figure 8: EAO Object Model

The primary objects in the EAO object model are as follows.

Connection: Maintains connection information with the mainframe TP Service.

Screen: A representation of the current "screen" in the mainframe application. This object provides a representation of the "screen" which users of an application would see if they were accessing the application from a terminal or were a FORMS client. Using the methods and properties you can drive the mainframe application from a program.

Field: Contains information about a single data entry field on a screen. The Screen object includes a collection containing all the fields on the screen.

Caption: Contains information about a single piece of background text on a screen. The screen object includes a collection containing all the captions on the screen.

Error: Contains extended error information about an error condition.

ValidValue, ValidRange, and ValidationCode: Contains information regarding the validation rules that apply to a field. A field object includes a collection of validation rule which apply to it. These rules are supported for FORMS applications only.

Each of these objects supports a set of properties and methods with which you can use to manipulate the object and its content. The primary EAO object is the Screen which represents the current screen in the application. Figure 9 illustrates how the screen object's Name property and Fields and Captions collection are populated based on the underlying "green screen".

Using the EAO Objects

To use the EAO objects to access an existing mainframe application you simply perform the following tasks.

1. Create an EAO **Connection** object.
2. Open a connection to the required mainframe application using the **OpenConnection** method on the **Connection** object.
3. On the successful opening of a connection a **Screen** object is created which represents the 1st screen, typically a login screen. Using the Screen object you can:
   - Retrieve background text from the **Captions** collection
   - Set field values using the **Fields** collection
   - Send data back to the mainframe using the **Send** method.
4. Repeat the above for the next screen.
5. Close the Connection using the **CloseConnection** method.

EAO in action is illustrated on the next page (Figure 10) where the sample Visual Basic code is listed for connecting to a TPMS application, navigating through an initial menu and entering a vehicle's details into the data entry form introduced in Figure 4.

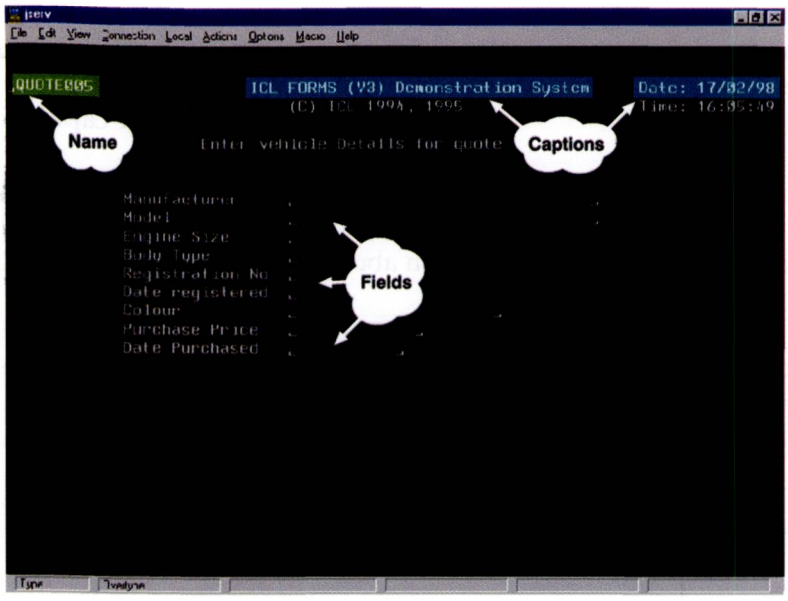


Figure 9: Screen object illustration

```
Public Sub EAOExample()

    Dim cnn1 As ICLEAO.Connection
    ' Open connection
    Set cnn1 = New ICLEAO.Connection
    Cnn1.OpenConnection "Server=galaxy;
                         Port=5004;"

    ' Assuming initial screen is MENU001,
    ' select the insurance quotation
    ' option, '1'
    cnn1.Screen.Fields(1).value = 1

    ' Send Data to the mainframe, this
    ' call uses the default parameters
    ' which equate to the equivalent
    ' of the Send key being pressed.
    cnn1.Screen.Send

    ' Display details of the next screen
    ' received
    Debug.Print "Current Screen:" & cnn1.
                                Screen.Name
    cnn1.Screen.Fields("Manufacturer") =
                                    "Ford"
    cnn1.Screen.Fields("Model") = "Fiesta"
    cnn1.Screen.Fields("Engine Size") =
                                    "1100"
    cnn1.Screen.Fields("Body Type") =
                                    "Hatch"
    cnn1.Screen.Fields("Registration No")
                                = "L519RMP"
    cnn1.Screen.Fields("Date Registered")
                            = "10/10/1994"
    cnn1.Screen.Fields("Colour")= "Red"

    ' Send the data
    cnn1.Screen.Send

    ' Close the connection
    cnn1.CloseConnection
End Sub
```

Figure 10: EAO code example

Enterprise Application Control (EAC)

The Enterprise Application Control can be considered as a terminal emulator ActiveX control, supporting the ICL 7561 and FORMS terminal protocols. In line with Windows DNA Cochise-2 separates the presentation from data with the EAC and EAO. EAO provides the access to the data while the EAC provides a presentation of that data. The EAC is built upon EAO with it providing a "data stream" which is then presented using EAC.

To use the EAC in a business solution, the application is connected using EAO, as described above, and then the EAC's EAOSource property is set to this object. The current screen will then be presented in the control. With this flexible approach mainframe interactions can easily be scripted and new functionality built to surround an existing application.

To illustrate how the EAC can be used in conjunction with the EAO, Figure 11 shows the sample Visual Basic code for connecting to a TPMS application with EAO and presenting the application's character interface through the EAC. The code also illustrates how individual screens can be intercepted with EAO's NewScreen event.

Screen Scraping with a difference

Cochise-2 incorporates a number of unique features when compared to other "screen scraping" technologies available from ICL and other vendors. These features include:

```
Private WithEvents cnn1 as ICLEAO
                                .Connection
Public Sub Form_Load
   Set cnn1 = New ICLEAO.Connection
End Sub

Public Sub ConnectButton_Click()

   cnn1.ConnectionString = "SERVER=
                        paul1;PORT=5000"
   cnn1.ConnectionTimeout = 30
   cnn1.SendOptions = eaoSendASynch
   cnn1.OpenConnection

   ' Pass the open connection to the
   ' invisible control eac1.EAOSource =
                                      cnn1
   eac1.Show = True
End Sub

Public Sub DisconnectButton_Click()
    cnn1.Close
End Sub

' The Application gets this event from
' the Enterprise Application Object
' before the control gets the same
' event, so the application gets a
' chance to intervene before the
' control displays the new screen.

Private Sub cnn1_NewScreen( ByVal str
                              as String)

    ' Are we currently showing the
    ' connection in the control
    If EAC1.Show = True then
        ' About to display screen in
        ' the control but do we want to
        ' take over from the control?
        if cnn1.Screen.Name = MENU001")
          then
          ' Take over the rendition.
          ' switch the control to
          ' invisible
          EAC1.Show = False
        endif
    endif
End Sub
```

Figure 11: EAC code example

- The use of a "changed field protocol" reducing the network traffic to the mainframe
- The TPMS application can be accessed directly through COM making it immediately available within any environment supporting COM. With Cochise-2 there is no "compile" stage or code generation as is often the case with other "screen scraping" technologies
- The application data is made available through the high level EAO object model. This representation of the data presents an abstract representation of the mainframe "screen". This representation enables fields to be accessed via their relative position on the screen, e.g. Fields(1), or via a name derived from any text to the left of the field, e.g. Fields("Surname"). This abstract representation provides an application using EAO with a higher level of resilience to any minor changes made to the screen layout in the future. This is in contrast with many other similar "screen scraping" technologies which expose data through its exact location on the screen.
- No dependence on any third party terminal emulator.

Conclusions

Integration is proving to be the only viable approach to leverage major application functionality and incorporate new technology. ICL's VME customers wish to maximize their existing investments in VME when building today's business solution. Cochise provides a powerful solution kit to allow them meet this challenge.

The Cochise integration approach has been endorsed by ICL's leading customers, including EDS who have chosen Cochise for their Infrastructure 2000 project for the Inland Revenue. This major project involving the rollout of a new Windows NT desktop across all the UK tax offices.

Acknowledgements

All trademarks are acknowledged.

The author wishes to thank Mike Buckhurst, Paul Lloyd, Paul Fitton, Paul Macdonald, Dave Stewart, and Steve Thompson for their contributions both to this paper and to the design of Cochise-2. Most of all though, he wishes to thank all who worked on the Cochise project, in whatever capacity, without whom there would have been no product.

Bibliography

BEAL A, "A World Wide Web interface to TPMS applications", ICL Systems Journal, Volume 11, Issue 2, January 1997.

CHAPPELL D, "How Microsoft Transaction Server Changes the COM Programming Model, Microsoft Systems Journal, January 1998.

DNA Windows(r) DNA Building Windows Applications for the Internet Age, Microsoft Corporation White Paper, October 1998.

ICL, "An Introduction to VME", ICL Publications, 1989.

HOLT N, "The Architecture of Open VME", ICL Publications, 1994.

MICROSOFT, "TechEd 98 Conference papers".

REED D, TREWIN T, TOMSEN M, "Microsoft Transaction Server Helps You Write Scalable, Distributed Internet Apps", Microsoft Systems Journal, August 1997.

WWW.MICROSOFT.COM

Biography

Michael Royster graduated from Staffordshire University with a B.Sc. in Computing Science. Prior to joining ICL in 1989, he worked at IBM's Hursley Park laboratory on the development of tools to support the use of the specification notation 'Z' within the laboratory.

A large part of his early career was spent within ICL's Open Teleservice programme providing an open infrastructure for delivering service electronically. During this period he held a number of development roles covering both UNIX and Microsoft Windows based developments. Since 1994 he has specialised in Microsoft Windows based technologies and in particular COM.

He was the Technical Design Authority for Cohise-2 and is currently working within the ICL Microsoft Solution Centre in Manchester on assignment based projects in similar TDA roles.

Michael Royster is a Chartered Engineer and a Member of the Institution of Electrical Engineers.

Engineering a Knowledge Utility: The National Grid for Learning

Chris Yapp

Information Society Programme

Abstract

It is frequently stated that Information and Communications Technologies (ICTs) are the key drivers of the global economy. What this means, for the IT industries, is that the industry as a whole is now becoming involved in public policy matters on a scale not seen before. Indeed we believe that the success of ICL, the "big picture" of the emerging Global Information Society and the knowledge economy are increasingly interdependent.

What this means in practical terms is that we have to become proactive in public policy and not merely reactive to individual client requirements if we are to achieve our ambitions of "touching one billion individuals" through the services and solutions that we deliver to our clients.

This paper sets out to explain the development of the National Grid for Learning as an exemplar where ICL has been proactive in creating a new marketplace, enabled by technological developments, and where the interaction with public policy goals is critical to the overall success of the project.

Our experience to date leads us to believe that many of the concepts are potentially reusable. Also, it is quite critical that the engineering and technical communities are involved in developing these new concepts at an early stage, if we are to ensure that public policy is well informed, and at the same time that individual projects contribute to the successful implementation of the "big picture".

Background

Early in 1994, a small team was set up within ICL called Project VIEW, the Virtual Interactive Electronic World. The aim of this corporate think tank or "skunk works" was to try to understand what the implications of cheap pervasive communications would be on ICL businesses and to identify new opportunities for the ICL group that would be created by market discontinuities created by new developments in Information and Communications Technologies (ICTs).

It is quite remarkable to look back at the state of the general industry view of the future in 1994. Awareness and utilization of the internet outside the academic community was very low, CD-ROM titles were only just beginning to appear in both the consumer market place and in industry. In fact, the US view was that the Global Information Superhighway was going to be built on the back of a "killer" application, Video-on-Demand.

A "high-level group", which included the CEO of ICL, was called together by Martin Bangemann [Bangemann, 1994], the European Commissioner for Industry, to respond to the American proposals for the superhighway. The resulting European approach differed from that of the US for some very important reasons.

The EU had a much higher unemployment rate than the USA and, in Europe, there was a belief that the new technologies were likely to be a net destroyer of jobs. The Bangemann group was itself an outcome of the seminal European white paper from Jacques Santer, "Europe, Competitiveness and Jobs".

Where the USA focussed on the *information superhighway*, the European approach was to look at the *information society*.

In May 1994, the Bangemann report, "Europe and the Global Information Society", was released

which outlined the key principles of public and private partnerships along with deregulation of telecommunications in Europe in order to foster enhanced european competitiveness as well as economic and social progress.

Some of our early analysis within VIEW was supportive of the European approach. In particular, the Bangemann report highlighted the key role of education and training in making sure that the benefits of technological change were as widely distributed as possible in both industry and society.

In particular, the phrase Lifelong Learning emerged into the political language as the key enabler of a prosperous economy and a cohesive society. What was quite clear was that ICTs were both a cause of some of the problems outlined above and a potential solution to the delivery of education and training.

Translating political rhetoric and aspirations into solutions that could be scaled to become accessible in principle and practice to every citizen of the EU, while respecting national sovereignty, was a daunting task.

The approach taken by myself and my colleague, Malcolm Napier, reflected Malcolm's engineering background with the CEGB and my own involvement with the MIT Sloan School of Management, "Management in the 1990s" programme (MIT90s).

Not having a clear definition of requirements to which to respond made it clear that we would need to work from first principles. For me, the key lesson of MIT90s was that throwing technology at an ill-defined problem could, and often would, make things worse. Two key questions emerged:

- What is the purpose of education and training?
- Why will/must it happen now?

The next section will outline the key features of the answer to these questions before looking at the approach to developing a policy framework within which Information and Communications Technologies (ICTs) could be deployed optimally to create a large-scale infrastructure to support education and training.

The Purposes of Education and Training

While talking to a large number of stakeholders in education and training I discovered a model proposed by Dr Oliver Sparrow as part of scenario work being developed at Chatham House. He explained why education and training were rising up the political agenda around the world.

His ideas felt robust and helped translate some of the words into an economic framework. There were, he suggested three roles which created the potential for either a vicious or a virtuous circle, namely personal growth, social cohesion and economic performance.

One of the tricky problems in this whole field is that everyone knows that education standards impact on economic performance, but trying to create a business case for education is very difficult. How, for instance, do you translate an increase in the number of graduates into an increase in GNP?

If Oliver Sparrow's suggestions were accurate then a look at the cost in quality in identifying under-performance or non-conformance might hold the key to an approach to raising educational performance in an economically justifiable manner.

UK benchmark data from various international sources suggests that there are examples of world-class practice in the UK in all sectors of education; in schools, colleges and universities as well as in industrial training. At the same time, the UK has a long history of under-performance in the lower 40%, notably that the number of adults with no qualifications is higher than in competitor nations.

The UK has had a long history of the use of IT in education at all levels and there are many examples of pilot projects and research reports that demonstrate that, if used appropriately, ICTs could contribute to raised educational standards. The biggest issue that confronts policy makers is to convert pilot schemes into major roll-out programmes.

There are many examples of major social and economic benefits that ICTs can bring, if the problems associated with the economics of national scalability and sustainability can be solved.

I would like to illustrate these by two examples, disabilities and crime.

First, children with disabilities have been shown in many pilot programmes to be able to overcome barriers to learning through the use of email, multimedia technologies and various specialist software and hardware. With new methods of working, it is no longer inevitable that many disabled adults cannot work and contribute to society or that they should be locked into low-skilled work (consider Stephen Hawking or David Blunkett!).

Second, early truancy is linked to criminal activity in the 'teen years. Even in areas of low educational attainment, our own trial at Bristol schools confirmed other research that ICTs in classrooms motivated learning. Also, there is a growing body of evidence that 40% or more of adults in prison have some form of learning difficulty. We do not know, but it is interesting to ponder that if IT were to be used to identify and support children's early learning, what the long-term effect on serious crime would be.

What all this illustrates is that potentially the economic and social benefits of widespread adoption of ICTs in education could be enormous. However, education has a reputation in many countries of being insular, conservative and resistant to change. So addressing the question of why this upheaval would and should happen now is quite central to building a long-term business in learning.

Fundamentally, I believe that there are two driving forces that make change inevitable, while the nature of the change is far from inevitable. These changes are the result of globalisation and the rapid obsolescence of knowledge.

First, technology is facilitating globalisation of economies and many industries. What is clear, looking at the role of IT in supporting globalisation over the next few decades, is that the economic logic of ICTs runs along the following lines.

People are for thinking, machines are for doing. If it can be "done" it can be automated. If it can be automated, in the global economy it can be done somewhere else more cheaply. This implies that, in the long term, for any major economy, the only sustainable new work and jobs are those that rely on human creativity, innovation, design, flair, teamwork and personal service. Looking at manufacturing industry over the last two decades and projecting forward shows higher output created by fewer workers. So the potential for increased worklessness without raising aspirations for education and training is very high and potentially frightening.

At the same time we are witnessing what I would call a reduction in the "half-life of skills". In the aftermath of World War II, there was a long period of stability in the global economy. Out of this came a *linear* life-cycle. In this model, an individual moved from education to training to work to retirement. There was a long period of near full employment and jobs for life existed for a considerable proportion of the population.

This started to break down in the late 1960s. The linear model of life was built on an assumption that a set of skills acquired in youth was sufficient to build a lifetime of employment, with a limited amount of top up training throughout the working life. What we have witnessed in the last thirty years is increasingly rapid and discontinuous change. Education and training throughout a working life is becoming a necessity for an individual to sustain employability. Life has become more episodic and the linear model has been supplanted by a *parallel* life-cycle where life starts in education, moves to training, but then working life has to be juggled with education and training to sustain employability (Figure 1). In Europe the problem has been that we have seen a very high level of unemployment, with the unemployed finding access to training difficult. At the same time, those in work have found working hours growing as well as increasing pressures for flexibility and multi-skilling in many, if not, all sectors of work. Those in employment have generally found access to training easier than those out of work, leading to a greater disparity in income across the skill spectrum, and this situation has been exacerbated by globalisation.

In some technical disciplines within industries concerned with ICTs, it could be argued that the half-life of technical skills is about eighteen months. It is notable that other industries, for example, biotechnologies are also going through a similar pattern of rapid obsolescence of knowledge and skills.

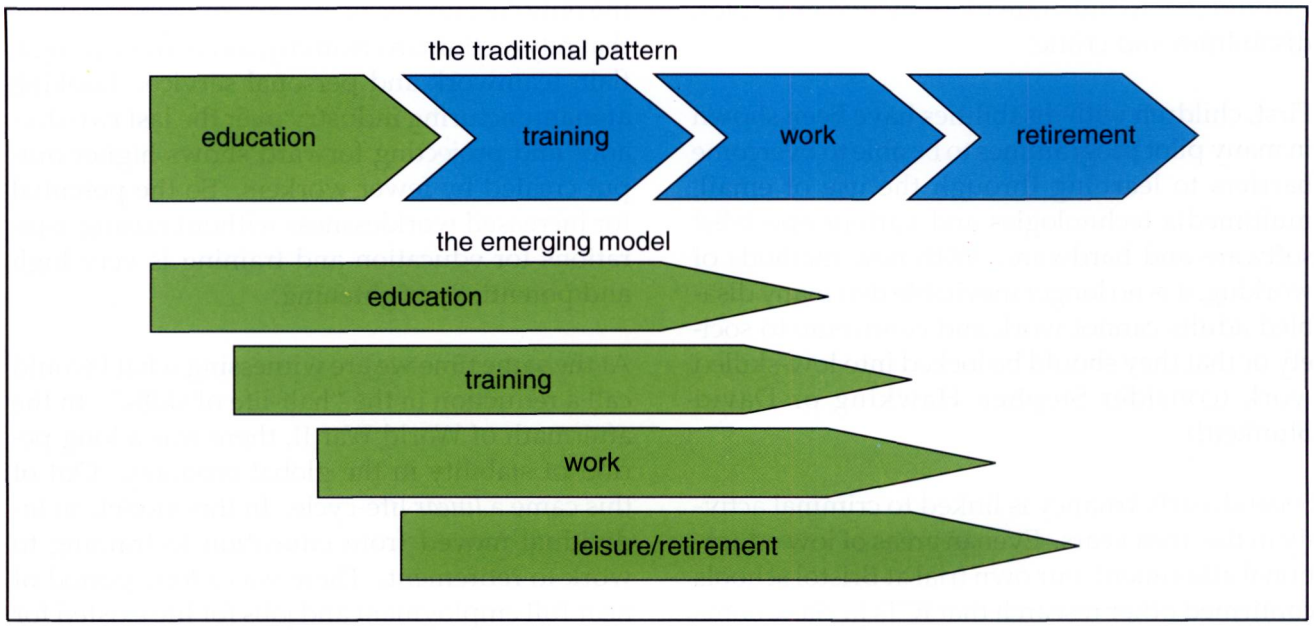


Figure 1: New Life-style Patterns

What I hope I have conveyed so far is a view of the policy issues and an agenda within which to seek a solution to applying technology to underpinning a modernized system of learning. What I will turn to now is the path to a solution.

The Information Utility

What I hope I have been able to illustrate so far is that ICTs are both a cause of and a potential solution to the changing structures of society, education and work.

One of the consequences of this central role for ICTs is that those of us in the ICT industries need to become familiar with the language and detail of public policy. Perhaps I should stress that, unlike the IT industry, the telecommunications industries have, for a long time, been either state monopolies or regulated utilities in the private sector. The opening up of telecommunications has brought with it the advent of competition, but technological convergence has brought with it the issues of regulation and other public policy agendas to a largely free market IT industry.

What characterizes this change for those of us from the "computer industries" is that governments and agencies start to get concerned or involved in a number of ways.

First, competition policy is affected by convergence, knowing where to act to prevent market dominance and abuse. This in turn leads to concern over universal service obligations. If these technologies are as wonderful and important as we claim, then this rapidly translates into a demand for democratization of the technologies, a citizens entitlement to access and pricing regimes that underpin universal access. The flat rate postage service is an example of a universal service obligation.

Telecommunications companies and TV companies have also been regulated over the traffic they carry on such issues as taste and decency. One area where we have experience of regulation has been in the area of data protection and privacy. My own experience has been that this has been treated as a speciality within the IT sector. The changes that I have outlined above are likely to result in this broader public policy agenda becoming part of every day life in ICL and similar companies.

Practically, the real difference that this makes is that in the past we have tended to work from a single customer's view of requirements, or from a market segment, such as retail in-store systems. In the future the interaction between a client's requirements and this broader agenda is likely to grow and in some sectors become central to building an economic solution.

So, the question arises as to what the new converged ICT industries, the internet and digital television are in public policy terms. The first time that we put our ideas on this into the public

domain was in a submission to the House of Lords Select Committee on Science and Technology for their enquiry into the Superhighway.

We suggested that the Internet could best be understood as a series of information utilities, sharing a common infrastructure. Rather than seeing the Internet as a new phenomenon with no precedent, we argued that examination of utility industries, such as water, electricity, gas and telecommunications could provide us with a framework for public and private sector interests to coexist.

In this section, I would like to describe briefly the concept of the information utility (Figure 2) and then look at the case that we have most fully developed, the knowledge utility, the national grid for learning.

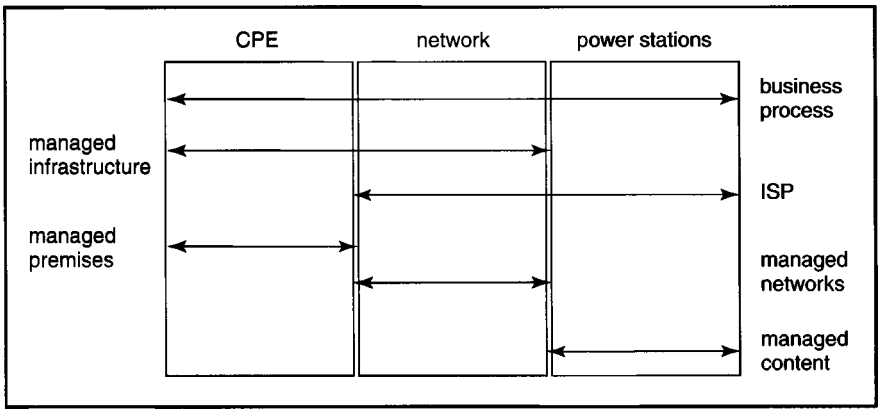


Figure 2: The Information Utility Model

Consider a situation in which we are trying to ensure universal access to the Internet in a country such as the UK. What is clear is that technology is not the problem. The real issue is policy and its deployment.

For example, while ISDN is available now in the UK on a universal basis, broadband is not. The availability of say 2 Mb or 34 Mb networks to every home is many years off. The economics of laying fibre optic cables in rural areas, for instance, is such that although urban areas are now largely cabled, much of the rest of the UK is not. Governments concerned about competition, public-private partnerships etc. are struggling to find a balance between public and private sector interests that will support an economic imperative for universal service. This remains a difficult task in all countries.

Even when we are working purely in the private sector, this will begin to affect us in my estimation over the next five years. For example, consider the widespread adoption of telephone banking. By exploiting the telephone in people's homes, it has been possible to re-engineer banks to use the underlying infrastructure and the growth of call-centres for consumer banking.

At some point, when an application is developed in the commercial sector which requires reliable bandwidth above ISDN capability, such as ADSL or above, then the economics of deployment of that application will be driven by the number of subscribers who have access to the underlying infrastructure.

There is, therefore, a potential win-win for government and industry if a stable framework for universal access can be achieved within a competitive environment. Few private sector companies would wish to see a universal service obligation mandated by government without evidence of long-term sustainability.

When, we switch on a light-switch, turn on the water tap, light the cooker, or pick up a phone we are experiencing an on-demand utility. We take these things for granted. What I would argue is that at some point in the near future economic competitiveness and social cohesion will require that access to information services will be as familiar and as readily available as traditional utilities.

The Information Utility model is essentially a generalization of the other utility models. I believe that it is possible to analyse a utility in three broad components; the customer premises equipment, the distribution network and the power stations. If these are clearly defined and the interfaces between them understood, it is possible to integrate a wide variety of technologies over a long period of time and to allow for innovation in all three areas.

Consider the electricity utility model. When you flick a switch, you are unaware of whether the power is coming from oil, gas or coal fired power

stations or indeed from nuclear or hydroelectric sources. On the internet, we are not sure if the server we are addressing is NT, Unix, Linux or, MacOS for instance. At the same time, if you plug your electric toothbrush into an electric socket, the fact that the system was not designed with this application in mind does not matter. For designers of ICT solutions over the next decade, this level of transparency and integration will be the standard for which to aim.

From a policy perspective, governments know that it is possible to have competitive markets in equipment for each of the three components of the utility. For instance, there are markets for customer premises equipment such as fires or ovens (electricity) or phones or faxes (telecomms).

In the case of the distribution network, the information utility could use for instance telephone lines, fibre optics, the electric cables, satellite, microwave or point-to-point radio. The interoperability and integration of a variety of networked technologies is more likely to achieve an economic model for universal broadband than any single technology on any reasonable timescale. Again, what is clear is that the design of solutions is going to be against a background of permanent transition between underlying technologies.

I have already touched on the different power stations that exist in the world of the electric utility. In the information utility, different power stations may support commercial applications (e-commerce), on-line government, education and training, support for the voluntary sector just to name a few. This in turn leads to the policy framework that could create National Grids for Business, NGfB, National Grids for government, NGfG, a National grid for Learning, NGfL, and a National Grid for Giving or Charity, NGfC.

One of the clear trends in ICTs that we have witnessed over many years is the increasing fuzziness of boundaries. Not long ago a bank was a bank and a shop was a shop, but ICTs have led to organizational drift. In the above models, the public library could easily become an access point for learning, government or economic activity.

While each of the three components can, in principle, support a competitive market in product supply, I think that it also provides us with a model for analysing outsourcing or managed services in an information utility.

It is possible to provide managed premises, managed networks or managed content services (power stations) as well as product supply to each of the three components. There are, I believe, three other service types.

There is the managed infrastructure, which is a service that covers both CPE and the network. The Internet Service Provider, ISP, provides a network and content services. There is also an end to end managed service, which is essentially a business process outsourcing activity.

This taxonomy of managed services needs to be tested more widely, but on discussion with clients in a variety of sectors, I believe it to be a useful tool in developing a shared language about what is meant by managed service.

In turn, it can help us, and clients, to think creatively about the fuzziness issue that I raised earlier. One example that I have given on a number of occasions is an example of the post office counter. I could see, for instance, a pensioner withdrawing her pension and ordering a book from the public library through a kiosk in the Post Office. The book would then be delivered, by the milkman, in the morning. In a rural area, this kind of co-operation or partnership could protect all three services.

I recognize that this is only the briefest of introductions to the concept of an information utility, but rather than stay at this level of abstraction, I wish to turn to the example where this has been most widely developed, the National Grid for Learning.

Community Learning Networks

I believe that the task of describing a system of education and training for the new millennium has to reflect the policy objectives that I have outlined earlier. I would summarize the core features of a successful implementation, as follows:

1. The creation of a *culture* that recognizes the value of and invests in education and training on a lifelong basis

2. *Access* to lifelong learning on a socially inclusive basis
3. The *content and services* to support individual learners
4. A *context* for lifelong learning.

It is important to understand that learning is a social and a socializing process, not a technical experience. It is quite clear that sacking the teachers, closing the schools and pumping learning through PCs and digital TVs is not a sensible approach.

For this reason, connecting every school to the internet or a PC for every child may be a necessary condition for providing an effective modern education system, but it is not a sufficient condition. As I stated earlier, throwing technology at an ill-defined problem, can and often does make things worse. The challenge outlined by my four points above can best be described as re-engineering education to support lifelong learning.

The model that we could find that best fitted the aspirations described above was the Community Learning model. There are many examples of these in the literature and in different cultures around the world. The best-known phrase to encompass this model is now well-known in educational circles, "It takes a whole village to educate a child."

Some of the key characteristics of the new ICTs seemed potentially to offer ways of changing the economics of access to learning. Most notably, what we have seen in other sectors of the economy, when impacted by ICTs, is that previously discrete parts of an organization become blurred and then different organizations' boundaries overlap untill different sectors of the economy overlap. The obvious example of this can be seen with cashback in supermarkets, solely a function of banks a short time ago.

If this is applied to a community learning network, consider the following as examples that ICTs might facilitate.

In a rural area, an adult could study for a college course from a branch of a library. A teacher could access in-service training from a university from a school classroom. Employees of a small firm that could not afford an in-house infrastructure for learning could access a course from a college using a local facility in a chamber of commerce.

In time, as the technologies mature and fall in price, it may be that universal access from the home will be practical. In the meantime, the provision of facilities in the community provide for a level of access that would enable learning to be brought to the learner through technology, rather than the learner having to go to the learning. What I have described is using the technology at the level of the community to provide a system of learning -on-demand. This is "Just-in-Time" (JIT) for learning.

The Knowledge Utility

If the core of the new system of learning is the community learning network, then scaling this up is the challenge. In ICL we started two major projects back in 1994 to gain experience that could help us shape our thinking on the new models. With the South Bristol Learning Network, SBLN, we started what has now become Cyberskills™ to look at how ICT skills could be raised among adults on a widespread base. At the time of writing we now have over fifty Cyberskills clients in a number of countries.

With BT, we invested in a project in Withywood schools in Bristol, Bristol Education on-line, BEON. From this, our first commercial service was developed on Merseyside.

We have now had a number of years experience in developing educational managed services for education of both children and adults. Our learning continues alongside the development of the business. Importantly, we used our experience and commitment in these projects, to approach all the UK political parties with our analysis as outlined above and were pro-active with politicians and civil servants.

The concept of a knowledge utility is essentially outlined Figure 3. First, there are various learning premises, starting with schools, libraries and colleges. In time this will grow to cover other institutions. Some examples that we have seen in various trials include football grounds, cyberpubs and shopping centres. Ultimately home access will occur, but this institutional ap-

proach at least offers the opportunity for every community to have access through shared facilities in a reasonably short time-scale.

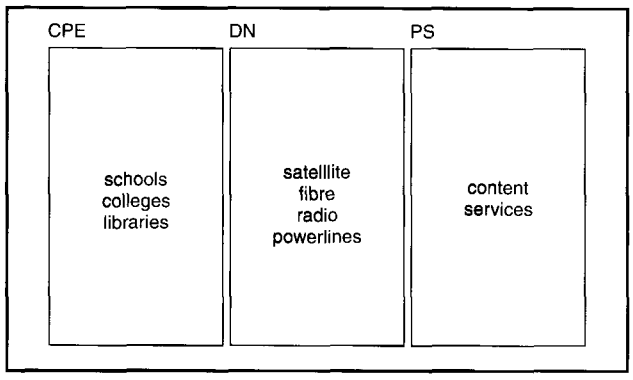


Figure 3: Concept of a Knowledge Utility

To cope with the different economics and availability of networks, together with different bandwidths, degrees of required interactivity and deliverables, a variety of network technologies are being deployed. The base level for school access in the UK is currently ISDN, but the Technology Colleges Trust, for example, is running a 2 Mb network to each school.

What is important however is not the infrastructure, but the range of educational power stations. What is envisaged for these is that a competitive market will be developed for educational content and services.

For educational content obvious examples would include on-line courseware, bulletin boards, on-line encyclopaedias, on-line dictionaries, virtual visits and navigation software.

Among the services, I would include financial accounting, academic record keeping, assessment and testing, curriculum support and statistical reporting. This area deserves more space than I can give in this article. My own belief is that the current WWW is at best an information web, or a learning resources web, but turning it into a learning web is a major engineering and intellectual exercise.

I foresee that some content will be free, some sponsored and some purely commercial. The best example that I can give from the utility sector is with telephone numbering. Consider the following 01256 (caller pays) 0800 (free to caller) 0891 (premium payment) 0345 (local call to national network). Already we are seeing the concept of educational portals being discussed in a number of projects.

It is this latter part that is currently least developed. One of the keys to understanding the economics of these new technologies for 'content' can best be summarized in an observation by David (Lord) Puttnam. He observed some years back that even if you made the best films in the world you were still dependent on American distribution networks. He noted that there was a vicious or virtuous circle, in that you could not afford the distribution infrastructure without adequate quantity and quality of content. At the same time you could not afford the content if you did not have a large distribution network.

Compared with traditional learning resources such as books, the up-front costs of CD-ROMS and Web-sites are very high. When it comes to major museums and galleries for instance, virtual access to major collections is very expensive. Some of the national museums in the UK for instance have over 50 million items. Consider the cost and navigational issues of a site with 50 million web pages! Add this to a population of 9 million school children in the UK wanting curricula access and the issues of design for scalability, extensibility and long-term viability are enormously critical.

In brief, I have outlined a policy framework at the national level that allows many projects to contribute "bottom up" to the creation of a national infrastructure. In the last section of this article, I will illustrate the UK government's approach to implementing the NGfL. For us in ICL the challenge is now to develop our own business within this evolving policy framework. We are still involved in various ways in developing the policy framework, but it is now public property and has a life of its own. Recently a number of other countries have shown interest in this policy framework and are looking at the approach taken in the UK.

Before this, I would like to give some personal views on the consequences for education of this infrastructure as it evolves.

Re-engineering Education

Just imagine that you wake up tomorrow and every school, every college and every library is connected to the internet. Is that it? Is that a national grid for learning?

I have written about this on a number of occasions, but here I can only outline a few of the key challenges to exploiting the infrastructure to raise educational attainment on a large scale. What is clear about the successful deployment of ICTs in any sector of society, is that what makes the difference between success and failure is 'change management'. In particular, teacher training is critical. In the UK there are 500,000 teachers, which makes the universal introduction of ICTs into schools probably the largest change management project since the creation of the NHS (and "BABY", *ibid.*) in 1948.

Looking at just-in-time in manufacturing, as well as at other on-demand services, shows a tendency to move away from individual performance to team performance. What I envisage is a re-engineered teaching force in the future with master teachers and para-professionals working in teams with ICTs taking the strain of administration and delivering much of the content, so that the teachers are their to do what is really important, support and motivation.

At the same time, I think that we shall also see a re-engineering of the curriculum. What just-in-time organisation supported by ICTs has enabled in manufacturing is mass customisation. I would envisage the current national curriculum evolving into a national framework for personal curricula to international standards. This personalisation of learning will be a reflection of the changing role of the teaching professionals.

One aspect of personalisation that I wish to comment on is educational special needs or disabilities. One of the areas that most excites me from our own trials and other research is that there is a growing body of evidence that ICTs suitably applied can overcome many learning barriers such as autism, dyslexia and physical handicaps. The potential to take the "dis" out of disability has profound social and educational potential. Turning it into reality is one of the great challenges.

The phrase in education is that ICTs will change the teacher from the "sage on the stage to the guide on the side".

While some of the details about how fully to implement and sustain a Knowledge Utility are still unclear, the creation of an overall goal or vision enables a large number of projects to work towards a common goal. A good policy framework can provide stability in the market and make for more business, but most importantly *better* business.

In the absence of a clear policy direction, it is difficult to invest with any certainty that there will be a real return or to manage risk sufficiently well to ensure a win-win situation with suppliers and clients.

In 1996, ICL approached the UK government and opposition with a costed proposal for a private finance vehicle to build a Knowledge Utility. This was important, because we used our experience to create a costed proposal, but also to show that ICL was serious about this sector as a business.

The incoming administration in the UK in May 1997 has developed and incorporated many of our ideas into its wider policy framework and we are now seeing other countries showing interest in the ideas and models developed in the UK. Many of my colleagues in ICL can feel proud of their contribution through projects and policies in thinking through this more pro-active role for us in public policy making.

I will now finish by considering, in overview, the UK government's development of the national grid for learning.

The UK National Grid for Learning

The many areas covered by projects contributing to the NGfL in the UK are summarized schematically in Figure 4.

The government is investing in IT for schools, along with teacher training and provision of content. A number of education action zones are being established to test more flexible methods of educational provision.

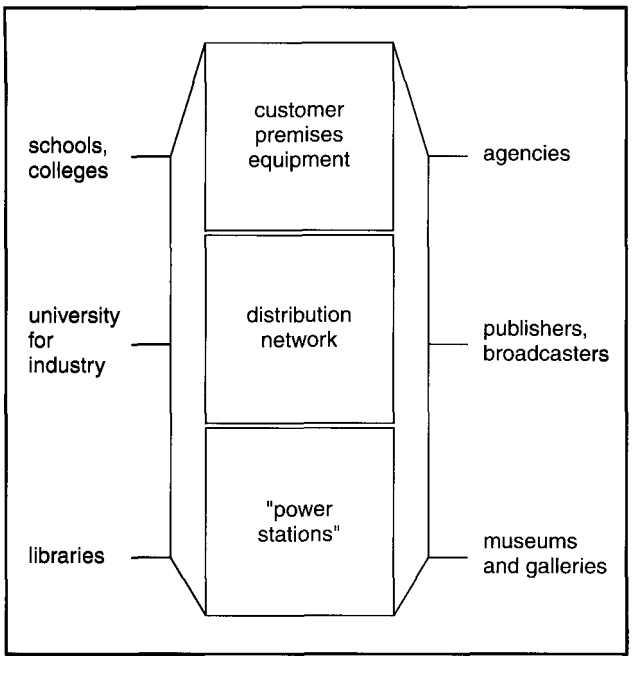


Figure 4: NGfL Project Areas in the UK

Additional funding is being provided for colleges. Alongside this, the government is to launch, in 2,000, a University for Industry, a learning brokerage available through call centres and on-line facilities to connect individuals to learning opportunities provided by colleges and the private sector. This in turn will be supported by individual learning accounts, potentially supported by smart cards, which will evolve into long term savings vehicles to fund lifelong learning.

A report from the Library and Information Commission, "New Library the Peoples Network", indicates how a national library network is being developed.

A fund has been established to allow museums and galleries to start making their collections available on-line for education.

A new digital learning channel is currently out for tender. Many traditional broadcasters and publishers are now experimenting with making their content available on-line.

At the same time, the government and its agencies are increasingly making their own material available on-line as part of the overall programme in government modernization.

A detailed reading of the various projects shows real efforts being made to create "joined-up thinking" and delivery across a complex and interacting set of organizations and sectors.

At the latest count, around £2 bn will be committed to the various projects indicated in Figure 4 over the current parliament.

The challenge for us in ICL is to build our business and expertise in this evolving market place, which is now growing rapidly.

Summary and Conclusions

Many colleagues in ICL have contributed to the ideas outlined above. What I have attempted to convey in this article is the process that we have been going through in one part of the company in trying to turn into reality our vision of the information society and of an IT services company as a thoughtful and pro-active partner.

We have made mistakes and learned much en route, but I believe that we have created some concepts and processes that are reusable in many areas of business, where scale, complexity and public policy interact.

To finish, on my own personal vision, in the 21st century, learning will be the largest industry in the world. ICL can be a major player in engineering a learning society. A Knowledge economy and a Learning society are the underpinning of the new economy.

I am very willing to provide further material to interested readers in this subject as I am aware that there are many areas which, of necessity, have been insufficiently developed in this paper. I may be contacted at Chris.Yapp@icl.com.

References

BANGEMANN, M., "Europe and the Global Information Society", DGIII (European Commission), 1994.

SCOTT, M.S., "The Corporation of the 1990s", Oxford University Press,1991.

YAPP, C., "Lifelong Learning; The Renaissance of Education", Masters of the Wired World" (edited by Anne Leer), FT Management, 1999.

DfEE, "Connecting the Learning Society", 1997.

Biography

Chris Yapp is an ICL Fellow specialising in Lifelong Learning and the Information Society.

He has been in the IT industry since 1980, joining from the Financial Times Business Information Service. He was Project Manager at Honeywell for the DHSS, National Unemployment Business System the world's first multi-vendor OSI operational system, before joining ICL in 1987.

At ICL, he was involved with the MIT " Management in the 1990s" programme which studied the organisational, management and strategic impacts of Information Technologies. He was also part of the team that developed ICL's approach to Systems Integration, Openframework.

Since 1994, he has been involved in developing ICL's approach to the emerging Information Society. In particular, he has been involved in the areas of ICT for education and training. He became ICL fellow for Lifelong Learning in 1997. This award was for his work on the National Grid for Learning and the ICL Cyberskills programme.

Chris Yapp has been involved in many policy groups in the UK and the EU including the Fryer committee, EURIM, British Library, DfEE, DTI and DGIII and DGXXII.

He is a frequent public speaker and contributor to publications on the social and economic impacts of ICTs. Most recently he contributed a chapter on the theme of the "Renaissance of Education" to the FT's "Masters of the Wired World".

Chris has an MA in Physics from Magdalen College, Oxford. He is a Fellow of the RSA, a member of the RIIA and the Strategic Planning Society.

Systems Management in ICL's Millennium Programme

Margaret Leigh

ICL High Performance Systems, Manchester, UK

Abstract

Systems Management tools abound in the marketplace. They range from highly focused, single function tools through to the enterprise management frameworks providing management capabilities which span the total management needs of the largest businesses. Within this overall range, management of the Data Centre places some highly specific requirements on the Systems Management tools.

Following on from the overview of High Performance Systems Division's (HPSD) Millennium programme [Procter, 1998], this paper describes some of the influences at work in the Systems Management sector of the industry and how High Performance Systems Division approaches the question of Systems Management in the Millennium programme.

Introduction

Within ICL, HPSD is focused on Data Centre computing and, specifically, on the hardware and software infrastructure that supports the central core of medium/large-scale business application services.

The Millennium Vision of Data Centre computing has been developed to reflect the trends in Data Centre computing and technology at the end of the 1990s and in the early years of the twenty first century. The vision, which is itself evolving, drives an ongoing product and capability programme known as the Millennium programme.

This article discusses the industry trends, market drivers and interfaces (both standard and proprietary) at work in the Systems Management arena. There are many Systems Management initiatives within the market aimed at improving the manageability of the desktop population and the networks which link these desktops. Standards, such as Microsoft's Universal Plug and Play and Sun's Jini, are specifically aimed at simplifying the process of adding desktop devices to a network, and allowing them to interoperate. Although very important in the overall management of the IT enterprise, this article does not deal with the topic of desktop network management. Instead, it concentrates on those aspects of Systems Management which are specifically relevant to management of the Data Centre. It describes how the industry trends apply to the emerging Data Centre environment based on industry standard volume components, with particular reference to ICL High Performance Systems Division's Millennium programme.

Industry trends

The trend towards re-centralization

During the late 1980s and the early 1990s, many enterprises moved towards a flatter organisation, with minimalist central departments and semi-autonomous operating divisions. Reflecting this trend, IT services were also devolved, and operating divisions were set loose to choose their own systems and applications. This trend led to the proliferation of departmental and distributed UNIX servers, which are now critical to businesses. More recently, this trend has accelerated with the addition of a large number of Windows NT servers. The cost of ownership and difficulty of maintaining and managing these systems is now seen as a major issue for businesses, which are looking for ways to reduce the overheads and increase the reliability and availability of these systems.

The above issues are today leading towards consolidation of IT services back into a Data Centre type of operation in which scarce skills can be concentrated and costs shared. However, the

operation is typically organised differently from the old centrally funded Data Centre. It is more like a Facility Management operation (and indeed may be contracted out to a FM supplier).

Facilities Management, outsourcing, mergers and acquisitions within large businesses are leading to larger, more complex, heterogeneous IT estates, the efficient management of which is an essential capability.

Market research carried out on behalf of ICL HPSD has shown that:

- most organisations have multiple servers, often from different suppliers, running a selection of operating systems
- most organisations have already moved, or are planning or considering moving, departmental systems back to the Data Centre
- most customers expect Windows NT (and Windows 2000) to play a much more significant role in operational systems in the future
- use of third-party systems management tools on UNIX and NT is not currently widespread
- all organisations have cost constraints and growing workload; i.e., they are under pressure to manage growing IT services with static or reducing resources.

Storage Area Networks

Another important trend is the change in the usage and capability of commodity peripheral products. Until recently, commodity peripherals such as disks and tapes were owned exclusively by a single server and could be used only by that server. Moves to make better use of under-utilised tape devices usually involved copying data from a client (such as a PC) to the server which would then write the data to tape.

Facilities for sharing peripherals such as robotic tape libraries have been available for many years in large Data Centres. Recent reductions in the price of such technologies mean that they are now economic for use in fairly modest sized Data Centres.

RAID disk subsystems, which provide very reliable data storage, are now shareable between a number of servers, leading to their economic use as a central data repository for a community of smaller servers.

Fibre Channel is an industry standard interconnect developed to overcome SCSI limitations in throughput, connectivity and distance. The simple loop topology of Fibre Channel-Arbitrated Loop (FC-AL) supports moderately large systems, for example, 50–100 disks can be connected through a single FC-AL, with an option for a second loop to provide full route resilience. Intelligent configurable hubs and switches allow the creation of virtual FC-ALs, providing a high degree of flexibility in the use of the Fibre Channel resource to satisfy throughput requirements of particular tasks such as back-up.

All the advances described above are leading to the adoption of a data-centric, rather than processor-centric, view of IT resources. This sharing of data storage and interconnection resources across a number of servers requires a commensurate degree of management control in order to exploit them effectively. Recognizing the importance of this management function and the lack of an agreed industry standard, each supplier of shareable resources is implementing a new management infrastructure and GUI for their product. This is leading to a proliferation of different management products, thus adding to the number of tools needed in order to manage the Data Centre.

The Role of Systems Management

Systems Management is provided to reduce the cost of ownership of IT systems whilst maintaining service availability. Its focus is the IT service provider, who is faced with spiralling costs and complexity caused by:

- diversity of equipment and vendors. This is a significant issue for Facilities Management companies and those taking Outsourcing business, where equipment may be inherited from a number of different customers
- the client/server paradigm. The move to multi-tier application architectures has introduced undoubted benefits in terms of applications development and ease of maintenance. However these client-server applications have given the IT service provider a more complex management task. It is often the case that each tier of a multi-tier business application runs on a completely different platform, for example, the GUI on a desktop

PC, the business logic on an intermediate server (often UNIX or NT) and the database on a mainframe or specialized server. For the business application to be available to the end-user, all these platforms, the comms links between them and all the application processes must be available

- greater geographic dispersion of systems due to the increasing popularity of multi-tier applications and globalisation of businesses
- demands for higher service availability.

One significant business aim is to reduce the management costs associated with a geographically distributed population of systems, where problem diagnosis, software upgrade, resource reconfiguration etc. may all require system administrators physically to go out to each system. In addition, the need has been recognized for each server's data to be reliably backed up and for suitable recovery processes to be designed and proven. These tasks can be more efficiently carried out by the IT department. There are opportunities for reducing the capital costs for back-up by centralizing the task and sharing resources.

To achieve maximum efficiency from the IT staff, the management interface needs to provide a common look-and-feel for each management task, regardless of the environments on which the business applications are running.

However, much of the marketing literature associated with Systems Management is based on the costs involved in managing the distributed client/server environment. It is presumed that most of the problem centres around management of clients, because of their sheer volume in comparison to the servers in the environment. This misses the point that problems with a server have a much greater effect on the total environment than problems with a few clients, and that the number of servers is growing swiftly, fuelled by the distributed server architecture promoted by Microsoft. More significantly, the management of servers in a Data Centre places different requirements on the Systems Management tools. Management of servers in a distributed environment can be carried out on a server by server basis. The complex inter-dependencies between resources in a Data Centre environment mean that the Systems Management tools used in this environment need to provide a common level of capability across the whole range of resources. The requirement is that a control action can be applied to a related group of resources in a co-ordinated fashion from a single command. The ACID properties normally associated with application transactions now need to be able to be applied to "Systems Management transactions".

System management products can provide much-needed assistance to the provider of Data Centre services in managing operational costs, skills requirements and the provision of required service levels. The need for automated assistance rises rapidly as the number of systems within a Data Centre increases.

The ubiquity of the World Wide Web, improvements in available bandwidth, and recent developments in Web-enabled management tools provide the building blocks for a solution which will permit remote management of systems, possibly from anywhere in the world subject to the security considerations.

Frameworks and Point Products

Enterprise management frameworks (CA Unicenter, HP OpenView, IBM Tivoli TME and others) take a holistic view, providing integrated management of applications, databases, servers, networks and desktops across an entire enterprise. However, a full-scale enterprise-wide implementation is a large investment in terms of money, skills and time. Furthermore, infrastructure projects are typically difficult to fund and organise. As a result, in spite of their benefits, the general take-up of frameworks remains low.

Aside from the enterprise framework products, there are a plethora of suppliers and products, each focused on particular parts of the enterprise network and/or on particular aspects of the system management process. However, a piecemeal, system-by-system approach to management can add as much to the problem as to the solution. A pragmatic approach is to adopt a small selection of these products for use on all systems throughout the Data Centre.

Management Interfaces

The ability to integrate a number of Systems Management products to provide a total management solution is primarily an issue of interfaces and interworking capability. Most manage-

ment products use a proprietary interface, which leads to the need for a separate integration exercise for every combination of products.

Support for industry standard interfaces, such as CMIS/CMIP in the OSI world and SNMP in the TCP/IP world, has gone some way to easing the integration problem. SNMP is a very simple protocol, initially devised for the management of comms networks. Its use has since been expanded and it is now a widely accepted standard, although its limited command set means that management products continue to rely on their proprietary interfaces to carry out the more sophisticated tasks.

The proliferation of different interfaces leads to increased costs for the vendors of Systems Management products, who have to port their products to all the leading operating system environments and integrate them with other leading management products in each environment, in order to achieve wide market coverage. For this reason, recent initiatives to introduce new industry-standard management interfaces with richer command sets have been widely supported.

Among these newer standards is the Desktop Management Interface (DMI), which is now widely used by business desktop hardware vendors to provide an interface for remote management of desktop systems. Several systems management products now accept information in DMI format.

Web Based Enterprise Management (WBEM) is another emerging interface, introduced with the intention of creating a new set of Network Systems Management (NSM) middleware services based on Internet technologies. Although adoption of the WBEM standard was initially slow, its acceptance among vendors of Systems Management tools has recently been growing rapidly, with a number of product announcements. If widely adopted, WBEM will improve the ability of management products to access/control a wide range of subject systems, and enable the integration of management processes employing different tools.

Wired for Management (WfM) is an Intel sponsored definition of a set of requirements for manageable servers. It defines categories of servers (Workgroup, Line of Business, Enterprise), the standards (i.e. DMI, SNMP, WBEM) which must apply for each category, and the minimum set of objects required to be manageable by these interfaces.

Millennium use of Management Interfaces

For historical reasons, the operating systems in a Millennium system each have their own approach to Systems Management.

Much of the management functionality of OpenVME is built into the operating system. As with all mainframe systems, the management interfaces in OpenVME are largely proprietary and not originally designed to fit into the wider enterprise environment which is prevalent today. The result is that there is a limited range of third-party management tools available to complement the native OpenVME management capabilities.

Management of UNIX is achieved through a mixture of in-built operating system capabilities and ISV supplied products. However, UNIX systems originated in a distributed environment, with the result that the management functions available tend to have a stronger bias towards management of a community of smaller systems. There are many management tools in the UNIX marketplace, although they are not all available on every flavour of UNIX.

For Windows NT, management functions are primarily supplied as add-on products either from Microsoft or from a vast array of third party software suppliers. Due to the sheer volume of the market for Windows NT, small start-up companies can be successful with products which solve a specific task (for example, licence management or virus protection) within one of the broad functional areas of management. However, many of these products, and to some extent Windows NT itself, do not address the issues of large systems. For example, most small departmental servers are administered and operated by the same individual, so the tools do not need to differentiate between the skills, privileges and processes involved in these different types of activity. However, in a Data Centre environment the day-to-day operation of the system is usually a separate role from that of the administrator. As such, the tools need to support the separation of the tasks associated with each role.

Given the different backgrounds and histories of the three operating systems in a Millennium system, it is hardly surprising that there are very few Systems Management products in the marketplace today which can provide a management function on OpenVME, UnixWare and Windows NT.

Provision of a common management infrastructure depends on a management interface which is fully supported by all three operating systems, and by the required range of Systems Management products. SNMP is the interface which offers the greatest degree of commonality. However, the interface does not lend itself to the full range of management functions which need to be addressed in a Millennium system. No other management interface is as widely accepted as SNMP for network management.

The emerging management interfaces (WfM, DMI, WBEM, etc.) provide the opportunity to put in place a more wide-ranging and functionally rich set of management capabilities based on open/de-facto standards, although there is still a problem due to the number of different standards being proposed. For this new generation of standards to provide a solution for Millennium, they need to be widely adopted by the hardware and software industry, so that Millennium systems can be built using components that support these standards.

Market considerations

There is still a good deal of movement amongst the framework providers, with existing leaders dropping back, and new players emerging. It is expected that the number of framework providers will reduce to two or three market leaders, with the scope and maturity of the products improving rapidly over the next few years. Much of this increase in the scope of the frameworks is being achieved by product acquisition.

There will continue to be a number of successful point product vendors, where the functionality of the point products exceeds that available within the frameworks. However, this area of the market is also subject to rapid change, with mergers, acquisitions and new start-ups all commonplace.

The trend for point product integration into the leading frameworks is likely to continue, allowing IT departments to add the superior functionality of chosen point products into their preferred framework.

Managing NT servers has become a key networked systems management (NSM) requirement. Many start-up companies are now delivering NT management capabilities, but these vendors do not have significant distribution channels. Meanwhile, established vendors are busy porting their existing solutions to NT.

Vendors of Systems Management products are subject to the same cost constraints as the providers of other software applications. This restricts them to porting their products to a small subset of the operating systems available in the market. The costs and timescales involved in porting applications to the OpenVME environment has historically caused a significant lag in the availability of market-leading Systems Management products on OpenVME systems.

Requirements for System Management in Millennium

Requirements for a System Management solution for Millennium include management of:

- A single OpenVME, UnixWare or NT instance
- Multiple homogeneous instances (OpenVME or UnixWare or NT)
- Multiple heterogeneous instances.

An objective is that Millennium will also be capable of being:

- managed as a stand-alone system
- integrated with point products managing other systems
- integrated into an Enterprise Framework infrastructure.

In all cases, there will be a set of peripheral resources, shared by some or all of the operating systems instances, to be integrated into the total management solution.

The suitability of existing Systems Management tools for the Data Centre environment tends to be taken for granted. However, this is not always

the case. For example, a Data Centre back-up solution ought to be able to cope with the very large data volumes associated with Data Centre systems. It should also be capable of meeting the performance requirements in order to complete the back-up (whether off-line or on-line) within an acceptable timescale. Backup products which originated in the distributed departmental client/server environment do not necessarily possess the necessary scaling and performance characteristics needed for the Millennium system.

The requirement is therefore to provide a System Management offering with Data Centre class capabilities, such as performance, robustness, rich functionality, and scalability. In addition, the objective is to select Systems Management products which are designed and implemented so that they do not introduce any single point of failure into an otherwise resilient system. A common presentation layer and user interface, which can run anywhere in the world, would provide maximum flexibility.

It is recognized that, where specific management tools are already in use in an enterprise, Millennium systems will need to be capable of fitting into the existing management infrastructure. This can be achieved via support of industry-standard management interfaces.

Architecture

Systems Management is a complex subject. The Millennium System Management architecture identifies thirty one separate management functions which need to be addressed to provide a fully managed environment. The functions are categorised as:

- Server Management—those management functions that are specific to particular platforms (e.g., related to the hardware or operating system personality in use)
- System Management—management of a relatively small number of IT systems, business critical and usually centrally located
- Enterprise Management—management of all aspects of the enterprise IT systems, including clients, servers, networks, etc.

The following section describes this architecture and explains the dependencies between the categories.

Server Management

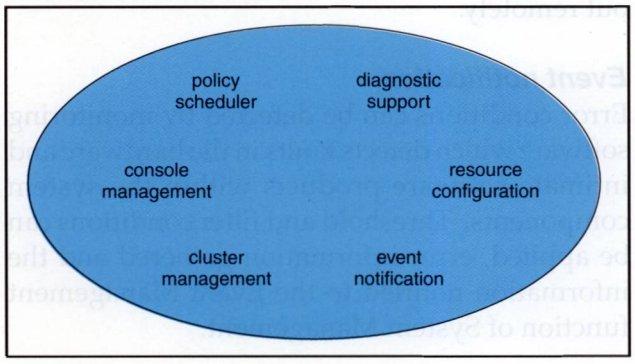


Figure 1: Server Management

In the Millennium context, Server Management is scoped to include the installation, configuration, initialisation, fault and exception monitoring and management of all hardware components, see Figure 1. The logging of server originated information into operating system logs, and the subsequent analysis and presentation of that information to System Management tools is also within the scope of the Server Management categorisation. Within this category, six specific functions have been identified.

Resource configuration

There are a number of different types of resource (e.g., processors, disks, tapes) that need to be configured within a system such as Millennium. Each component vendor supplies configuration utilities which apply to their particular devices; for example, a disk subsystem vendor may provide a utility to create RAID sets on a group of disks.

For flexible use of resources shared between heterogeneous systems, a further level of management allocates the resources, either logical or physical, to the required operating system instance. This allows a "virtual platform" to be defined, and subsequently changed.

Console management

It is normal at the moment for each server and each peripheral subsystem to have its own dedicated console, from which initialisation, configuration and error management actions are performed. This leads to a proliferation of consoles in a large server system. The console management function provides features which minimize the number of such attached consoles and allow

the associated management actions to be carried out remotely.

Event notification

Error conditions can be detected by monitoring software which detects faults in the hardware and intimate software products within the system components. Threshold and filter conditions can be applied, error information gathered and the information notified to the Event Management function of System Management.

Policy scheduler

One of the requirements of an enterprise server is consistent management of multiple concurrent applications with different business priorities. Many mainframe systems have had this capability for a considerable time and most mainframe systems run concurrent mission-critical applications. Today, Windows NT and UNIX servers predominantly run single applications. Where present, current UNIX schedulers attempt to allocate resources in a "fair" manner. ICL has developed a Policy Scheduler which does not allocate resources between all the processes in a system on a "fair" basis but allocates them according to business need. This has been achieved by applying the OpenVME scheduling model to UnixWare, allowing the business to define the relative priorities of the applications running on the server. HPSD are in the process of implementing the same scheduling model in a Windows NT environment.

Cluster management

A High Availability Cluster has two processor subsystems with shared disk storage. This provides redundancy for the whole processor subsystem. Failure of a processor subsystem is automatically detected, and critical workloads can be "failed over" to the other processor subsystem in the cluster, according to pre-defined rules.

Diagnostic support

Timely and accurate resolution of problems will sometimes require access to the failing system to gather diagnostic information. The Diagnostic Support function of Server Management provides an infrastructure to enable problem diagnosis to be carried out via a remote link by the support service provider.

System Management

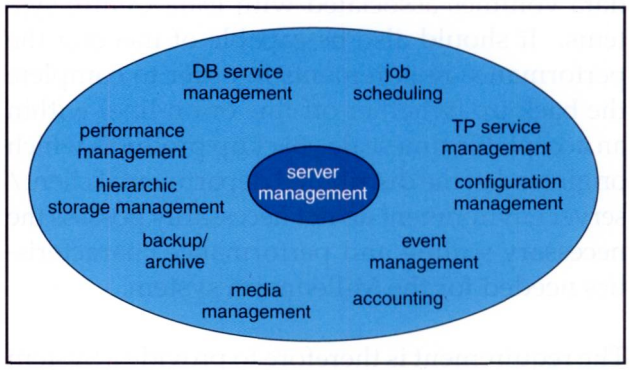


Figure 2: System Management

The System Management category, see Figure 2, is primarily focused on the efficient operational management of the subsystems within a Millennium system. It provides management tools which allow automation of the day-to-day management functions, reducing the amount of human intervention required and allowing the management actions to be carried out remotely.

The following are examples of the functions in this category.

Job Scheduling

The Job Scheduling application allows tasks to be started at predefined times and/or when defined criteria have been met (e.g., a previous task has been completed successfully). Thus a suite of tasks can be run automatically, with inter-dependencies being catered for. Alerts can be raised if any task encounters a problem. Tasks need to be able to be scheduled in this manner across heterogeneous environments.

Back-up, Archive, Media Management, Hierarchic Storage Management

These functions are used to ensure that data is secured in a controlled, manageable and recoverable manner. Multi-tier application architectures lead to different parts of a business application being able to reside on separate systems, and even on different operating systems. The target is to be able to manage a co-ordinated back-up of the data associated with this style of application across a heterogeneous set of systems or subsystems. Integration of the Back-up tool with the Job Scheduler allows back-up operations to be managed as part of the overall suite of tasks.

Event Management

Event Management provides monitoring of events and alerts from the components of a heterogeneous environment. Error conditions are detected by the Server Management function and passed up to the Event Management tool. Events and alerts from a number of components can be correlated, and threshold and filter conditions can be applied. Rules can be defined which allow actions to be invoked—either to carry out automated actions or to notify a person by email, pager etc.. Where the event cannot be handled locally, Event Management can be integrated with the Problem Management and/or Help Desk functions of Enterprise Management to allow the event to be escalated.

Performance Monitoring

Performance Monitoring provides the ability to gather statistics across a heterogeneous environment, analyse the statistics and present the information, usually in a graphical form. Tailoring is available at all these stages, allowing the business to define the frequency and amount of statistical information gathered, what analysis and correlation actions should be applied to this data and how it should be presented. Monitoring can be carried out on-line or using historical data Where an issue is identified, the performance monitoring tool provides the ability to drill down to the required level of detail, to assist in problem diagnosis.

Enterprise Management

This category of management tools, see Figure 3, is targeted at managing all aspects of the IT infrastructure which support the business; i.e., processes and procedures as well as hardware and software components, networks etc..

Millennium systems supply information from the Server and System Management layers of the model, which can be used by the Enterprise Management products in order to build up a picture of the status of the whole IT infrastructure.

For example, events notified by the Server Management layer can be passed up to the Event Manager in the System Management layer. If the event cannot be satisfactorily resolved locally at this layer, it may be further escalated to the Problem Management and/or Help Desk products in the Enterprise Management layer. The ability to correlate information from a number of different systems improves the diagnosability of the underlying problem, resulting in faster, more accurate corrective actions.

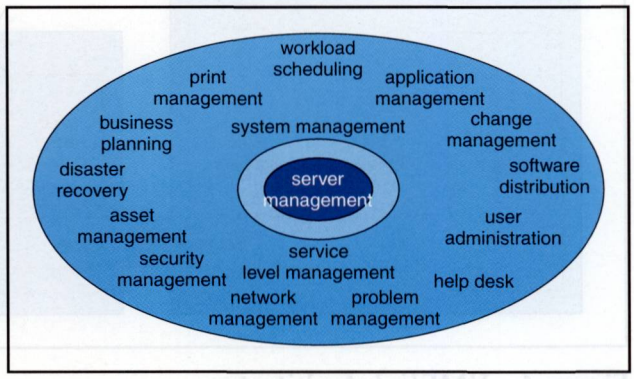


Figure 3: Enterprise Management

Strategy

The strategy for System Management of Millennium systems is to select market-leading Systems Management products and to integrate them together with the other components of the Millennium system, to provide Data Centre management capabilities in chosen functional areas.

HPSD has developed a common infrastructure which allows OpenVME to be integrated into the market leading Systems Management tools and frameworks with minimal porting or development effort. The VMELink architecture is shown in Figure 4.

The VMELink architecture allows the integration of OpenVME into the wider management world, by providing a means of accessing existing OpenVME management commands and data from the market-leading Systems Management tools. Most Systems Management tools are designed on a client-server model, so, to be able to be managed using these tools, OpenVME must appear to act as a client. Traditionally, this would have required ICL or the software vendor to port the client code into the OpenVME environment and required OpenVME to support the communications protocols used by the tool. In the past, the resource intensive nature of this exercise limited the number of tools which could be made available on OpenVME. Instead of needing to port all the client functionality of a particular product to OpenVME, the VMELink architecture

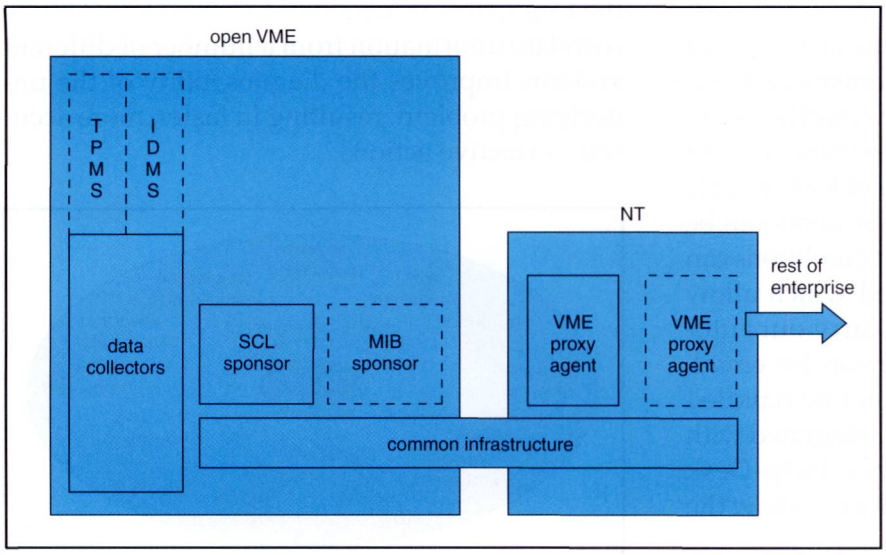


Figure 4: VME*link* Architecture

provides an infrastructure which allows a Proxy Agent, usually running on Windows NT, and an SCL Sponsor running on OpenVME to act as a 'gateway' between the Systems Management product and the standard OpenVME interfaces.

A Systems Management tool, running, for example, on a Windows NT platform, can request data using its normal interface. An OpenVME Proxy Agent for that tool converts this request into SCL calls and passes them across the VMELink Common Infrastructure. Where a number of SCL commands are required in order to provide the information requested by the Systems Management product, this will probably be run as a script in OpenVME. The Common Infrastructure provides a 'pipe' between OpenVME and the Windows NT system running the Proxy Agent. It allows SCL calls and their results to be passed between the Proxy Agent and the SCL Sponsor. The SCL sponsor on the OpenVME platform executes the SCL commands or scripts in order to collect the required information and pass it back across the Common Infrastructure to the Proxy Agent. The Proxy Agent can then manipulate the data as required and pass it back to the Systems Management tool in the required format.

Given the plethora of management functions to be addressed (as described in section 4.1), and the wide range of products available in the marketplace, it is important that the strategy for System Management of Millennium systems includes some method of prioritizing what is to be done first. Much market research has been carried out in the Systems Management arena, both privately on behalf of ICL and by the leading industry analysts such as Gartner and IDC. This research tends to converge on a small number of functional areas as being critical to the management of Data Centre systems. These areas have therefore been chosen as the focus of the initial product selection and integration activities within the Millennium programme. These functional areas cover job scheduling, event automation, managing performance, storage management and problem management.

The objective is that for each management function, all the subsystems of a Millennium system will be manageable via a common user interface. Thus, for example, events from the Windows NT, UnixWare and OpenVME operating systems, the processor subsystem, storage area network, etc. can all be managed via the GUI of the chosen Event Management product. This type of integration is primarily aimed at the day-to-day operation of the system. It provides a high-level view of the health of the overall system, and allows management actions to be carried out using Operating System independent interfaces. It is important to note that this type of management product is not a replacement for the full set of native management tools and interfaces provided by each of the supported Operating Systems (Windows NT, UnixWare, OpenVME) and other components of the system. These will continue to be supported. If the system administrator or diagnostician requires more detailed information or control than can be achieved using the standard tools, then the native tools and commands for each operating system will continue to provide the appropriate access.

As described in the early sections of this article, Systems Management is a highly volatile and rapidly evolving sector of the IT industry, in terms of products, interfaces and companies. It is the aim of HPSD's Millennium System Management policy to strike a balance between stability and leading-edge technology, between the latest functionality and the Data Centre expectations of reliability, scalability and performance.

Roadmap and status

Significant steps towards the realization of the Millennium vision for System Management have already been achieved within the Trimetra product range.

The Trimetra Y system [Martin & Stewart 1998] provides a single consistent framework within which all the component parts of the system can be managed. This comprises a common administration and operation system together with a common set of facilities for the support of the components of the system, covering problem notification, diagnosis and correction.

The Trimetra Xtraserver Pi [Messham, 1998] includes selected market-leading Systems Management products which have been integrated and tested with the system. The functional areas for which products are supplied are hardware management, back-up, performance monitoring, application management, print management, batch management, workload scheduling and support service.

Over the next few years, the System Management capabilities of the existing product ranges are planned to converge. The Millennium programme will use market-leading Systems Management products and industry standard interfaces to build a common management infrastructure. This infrastructure should enable a Millennium system, consisting of multiple heterogeneous servers and their shared resources, to be managed as a free-standing system or to be integrated into the management of the enterprise.

Conclusions

After years of decentralization of IT infrastructure, systems are beginning to be consolidated into Data Centres. Systems Management of the resultant heterogeneous set of systems is seen as a fundamental requirement, since it provides the opportunity to reduce the cost of ownership whilst maintaining service availability. There is a wide range of management functions which need to be addressed to provide a fully managed environment, ranging from management of individual hardware components such as processor subsystems through to overall enterprise management, targeted at managing all aspects of the total IT infrastructure, including the business processes and procedures. The Systems Management marketplace offers a rich selection of products ranging from those tightly focused on solving a specific management issue to the Enterprise Management Frameworks which provide integrated management of applications, databases, servers, networks and desktops across an entire enterprise.

HPSD's Millennium vision and programme includes Systems Management as one of its key capabilities, concentrating on those management functions which have been identified as critical to the management of Data Centre systems, namely job scheduling, event automation, managing performance, storage management and problem management. In each of these functional areas, the Millennium Systems Management strategy is to provide an integrated management solution which takes advantage of the market-leading Systems Management products, together with HPSD's VMELink architecture to enable OpenVME integration into the overall management infrastructure.

Glossary

ACID	Atomicity, consistency, isolation, durability
CMIP	Common Management Information Protocol
CMIS	Common Management Information Standard
DB	Database
DMI	Desktop Management Interface
FC-AL	Fibre Channel Arbitrated Loop
GUI	Graphical User Interface
NSM	Networked Systems Management
OpenVME	ICL's operating system
OSI	Open Systems Interconnection
Platform	A collection of hardware and software components with the ability to process and store information. The hardware includes processors, store,

	peripherals and network controllers; the software includes operating systems
SNMP	Simple Network Management Protocol
TP	Transaction Processing
WBEM	Web Based Enterprise Management
WfM	Wired for Management

Acknowledgements

I would like to thank all the members of HPSD working on System Management. I offer my especial thanks to the late Brian Procter, Chris Williamson, Ivor Lewis, Andrew Brightwell and Barrie Archer, who helped me in the formulation of the ideas expressed in this paper.

Bibliography

PROCTER, B.P., "The Enterprise Datacentre—ICL's "Millennium" Programme", ICL Systems Journal, Vol 13, Issue 1, 1998.

MARTIN, C., STEWART, C.P., "Trimetra UNS," ICL Systems Journal, Vol 13, Issue 1, 1998.

MESSHAM, D.K., "Trimetra Xtraserver," ICL Systems Journal, Vol 13, Issue 1, 1998.

Biography

Margaret Leigh joined ICL in 1986 with fifteen years experience in designing and implementing applications software. She joined a small team of validation engineers in Mainframe Systems Division (later to become High Performance Systems Division) validating the first multi-node Series39 systems. Over the next 10 years, she was involved in the validation of most of the hardware and software products released by HPSD, becoming the validation strategist for the Product Validation & Release group. In this role, she was responsible for designing and costing major validation projects, including OpenTP, Goldrush and Xtraserver. She is now a member of the HPSD Systems Architecture team, responsible for Systems Management in the Millennium programme.

"Baby's" Legacy—The Early Manchester Mainframes

Chris Burton

Abstract

There has been a great deal of interest in the celebrations surrounding the 50th anniversary of the first running of an electronically-stored computer program in Manchester in June 1948. This article summarizes the history of the first Manchester computing machine and its successors, some of which became Ferranti and then ICT and ICL products. One theme will be the recurring concerns with the important role of storage mechanisms. It is noted that there is a continuity in design of computers from that first computer to current ICL high-performance products.

Introduction

The history of the computing industry is littered with important breakthrough events, some more or less precisely and others vaguely defined. As Professor Michael Williams, University of Calgary, has pointed out, if you add enough defining adjectives to an artefact it is easy to make it a world first [Williams, 1998]. What constitutes a world first worth talking about is also a matter of opinion and context; for example, is it more important to articulate an idea, or to realize that idea? So debates about whether the first computer was Zuse's Z1, or Colossus, or ENIAC or the Manchester "Baby", and was the "father of computing" Charles Babbage, Alan Turing or John von Neumann will entertainingly continue indefinitely. However, all computer historians agree that an extremely important event took place on 21st June 1948, and some would say that that date is the true beginning of the age of computing which we have experienced during this last half-century.

On that day for the first time, a program stored in the internal electronic store of a general purpose electronic digital computing machine ran and produced the correct answer. Everyone who has hands-on use of a modern computer would recognize the event as something they do every day, and no earlier machine was like that. The computer was the Small-Scale Experimental Machine (SSEM) built at the University of Manchester by Williams, Kilburn and Tootill, and is popularly known as the Manchester "Baby" computer.

That development did not arise spontaneously but as a result of mushrooming interest and activity in electronic computation, mostly associated with the Second World War. There are many threads and connections going back to the 1930s and earlier, but by 1945 there were a few dozen people in the world who understood the possibilities of digital computers, and who were beginning to communicate their ideas. It is striking how poor this communication was—groups and individuals operated largely unaware of what others were doing or had done. It is easy to forget that in 1945, it was necessary to book a transatlantic telephone call several hours before the call itself, even assuming the caller could afford such a luxury. Learned papers were published at a rather leisurely pace, and fax, email, email lists, and Usenet news were unheard of!

This paper will attempt to set a historical context which led to the early Manchester computer development. The SSEM itself will be described, and then some of the subsequent machines which were designed at the University of Manchester, some of which became Ferranti and ICT, and contributed to ICL, products. The author claims no originality for the material presented here - indeed readers whose appetite is whetted are urged to follow some of the excellent histories which cover the subject matter with greater detail and authority [Lavington, 1980, 1998], [Randell, 1973], [IEEE Computer Society, 1953]. The emphasis in those early computing machines was on hardware design and description—software barely got a look-in. This is in complete contrast to the situation now at the turn of the century, when the computer industry is dominated by software considerations, with the supporting hardware platforms being taken very much for granted.

The importance of data storage as a concept and as a technology should be apparent throughout this story. Babbage used the word "Store" to denote the place where numbers were held ready for manipulation, but the anthropomorphism "memory" was adopted in the USA from an early stage[1]. The author campaigned unsuccessfully in ICL for many years in the 1960s to reject that word in favour of "store", but industry pressure was overwhelming, and it is noted that this Journal adopts the norm!

The Historical Context in the 1930s and 1940s

We can too-readily focus exclusively on the Manchester work assuming a big-bang development, but a proper understanding can only come from seeing how one thing led to another such that Manchester got there first. This fairly lengthy section is necessary to set the scene for the creation of the SSEM.

For the time being, we can disregard the extraordinary work on the Analytical Engine by Charles Babbage from the 1830s onwards, because that work was unknown to most engineers who were pioneering the early computers. By the 1930s, business data-processing was well-established (though not by that name) in the form of punched-card installations with means for sorting data, repetitive simple calculations and printing tabulations. For very specific applications, complicated desktop machines, e.g., the Comptometer, were used, being elaborations of mechanical adding machines. But requirements for scientific calculations were ill-served. The principal aids to scientific calculations were logarithm tables, the slide-rule and the mechanical adding machine. However, two noteworthy attempts to mechanize scientific calculation during the 1930s should be mentioned.

The first was the remarkable initiative of Konrad Zuse in Germany to create a machine for repetitive scientific calculations, started when he was a mature student in 1934 [Randell, 1973]. This work appears to have been done without knowledge of the efforts of other investigators, yet by 1938 he had built a mechanical calculator externally-programmed by a punched program tape. This had a mechanical store for numbers, using floating-point binary representation, but no conditional branching. It was very unreliable, and Zuse embarked on subsequent machines using electromechanical relays and later some electronic elements. His work was barely known outside Germany, and does not appear to have had any influence on other researchers.

The second notable development was of a machine to perform calculations in physics, typically solutions of sets of equations. This machine, the Atanasoff-Berry Computer, was designed by John Atanasoff in 1938, and had been built by him and Clifford Berry at Iowa State University by 1942 [Randell, 1973]. It used binary arithmetic using valve technology logic gates, and had an electromechanical store for about sixty 50-bit numbers. There was no program that we would recognize as such. During World War II, the US government does not seem to have taken advantage of this remarkably early electronic calculator, probably because bigger systems were already on the horizon. A working replica of the ABC was completed at Iowa State University in 1997.

For the purpose of this history, we can set aside further electromechanical computer developments such as the Harvard/IBM calculators and the Bell Laboratories relay calculators, and focus on electronic devices. The 1,800-valve Colossus code-breaking machine is often called a programmable computer, but it was developed for a highly special purpose, and it would be a stretch of terminology to regard it as general-purpose. But it did show that large assemblies of electronics were sufficiently reliable to do useful work, and gave Alan Turing and Max Newman and others an insight into the possibilities for general-purpose computers after the war, although no mention of Colossus itself and its work could be disclosed until the 1970s.

Highly visible from 1946 onwards was the mighty ENIAC computer built at the Moore School of Electrical Engineering at the University of Pennsylvania, and operational in 1945. This, like Colossus, was aimed at a specific task, the calculation of ballistics tables. However, it could be re-

[1] Curiously, the EDSAC team at Cambridge who later did much to reveal Babbage's work, used the term "memory" in 1948, presumably because of Wilkes' contact with the Moore School. Perhaps an indication of the independence of the Manchester team is their use of the term "store".

programmed for different tasks by reconnecting the various calculating units in different configurations. It again demonstrated that adequate reliability could be achieved with the available electronic technology, in this case using 18,000 thermionic valves. The Moore School team attracted John von Neumann to contribute to ideas for a general-purpose computer, and a series of lectures on the principles of computer design held there in 1946 influenced several teams to start their own construction projects.

One of the attendees was Maurice Wilkes from the University of Cambridge, whose team started work shortly afterwards on the EDSAC, a machine based on ideas similar to those of the Moore School. Cambridge, however, produced a working machine before the Moore School. With his exposure to Colossus, Alan Turing started a computer design at the National Physical Laboratory, intended to lead to the Automatic Computing Engine, ACE. Both these teams started from a mathematical computing tradition, but the Manchester work described next was founded in electrical engineering.

Original SSEM

During World War II, F.C. Williams led a group at the Telecommunications Research Establishment (TRE) at Malvern responsible for electronic circuit designs for radar. The group included Tom Kilburn and G.C. Tootill. In 1945 and again in early 1946, Williams was invited to visit the MIT Radiation Laboratory to contribute to writing a series of textbooks on all aspects of radar. While he was in the United States he saw experiments aimed at storing a radar image on the screen of a cathode ray tube (CRT) by detecting the electrostatic charges inside the screen caused by the CRT beam. This would have been research familiar to him because of the work by his own group on precision timing circuits associated with radar. On the second visit, he also called at the ENIAC project at the University of Pennsylvania, saw the computer and heard about their plans for further development of a general-purpose machine, EDVAC. He must have learned of the search for an electronic-speed, large-capacity storage system at this time, and he thought that the electrostatic charge experiments on CRTs could be used as a computer store. He returned to TRE and set up experiments with his colleagues such that by the end of 1946 they could store a single digit, and Williams filed a patent for a CRT store for a computer in December 1946.

At this time, Williams was appointed Professor of Electrotechnics at the University of Manchester. He brought his colleague Tom Kilburn on secondment from TRE to continue the CRT research, together with the equipment they had been using. During 1947 they filed several more patents on different aspects of the subject and by the end of that year they could store 2048 digits on a CRT screen for several hours. Throughout this time, they were well aware of the efforts of several other teams towards building a large automatic computing machine, and the central requirement for an adequate storage system for such a machine. Although they were in occasional contact with Alan Turing at the National Physical Laboratory and with the Professor of Mathematics, Max Newman, in Manchester, it is clear that Williams and Kilburn had their own independent views on how to design a computing machine.

The Williams-Kilburn CRT store seemed to fulfil the requirements, and in order to verify its usefulness "in the hurly-burly of computing", Kilburn decided to build a minimal computer around it. They were joined by their TRE colleague Geoff Tootill, and essentially Kilburn and Tootill built the Small-Scale Experimental Machine during 1948. It successfully executed a small program on 21st June 1948, the first time in the world that a general-purpose stored program computer operated [Williams and Kilburn, 1948].

The objective of the Manchester team at that time was to improve the technology for computing machines generally, and the store particularly. The new field of program development held no interest for them. As a result, the SSEM was small in capacity and in functionality (so was aptly called "Baby"), but it had all the features to make it general-purpose and universal in application. It was therefore quite unlike all earlier machines such as Colossus and ENIAC which imposed some or other constraints on their flexibility.

The simple structure of the machine can be seen in Figure 1. It has a classic von Neumann architecture, holding both instructions and data in the same store.

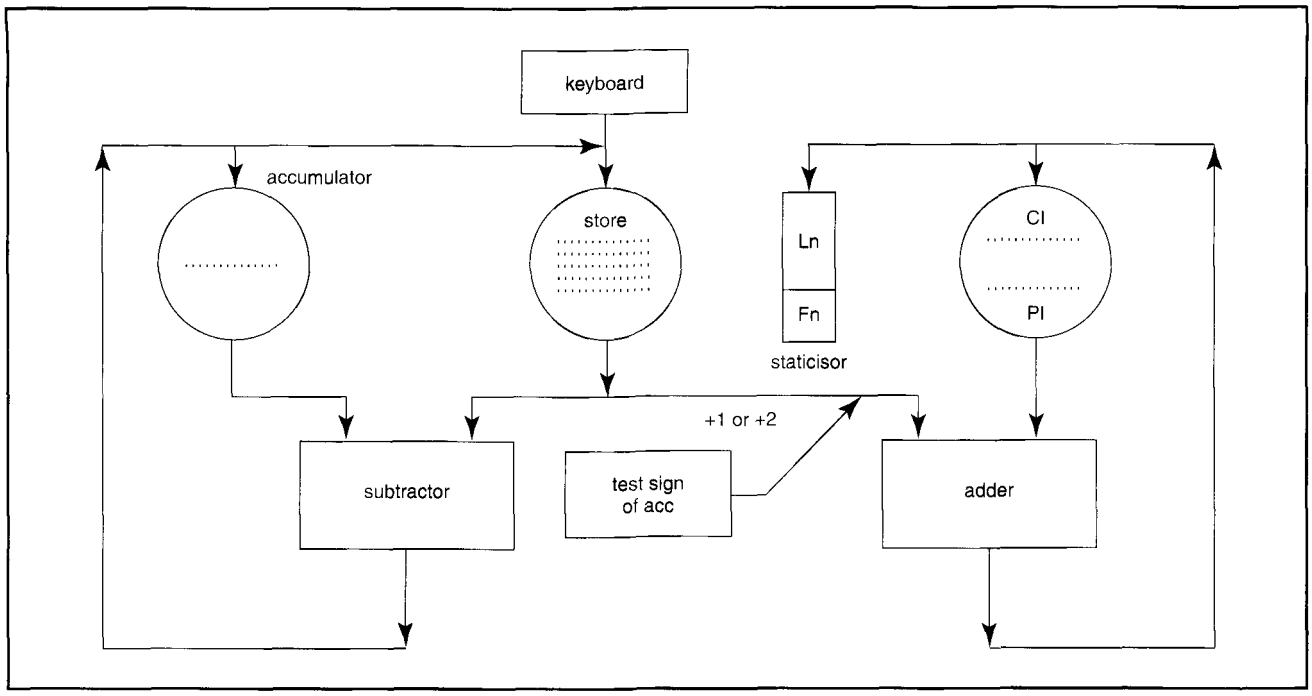


Figure 1: Simplified schematic of SSEM

The main characteristics of the SSEM were:

- 32-bit word length
- one instruction per word, single address order code
- 7 instructions in order code, but only one arithmetic operation—subtract
- 32 words in the store, but up to 8192 words addressable
- serial binary arithmetic using 2's complement integers in an implied accumulator
- speed approximately 700 instructions per second
- input of data—direct entry of 0s and 1s into the store by push-buttons
- output of data—view the binary content of the store on a monitoring CRT as a pattern of dots and dashes.

Having built the small machine and tried several programs on it, the team doubled in size by taking on two research students and bringing in Alec Robinson who had been building a binary digital multiplier in the department, using the CRT storage technology. Max Newman took a close interest in the machine at this time, and during the summer the idea of a special register for modifying addresses in instructions, what we now call an index register, was invented and incorporated. It was also realised that even with more CRT stores, the target storage capacity (10,000 words) was unlikely to be attainable, and that cheaper, if slower storage would have to be added. Consequently, at an early stage, a magnetic drum was attached to the experimental computer, the original being the piston of a Wilson cloud chamber which Williams had had nickel plated! By 1949 the enlarged machine was being called the prototype Mark 1. A photograph of part of the machine at this time is shown in Figure 2.

So the foundations were being laid for some characteristics of Manchester designs—innovative inventions like the index register which used an electronics solution to simplify a programming problem, and architectural strategies such as the two-level store aimed at finding a good performance versus cost in the storage system. It is interesting to contrast the approach of the Cambridge EDSAC team at about the same time, 1947 to 1949. The EDSAC was aimed squarely at getting a computing service going early, with a simple machine design, and in making the programming task less arcane. The Manchester team was aiming to help the programmer with innovations, but it took time for these to be easily accessible.

But these technical considerations were eclipsed by a more important event which took place in October 1948. The Ministry of Supply contracted with Ferranti to build an engineered version of the computer, "...to Professor Williams's instructions".

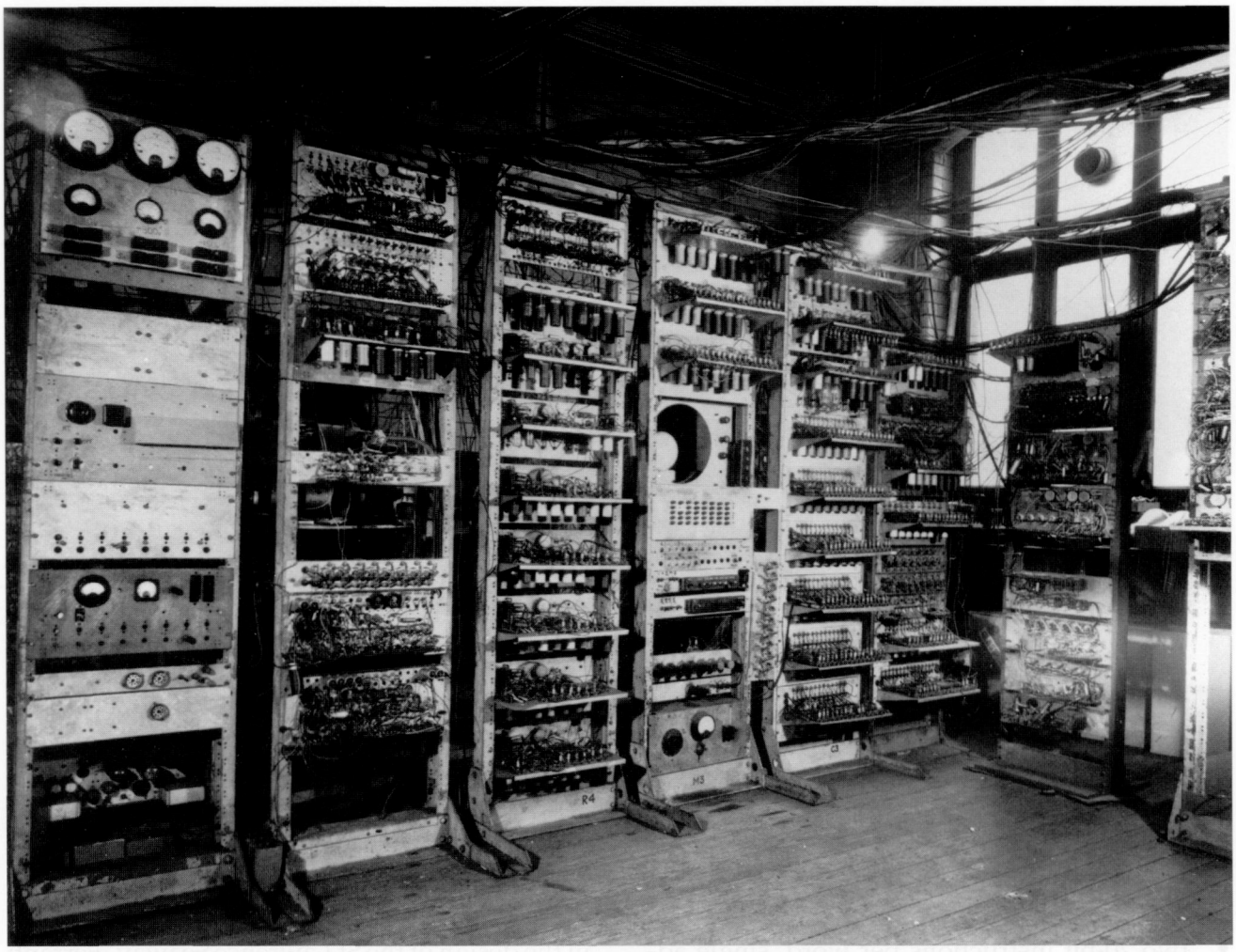


Figure 2: Part of the Prototype Mark 1 at the University of Manchester in early 1949

So began a long and valuable association between the University and Ferranti.

The Ferranti Mark 1 Computer

The privately owned company, Ferranti, had acquired a major expertise in radar development, and in electronics generally, during World War II [Tweedale, 1993]. Already in 1948 it had sent Dr D.G. Prinz to the United States to learn about electronic computers and whether it was a field in which the firm could exercise its talents. Being located in the Oldham area it was well-placed to collaborate with the University. Several engineers from the Instrument and Radio sections formed the initial Computer group. One of the first tasks was to deliver eight CRT storage units to be added to the University computer. The University set out to enhance and enlarge the experimental machine to become the University of Manchester Mark 1, while in parallel Ferranti started the development of a similar machine under the guidance of the University. Both Alec Robinson and Geoff Tootill were employed by Ferranti to contribute to the design.

The University Mark 1 provided a computing service up to the autumn 1950, when it was dismantled and scrapped. The first Ferranti Mark 1 was delivered to the University in February 1951, where it resumed the provision of a computing service.

The Ferranti Mark 1 was very well engineered and a good description of its sound engineering principles exists [Pollard et al, 1953]. The electronic technology was very similar to that of the University Mark 1, using the same valve types. It used Williams-Kilburn storage, employing a special quality CRT developed for the purpose by GEC.

The main characteristics of the machine were:

- 40-bit word length

Figure 3: The first Ferranti Mark 1 computer as delivered to the University of Manchester

- two 20-bit instructions per word, single address order code
- 26 instructions in order code, including multiplication
- 8 index registers
- 256 words in the store (eight CRTs)
- 3,840 word drum store
- serial binary arithmetic using 2's complement integers, double-length accumulator
- speed approximately 930 instructions per second; multiply 2.16 mS
- input of data—5-bit paper tape reader
- output of data—5-bit paper tape punch or teleprinter

The Manchester Autocode programming system was developed on this machine by R.A. Brooker. Although it was a serial machine, it had a competitive performance because of the availability of the index registers, the immediate access of the CRT store, and the relatively fast multiply instruction. A photograph of the machine is shown in Figure 3.

A second Mark 1 computer was sold to the University of Toronto where it is credited with helping in the design of the St Lawrence Seaway. Ferranti then redesigned and enhanced the machine to become the Mark 1*, pronounced "Mark 1 star". This had 384 words of CRT storage, a 16,384 word drum store and an enlarged instruction set. By 1957 seven machines had been delivered.

MERCURY

Once the first Ferranti Mark 1 computer had been delivered to the University, in 1951, Tom Kilburn's team started to design a successor, Mark 2, the Megacycle Machine, or "Meg". The aim was to build a machine an order of magnitude faster than the Mark 1 with enhanced reliability. The basic architecture was similar to its predecessor, but with parallel access to ten bits at a time from the store, and with a very fast serial arithmetic unit. Furthermore, hardware floating point arithmetic was provided. The development took place in collaboration with Ferranti and with the intention that the new machine would be put into production. The prototype machine used CRT storage, which had now been refined by the University team, but it was intended that the production machines should use ferrite core stores

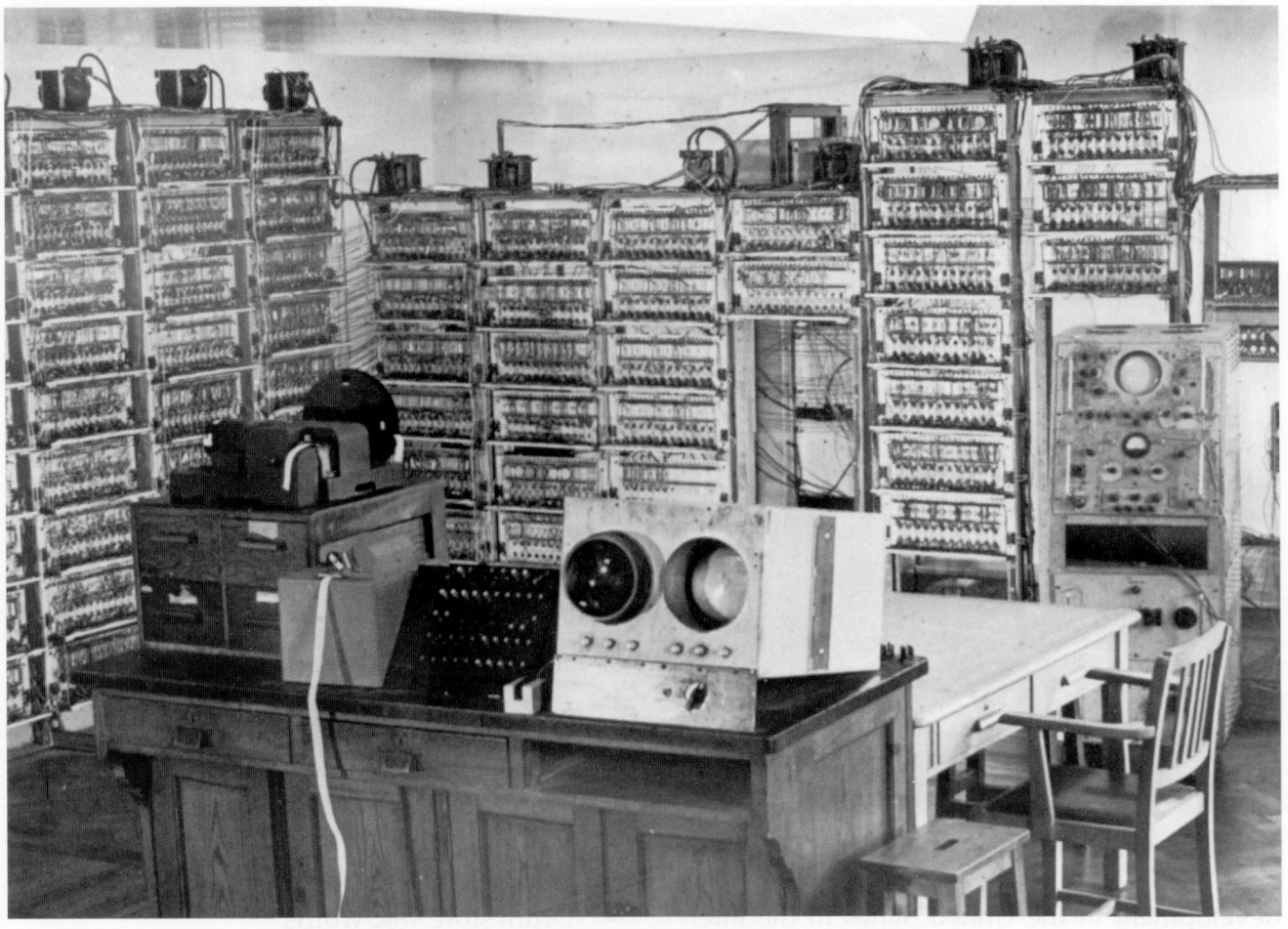


Figure 4: The Mark 2, or "Meg", to be put into production as Mercury

which were then just becoming available. It was conceded that core stores were less troublesome to set up and were more reliable than the CRT store. The prototype first ran in 1954, and the first production Mercury (as it was named by Ferranti) was delivered to Norway in 1957. Figure 4 shows "Meg", the prototype Mercury.

The specification of Mercury can be summarized as follows:

- 10-, 20- and 40-bit word length
- 20-bit instructions
- ferrite storage 4 blocks of 1024 words of 10 bits and 10 microsecond cycle time
- four magnetic drums each containing 4,096 40-bit words
- parity checking for internal and external storage
- eight 10-bit index registers
- floating point addition 180 microseconds and multiplication 360 microseconds.
- paper tape input and output, though punched cards were an option.

Nineteen Mercurys were delivered, at the time amongst the most powerful computers available in the UK. Architecturally Mercury was not greatly different from the Mark 1, and could be regarded as a fast, reliable consolidation of that earlier design.

ATLAS

In 1956, during the long delay from the Mark 2 working in 1954 to the delivery of the first Mercury, Tom Kilburn and his team embarked on the ambitious development of Muse, the microsecond computer. This transistor machine, started only eight years after the success of the Baby, was intended to be very fast (one million instructions per second, cf. Baby at 700 instructions per second). A large number and variety of peripheral devices were to be connectable to make it suitable for business use, and in particular, a very large internal storage capacity was considered essential.

Although Baby had run the world's first stored program, the massive investment in computer

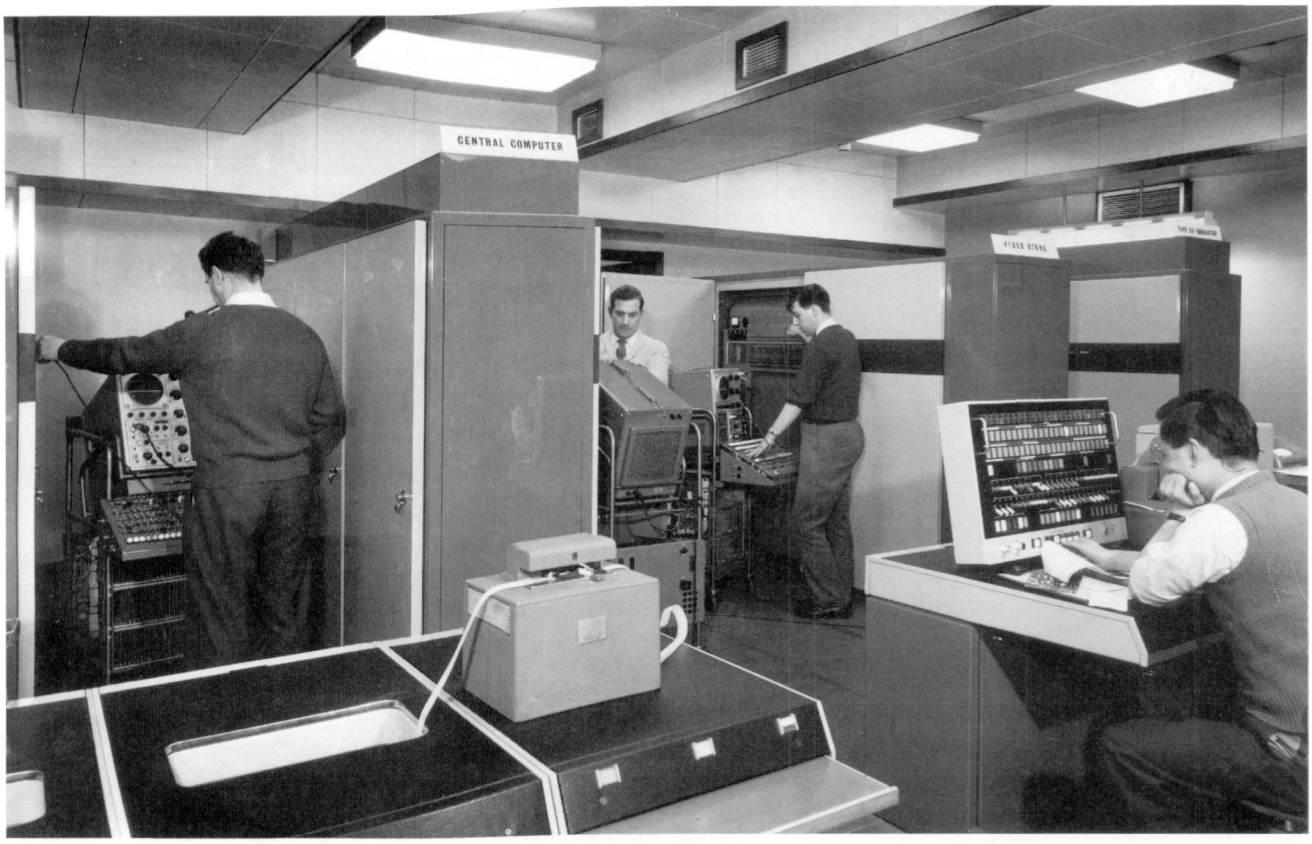


Figure 5: Part of Atlas being commissioned by University and Ferranti engineers

development in the United States in the intervening years meant that British developments were outstripped in speed, and far outstripped in number of computers and hence user experience. The specification for Muse was intended to redress the balance in terms of performance.

As was the case with the original Mark 1, many innovations were developed in Muse. Concepts familiar in present-day machines were pioneered, such as virtual storage, pipelining, an interrupt system and interleaved paged store. There was an ambitious permanent operating system, the Supervisor, which provided job scheduling, spooling and multiprogramming. Some of these developments took place when the Ferranti Computer Department, with government funding, collaborated with the University from 1959 onwards, by seconding teams of engineers and programmers to work on the project. At this time the machine became known as Atlas, and it provided a user service from December 1962. It is thought to have been the fastest computer in the world at the time. Yet its specification, modest by modern standards, was:

- Main store 16,384 words of 48 bits
- Drum store 96K-words
- Read-only store (e.g. for interrupt routines) 8 K-words, 300 nS access time
- 127 integer registers, each 24 bits
- Integer instruction times typically 1.5 microseconds, floating point 2.7 microseconds.

Part of the first Atlas installation is shown above in Figure 5.

A second Atlas was sold to London University and a third for joint academic use to The Atlas Computer Laboratory at Harwell. This machine had 48K of main store and a vast array of peripheral devices. A simplified variant of Atlas was sold to Cambridge University as the Titan, and two similar machines were sold for scientific research work, a total production of six machines.

The Ferranti Computer Department became part of ICT in 1963, as work on Atlas was finishing. The new organisation focused on a much larger and wider market, adopting the 1900 series as its product, so that the impetus to develop and sell more Atlas machines faded. As we shall see, however, the investment in skills and innovation was not totally lost.

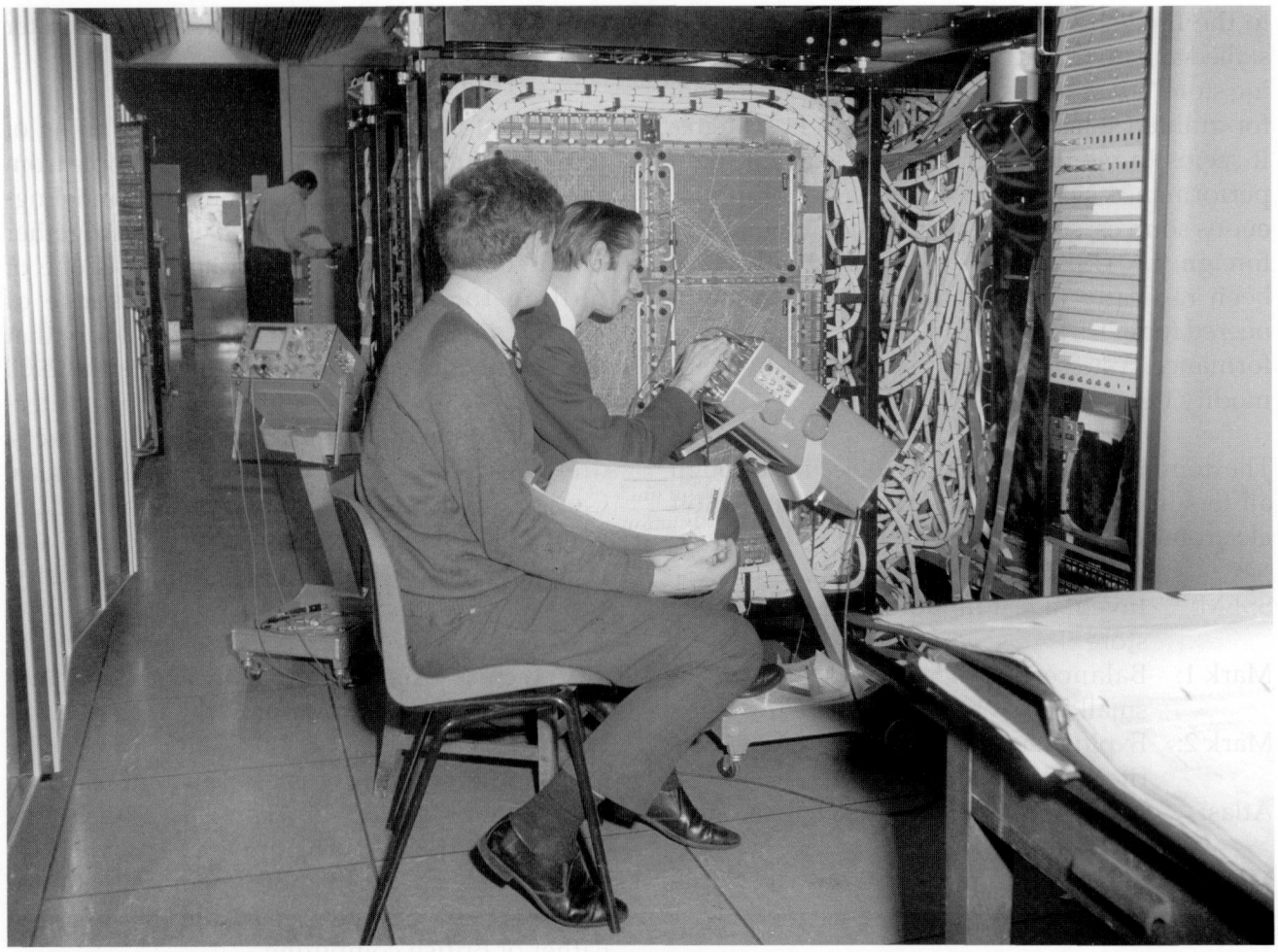


Figure 6: Part of MU5 being commissioned

MU5

The University team under Professor Kilburn embarked on yet another high-performance computer design in the mid-1960s. The target was to be about twenty times the speed of Atlas. Better understanding and analysis of procedural language instruction execution led to an architecture being implemented with a number of special hardware units, such as associatively-addressed slave-stores, address-translation mechanisms and very fast data exchange switching. The instruction set was also improved, so that the machine would be optimized to execute code compiled from high-level language programs. Part of the machine being commissioned can be seen in Figure 6.

As the design and construction of MU5 proceeded, again supported by engineers and production facilities from the West Gorton works, so ICL was embarking on the design of the New Range, which was announced in 1974 as the 2900 series. Although the architecture of New Range drew on many different excellent computer science ideas, there are recognisable similarities between it and MU5. Perhaps this is not surprising, as both systems had similar design objectives. MU5 was really the last big high-performance computer design at the University of Manchester, marking nearly thirty years since the Baby ran its first program [Lavington, 1993].

The Influence of Manchester University on ICL Computer Design

It is most important to remember that ICL's high-performance computer designs owe much to all the main merged companies. The Cambridge EDSAC was re-engineered as LEO 1 in much the same way as the SSEM became the Ferranti Mk 1. Whereas the Ferranti machine was aimed at scientific work, LEO was primarily intended to solve business problems, and the pioneering work on the computerisation of business done by Lyons was an important input to later ICL products. The National Physical Laboratory work on ACE led

to the English Electric Deuce and creation of a skilled and experienced design team. Their "nesting store" ideas in the KDF9 survive as the taken-for-granted stack operations in current systems. It seems to me that the design of current high-performance systems owes little to non-indigenous sources; some ideas may have emerged in foreign (i.e. US) systems, but they would have been radically thought through and re-engineered for ICL's purposes. Of course, lower-performance systems are simply realized by commodity items such as the PC.

The dominant hardware design objective through all this history seems to be how best to exploit electronic data storage:

SSEM:	Invent a fast random access electronic store
Mark 1:	Balance performance/cost using a small fast store with larger slow store
Mark 2:	Exploit better technology storage (Ferrite cores)
Atlas:	Invent one-level store , i.e. virtual storage with paging
MU5:	Exploit efficient use of store (e.g. Name store)

But other innovations leading to efficient high-level language support have evolved through the series of machines; for example index registers:

SSEM:	One
Mark 1:	Seven
Mark 2:	Seven
Atlas:	127
MU5:	One.

This vividly illustrates the eventual satisfying of the compiler writer's wish for the number of registers to be "none, one or infinity"!

Of course, computer designs do not just appear out of abstract architectural ideas. The collective knowledge and experience of the design engineers themselves is what appears in a computer design. Insofar as ICL represents the British computer industry, so all the predecessor companies' history has had some input to current products. However, the close collaboration between the University of Manchester and Ferranti and ICT where many dozens of engineers shared the development work over a prolonged period inevitably created a large pool of expertise conversant with University thinking behind computer design.

After the formation of ICL in 1968, planning of the "New Range" started in earnest. People from the different traditions, i.e., mainly Ferranti, English Electric, ICT, EMI and LEO, worked together on the planning, and my recollection is one of pulling together rather than any sense of blinkered independence. There is no doubt that the "Synthetic Option", which became the 2900 architecture and so through to Nodal Architecture and current high-performance products, drew heavily on MU5 design in several areas. Campell-Kelly cites advanced process-management, efficient high-level language execution and efficient data management [Campbell-Kelly, 1989].

Thus there is an easily-traced thread of continuity in excellent computer design which can be followed all the way back to the SSEM. It would be foolish to ignore the crucial contributions from many other separate sources. That momentous event when the first program ran in June 1948 is recognizable as just like what computers do today—we could proudly regard the "Baby" as the Father of British computing.

Acknowledgements

The author acknowledges his debt to the real computer historians whose works are mentioned in the bibliography, but stresses that errors of interpretation in this paper are entirely his responsibility. Professor Hilary Kahn kindly gave permission for reproduction in this paper of a number of the photographs in the Dept. of Computer Science, University of Manchester.

Bibliography

CAMPBELL-KELLY, M., "ICL—A Business and Technical History", Clarendon Press, Oxford, 1989.

IEEE COMPUTER SOCIETY, Annals of the History of Computing, Special Issue on Computing at the University of Manchester, Vol.15, No.3, 1993.

LAVINGTON, S., "Early British Computers", Manchester University Press, 1980.

LAVINGTON, S., "A History of Manchester Computers", The British Computer Society, 2nd edition, 1998.

LAVINGTON, S., "Manchester Computer Architectures", IEEE Annals of the History of Computing, Vol.15, No.3, 1993.

POLLARD, B.W. and LONSDALE, K., "The Construction and Operation of the Manchester University Computer", Proc IEE, Vol.100, Part II, 1953.

RANDELL, B., (ed), "The Origins of Digital Computers—Selected Papers", Springer-Verlag, 1973.

TWEEDALE, G., "A Manchester Computer Pioneer: Ferranti in Retrospect", IEEE Annals of the History of Computing, Vol.15, No.3, 1993.

WILLIAMS, F.C. and KILBURN, T., "Electronic Digital Computers," Nature, Vol. 162, No. 4117, p 487, September, 1948.

WILLIAMS, M.R., "A Preview of Things to Come: Some Remarks on the First Generation of Computers", Conference on the History of Computing, Paderborn, August 1998 (To be published by MIT Press).

Biography

Christopher P. Burton has a BSc in Electrical Engineering from the University of Birmingham. He is a Chartered Engineer, being a Fellow of the IEE and the BCS. He worked on computer hardware, software and systems developments in Ferranti and then ICT and ICL from 1957 until 1989, nearly all based in the Manchester area. He is a founder member of the Computer Conservation Society and led the team responsible for building a replica of the Small-Scale Experimental Machine, for which he was awarded an honorary MSc by the University of Manchester, a Lovelace Medal by the British Computer Society and an ICL Chief Executive's Gold Award for excellence.

Obituary—E.C.P. Portman

E.C.P. (Charlie) Portman, one of ICL's leading engineers and known to everyone as "Charlie", died on 19th December 1998 on the threshold of his retirement from ICL. He was sixty five.

Charlie Portman had been on the Editorial Board of the ICL Systems Journal and the Editor, who first met Charlie some thirty five years ago, wishes to express his appreciation for his inestimable contributions over many years. His colleagues on the Board will greatly miss his wise counsel.

Colleagues of Charlie Portman, who knew him for much of his professional career, offered the following contributions to this obituary, for which the Editor is extremely grateful.

Peter Hall (retired Director of ICL) writes:

This is difficult to write; not because there is not much to say about Charlie, but because what I want to say is difficult to put across without being accused of exaggeration, the liberal use of cliches, lack of real sincerity, or the comment "he would say that anyway wouldn't he". But he was a special person.

I first met Charlie in the mid 1950's when I found myself the manager of the Ferranti Computer Department. The Department was in a state of turmoil (the previous manager having left under a cloud), with various factions lobbying for their pet projects each requiring funds from our loss making operation. Not a happy ship. The one positive thing, which in itself made for a lot of problems, was that the organisation was dense on the ground with very clever people, most with their own agenda. The prima-donnas amongst them (and there were more than a few) bombarded me with proposals to save the operation, each pointing out that only their ideas would ensure that our computers were the best in the world. Although a senior man, Charlie was not amongst this group. I soon learnt that Charlie was always there for advice and that whatever decision we arrived at, whether consistent with his views or not, he would be a most loyal member of the team, bringing all his most considerable gifts to bear on the job in hand.

During my time at West Gorton Charlie worked on the "neuron" ballot box logic which was used first on a test bed called Newt, which we developed into the small scientific computer Sirius. Sirius was a great success, by our standards and by the standards of the time, but when the same technology was used in the very much larger Orion computer there were considerable difficulties. Charlie led the team which solved the problems. This was a major technical achievement and, although Orion deliveries were much delayed, the system, with its Operating System, OMP, was one of the most sophisticated of its day. I shall never forget Charlie's loyalty and dedication during this very worrying and difficult period. With Orion out of the way, Charlie played a major role in the 1900 story and, indeed, in most of the future West Gorton developments.

I left West Gorton in 1967, and so had less direct knowledge of Charlie's work from then on, but I know that, viewed from Putney, it was clear that his expertise and experience were a major influence on hardware and software development. Others can comment with more authority than I can on his work on the Alvey Demonstrator Project and his contribution to the collaborative multiprocessor work. I am sure however that, even though Charlie had retired a month or so before he died, his technical advice and encour-

agement will be sorely missed by all his old colleagues.

Over the last couple of years or so I have seen much more of Charlie. He was one of Chris Burton's team rebuilding the Manchester "BABY". Even though he was so very ill he put in many hours getting the cathode ray tube stores to work; he was, as always, a dedicated member of the team. I have in my mind a vivid picture of two of my old friends looking very puzzled at the CRT store—Tom Kilburn and Charlie Portman; each in their own way giants of the computer world.

Charlie was never ambitious in the sense of wanting to climb up the ladder of managerial status; his interests were in technical excellence. His ambition was always to produce the best solution to the technical problem set. He enjoyed a technical challenge; the more difficult the better! He was very proud of his ICL Fellowship.

As a man, Charlie must be described as one of nature's gentlemen. At his funeral someone commented to me that he had never heard Charlie say anything unkind about anyone. Neither had I. The fact that so many of his colleagues paid their respects to him on that day said a lot about the way he was thought of within the company. To me he was, for a long time, a valued and loyal member of my team, and for over 40 years a very dear friend.

Chris Burton (for many years a close colleague) writes:

One word which sums up the effect Charlie had on his colleagues is inspirational. For many years he was my manager, but for forty years, for me as for so many others, he was my mentor. He did not autocratically dictate to us what we should do—he advised, he guided, he inspired us with what to do.

Charlie Portman graduated in Electronic Engineering from the University of Liverpool and joined the Ferranti Computer Department in the Magnetic Drum Laboratory in 1954. He was soon involved with the new computers, Mercury and Sirius, where he started to show his grasp of overall systems engineering. In 1960, he became part of the Orion 1 design team where his clear understanding of the interaction of software, hardware and a desired system specification could be brought to bear on pioneering work with multiprogramming. Charlie took the first (unfinished) Orion to AB Turitz in Gothenburg and there completed the hardware and software so that the system was accepted by the customer. This was a major manifestation of his skill in motivating staff of different disciplines to achieve a goal in difficult circumstances. It was probably a defining episode for Charlie himself—he often recalled some of the events of that project with nostalgia and justifiable pride. His star had started to shine!

As the best systems engineer in West Gorton, he led all the new 1900 series hardware developed there, and participated in product planning for these larger systems. Once the hardware designs were established, he took responsibility for all hardware-oriented software for large systems; i.e., test software, executive-type software and design automation. He then carried this work through into the corresponding support for the early large-scale 2900 series machines. His responsibilities were now extensive, he had staff in Manchester, Kidsgrove, Stoke and Stevenage. He met the challenge of these different tasks, which posed novel management problems in the 1960s and early '70s. He had been referred to in Fred Brooks's well-known book, "The Mythical Man-Month," as a person who knew about team-management. Yet he steered clear of the various management fashions of the time. He preferred to think of himself primarily as an engineer.

In the mid-1970s, Charlie turned away, temporarily, from managing big teams. He wanted to get closer to his love of the technically new and exciting. Having started his career in valve electronics, he turned towards advanced developments, particularly the way in which ICL could exploit the falling cost and proliferation of silicon technology, the architecture of very large systems and the role of federated and networked systems, when all these ideas were in their infancy. He had a marvellous technical imagination, which often I felt was nourished by his wide reading and appreciation of classical science fiction. Once in a discussion, when we were struggling to find the right direction, he suggested evocatively that we should think in terms of, "... stepping forward into the future hand-in-hand with our friendly robot."

When the national Alvey Programme was set up, Charlie naturally became the Project Manager of the largest of the Demonstrator Projects, applying Artificial Intelligence ideas to decision support in the Department of Social Security. His reputation as a great guy to work for spread to the two industrial and four academic collaborating organisations and a number of departments in the DSS. After the successful conclusion of the five-year collaboration, he managed those groups inside ICL exploiting the lessons learned about decision support in that large project.

His wisdom and understanding of the structure of large and complex computer systems, and his ability to explain and clarify the issues for other people always attracted colleagues seeking advice on special technology as well as product planning problems within the company. He inspiringly brought out the best from everyone who worked with or for him, yet he modestly played-down his own role in so many of ICL's successes. He firmly believed in the power of individual skills and integrity when faced with distasteful organisational impositions. His own utter honesty, humanity and compassion seemed to be entirely natural, yet were surely reinforced by his wife Sylvia who would often be seen at his side.

Charlie drew much satisfaction from his appointment as an ICL Fellow and then as an ICL Emeritus Fellow, and he was also awarded an ICL Chairman's Gold Medal in 1998. Over the last few years Charlie worked closely with the IC Parc team at Imperial College continuing to inspire and guide them in their research into operational scheduling tools. He also took an active practical interest in the history of our computer industry. He was a key member of the SSEM Rebuild team that rebuilt the "Baby" computer, which was at the centre of the celebrations in Manchester of the Golden Anniversary of the running of the first stored computer program in 1948. Together we were recapturing the fun in our enterprises!

Many people were proud to be counted as a friend or colleague. We will miss Charlie's warmth and wise counsel. We can rejoice that the glow remains of our contact with a man of great intellectual power, of modesty and integrity who had a humane respect for everybody.

Jack Howlett (founder Editor of the ICL Technical Journal) writes:

I had known Charlie for something over forty years, since the mid-1950s in fact. I was then running the Computer Group at Harwell, and we were planning to install our first properly engineered, industrially-produced computer: we had a home-produced machine, and used those of several other organisations, but that's by the way. We had chosen the Ferranti Mercury as the most advanced British machine at that time, and indeed it was a fine machine (it had built-in floating-point arithmetic, which was then quite something), although it was soon to be overtaken by the IBM 704.

Charlie wasn't closely involved in the design and production of Mercury but he knew plenty about it, and I quickly got to know him and to like him greatly. This liking and admiration only Increased as the years went by: he had a remarkable breadth of knowledge, an enviable command of detail, and a striking ability to get to the bottom of any problem put to him and to explain what was often quite complicated electronic behaviour in simple, easily-understandable terms. He certainly contributed greatly to my life and I feel very privileged to have known him.

I often stayed in his lovely, elegant house—the Moat House, in the village of Haigh on the edge of the Lancashire moorlands: not at all far from, of all places, Wigan—non-Lancastrians are always surprised by the beauty of the countryside in these parts. Here his charming wife Sylvia always made me most welcome and to her I send my greetings and most grateful thanks.

Tom Hinchliffe (former Managing Director at West Gorton) writes:

I joined Ferranti at West Gorton in 1960 and stayed there through various mergers until my retirement as Managing Director at West Gorton in 1996. I first met Charlie Portman in 1960 when he was leading the Sirius design team and Charlie became very influential in my early career.

Charlie was to lead many computer design teams in his lifetime and I first worked for him attaching card equipment to Orion. Later, he offered me my first design processor role on 1904E. He

was a brilliant engineer and led from the front by personal example and was never a 'traditional' manager.

He had the ability to visualise problems clearly, to propose elegant solutions and could explain them logically and concisely. I can still remember Charlie explaining the 'paged 1904E' after which the concept of paging seemed so simple that I wondered why I had ever been confused.

Charlie was always willing to spend time talking to his engineers, individually or in groups, and this always seemed to make the job easier. He was a tremendous inspiration and everyone was proud to be part of his team.

He was also good for a laugh. John Allanby was always startling people by dropping the 'venetian blinds' when someone was deeply engrossed in thought. We made a plan and Charlie acted as lookout to warn us when John was returning from tea, so we could get our own back. The 'air-horns' we had attached to the 'Run' switch, froze John to the spot!

Charlie also gave my wife her first driving lesson in a 'Formulae 1 go-kart', in the rear car park at West Gorton. Eva did not drive but Charlie persuaded her that driving was simple and that she should have her turn. He told her to hold the clutch pedal down while he put the kart in 1st gear. He then said just let the clutch out and the kart will move slowly—the kart moved quickly and she screamed, he shouted use the brake, she chose the accelerator and I can still see Charlie chasing the kart across the car park in panic. Fortunately, Eva found the right pedal and the kart stopped just short of a very large brick wall, with Charlie close behind.

There are many, many stories about Charlie and all of them, whether professional or social, always highlight Charlie's real enthusiasm, his genuine care and consideration for other people and his brilliant engineering ability.

Charlie won an ICL Chairman's medal last year and was one of only nine medal winners. These were presented at an event hosted by Michio Naruto to celebrate 50 years of Mainframe Engineering Achievement and I had the pleasure of writing the citations for most of the medal winners including Charlie. Unfortunately, Charlie was too ill to receive his medal in person and I am repeating the wording of his citation here, so that everyone can appreciate the tribute that was read out for Charlie at the event.

Chairman's Medal Citation

Charlie Portman : Ferranti 1954 – ICL 1998

Charlie Portman joined Ferranti Limited at Moston in 1954 to work on Magnetic Drum systems but soon became involved with processor design and led the development teams for Sirius and Orion.

It was Charlie who went on to lead the development of all the 1900 series machines at West Gorton.

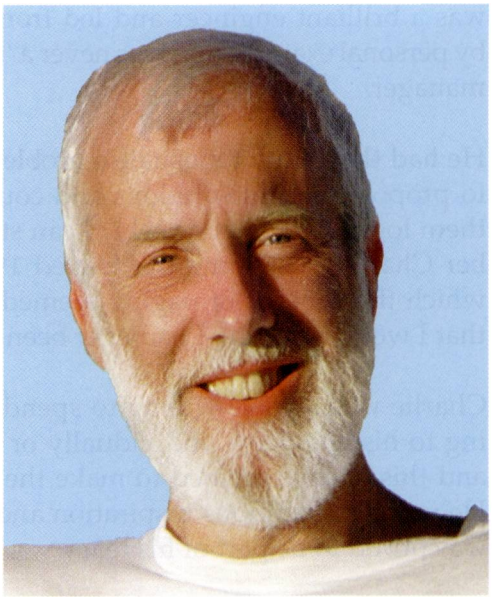
In 1972, he moved to software, taking responsibility for all test software and 1900 executives. Later responsibility for 'Design Automation' and early 2900 systems were added.

In 1984, Charlie became Project Manager of the largest UK Alvey Directorate Project, applying artificial intelligence ideas to decision support in the DHSS. This five-year project involving many major external collaborations was completed successfully. Since then he has continued with knowledge engineering and is currently working with Imperial College, London.

His reputation for understanding the real technical issues and explaining them in simple terms is second to none and he is a real inspiration to everyone he works with. Charlie is an ICL Fellow and retired earlier this year.

T.A.Hinchliffe
Citation date : 9th November 1998

Obituary—B.J. Procter



Brian J. Procter, one of ICL's outstanding sytems engineers, died suddenly on 24th February, 1999, shortly after his retirement from ICL.

Two colleagues of Brian Procter, who had known him throughout much of his career, offered the following contributions to this obituary. These notes, for which the Editor is extremely grateful, are based on tributes made at his memorial service.

Tom Hinchliffe (former Managing Director at ICL, West Gorton) writes:

Brian was a brilliant computer engineer who made an outstanding technical contribution to ICL, throughout the whole of his career and established a very distinguished personal reputation right across the computer industry. His lifetime achievements have rightly earned him a place in the 'ICL Hall of Fame'. Brian joined EMI in 1955 and worked on the EMI-DEC machines. When ICT was formed in 1963, he transferred to Stevenage to lead the teams designing the smaller 1900 processors.

Brian's personal landmark machine was the highly acclaimed 2903 which positively bristled with innovation and, together with the 2904 variant, achieved sales of over 2000 systems—an amazing achievement in those days.

In 1976 Brian relocated to West Gorton where he introduced many new innovations to the New Range of ICL computers—the 2900 systems and Series 39—including Nodal Architecture and the use of fibre optics some five years ahead of IBM. He also personally championed the technical development of 5th Generation Parallel Processing across Europe.

Most recently, as the High Performance Systems, Chief Architect, Brian was responsible for the technical direction of the new ICL Millennium system—a role in which he continued until his untimely death. He had remained at the leading edge of computer technology for forty three years.

I have worked with Brian for over twenty five years and I can think of no-one who has made a bigger or better technical contribution to ICL and Mainframe Systems than Brian. Brian was made one of the first ICL Fellows in 1991—a tribute to his tremendous technical talent. Last year he was also awarded an ICL Chairman's medal for 'Outstanding Engineering Achievement' over the last 50 years—one of only nine medals that have been awarded.

As will be appreciated Brian made a leading contribution to the development of the British Computer Industry and to ICL in particular.

I feel very privileged to have worked with Brian, as I am sure we all do. We will always remember him with great affection and friendship. He set a wonderful example to us all. Above all, he enjoyed his job and worked amazingly hard. He had the ability to express complex technical matters in an easily understandable way, he never sought glory for himself and never had a bad word to say about anybody.

He was both a brilliant engineer and a tremendous inspiration to all his colleagues but was, at the same time, a quiet, humble and thoroughly likeable person.

ICL was very lucky to have such a great engineer and I was very fortunate to have such a good friend.

Nic Holt (a close colleague for many years) writes:

Tom has paid eloquent tribute to Brian's outstanding achievements and his unique contribution to ICL's technological strategy over a period spanning four decades. I'd like to offer a personal appreciation—a token of gratitude—on behalf of all of us who have had the pleasure of working with Brian over the years.

Brian joined the industry in its infancy. They were exciting, those early days, a voyage into the uncharted waters of new technologies. The sophisticated tools we take for granted now did not exist, and the pioneers needed to be resourceful and inventive, and truly multi-disciplinary. Brian retained that spirit of adventure in everything he undertook thereafter.

He was fascinated by new concepts and technologies, eager to understand them and to explore the opportunities they presented. He was always prepared to take time to share his knowledge and understanding—and his enthusiasm—with others. His evident delight and zeal in tackling these new challenges were utterly infectious, a profound source of inspiration for engineers and others alike.

Brian's technical expertise and incisiveness were widely respected, and he invariably brought fresh insights to complex topics when working with leading researchers and technologists; but he was equally adept (and patient) at explaining and clarifying difficult issues in terms that were easily understood by those less knowledgeable than himself—new graduates, non-technical staff, senior management, as well as customers and partners.

Brian's enthusiasm and guidance ensured the sound development of many of ICL's most successful engineers. For them he was a willing and stimulating mentor, always ready to listen to their ideas and offer perceptive advice and encouragement, a trusted friend and steadfast supporter. Consequently, highly motivated teams always formed naturally around Brian.

Brian greatly enjoyed discussing new ideas, which would be critically evaluated from every angle, turned on their heads and every side avenue explored. Invariably, new lines of thought would emerge, other trails to follow—sometimes far removed from the original starting point. Working with Brian was truly uplifting, and always a pleasure.

Brian was immensely thorough in everything he did—indeed, a perfectionist. Before embarking on any project, the background would be minutely researched—whether some new technology, powered sprays and scaffolding for painting the outside of his house, or equipment for his photography. Every source would be consulted, every option analysed until he was sure of his ground. Design weaknesses would be quickly identified and ingeniously rectified—he always delighted in identifying improvements, quite prepared, if necessary, to invent his own solutions. He would then work with dedication to complete what he had set out to achieve.

Thomas Edison said that "Genius is one percent inspiration and 99 percent perspiration". In Brian's case such arithmetic is too simplistic. As one colleague put it: "He was an inspirational guy", and he always gave 100 percent of himself to everything he did, professionally and personally. And so each of us will always remember Brian with great affection, for his modesty, for the caring and personal interest he took in everyone around him, and above all for his gentle nature and unfailing kindness.

Previous Issues

Vol. 13 Iss. 1 – Autumn 1998

Guest Editorial
An Architecture for Commercial On-line Internet Services
The Enterprise Datacentre—ICL's "Millennium" Programme
Trimetra DY and the Emulation of OpenVME on Intel Hardware
Trimetra UNS
Trimetra *Xtra*server
Millennium Data Access

Vol. 12 Iss. 2 – November 1997

Workflow—A Model for Integration
SuperVISE—System Specification and Design Methodology
Process Modelling using the World Wide Web—ProcessWise™ Communicator
Mobile Applications for Ubiquitous Environments
Middleware Support for Mobile Multimedia Applications
INDEPOL Client—A 'facelift' for mature software
Using the ECL^{i}PS^e Interval Domain Library in CAD

Vol. 12 Iss. 1 – May 1997

Java™—An overview
Mobile Agents—The new paradigm in computing
The SY Node Design
Discovering associations in retail transactions using Neural networks
Methods for Developing Manufacturing Systems Architectures
Demystifying Constraint Logic Programming
Constraint Logic Programming
ECL^{i}PS^e—A Platform for Constraint Programming

Vol. 11 Iss. 2 – January 1997

The Year 2000 Problem
Working with Users to Generate Organisational Requirements:
The ORDIT Methodology
Network computing with remote Windows
Neural Networks
Short-term currency forecasting using neural networks
Helping Retailers Generate Customer Relationships
The Systems Engineering Excellence Model
Cochise: a World Wide Web interface to TPMS applications

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 ECL^{i}PS^e
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 Businèsses
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 Syste

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
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

Josie Abbey
Group Technical Directorate
ICL, Westfields, West Avenue, Kidsgrove, Stoke-on-Trent, ST7 1TL
Telephone +44 (0)1782 794815
Email: Josie.Abbey@icl.com

or

The Editor, V.A.J. Maller
Telephone +44 (0)1438 833514
Email: V.A.J.Maller@lboro.ac.uk
or Victor.Maller@icl.com

ICL Systems Journal

Guidance for Authors

Content

The ICL Systems Journal has an international circulation. It publishes papers of a high standard that are related to ICL's business and is aimed at the general technical community and in particular at ICL's users, customers and staff. The Journal is intended for readers who have an interest in computing and its applications in general but who may not be informed on the topic 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 not usually 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 material relating to unannounced products of any of the ICL Group of companies.

Letters to the Editor and book reviews may also be published.

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 employed by a company in the ICL Group. All papers will be judged on their merit, irrespective of origin.

Length

There is no fixed upper or lower limit, but a useful working range is 4,000-8,000 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.

Abstract

All papers should have an Abstract of approximately 200 words, suitable for the various abstracting journals to use without alteration.

Presentation

Printed (typed) copy

A typed copy of the manuscript, single sided on A4 paper with 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.

Electronic version

Authors are encouraged to submit either a magnetic disk version of their manuscript or a compressed e-mail attached file or both. The format of the file should conform to the standards of any of the widely used word processing packages or be a simple text file.

Diagrams

Line diagrams will usually be redrawn and professionally lettered for publication, so it is essential that the originals 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 be numbered for reference in the text and the text marked with the reference and an appropriate caption to show where each should be placed. Authors should check that all diagrams are actually referred to in the text and that copies of all diagrams referred to are supplied. If authors wish to submit drawings in an electronic form, then they should be separated from the main text and, ideally, be in the form of EPS files. If an author wishes to use colour, then it is very helpful that a professional drawing package be used, such as Adobe Illustrator. PowerPoint diagrams may be used to indicate the author's requirements.

Tables

As with diagrams, these should all have captions and reference numbers. If they are to be provided in electronic form, then either a standard spreadsheet (Excel) should be used or the data supplied as a file of comma/tab separated variables. A printed version should also be supplied, showing all row and column headings, as well as the relevant units for all the quantities tabulated.

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.

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.

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, to conform to the Journal's general style for spelling, punctuation or notation, will be made by the Editor.

Proofs, Offprints

Printed proofs are sent to authors for correction before publication. The Editor will, however, always be prepared to send electronic versions to authors, either in PDF or as output files from the production system used for the Journal: PageMaker, Illustrator and Photoshop.

Copyright

ICL Systems Journal Spring 1999

ICL Group Technical Directorate
Westfields
West Avenue
Kidsgrove
Stoke-on-Trent ST7 1TL