ICL TECHNICAL JOURNAL

Volume 7 Issue 2 November 1990

Published by
International Computers Limited
at
Oxford University Press

AN STC COMPANY

ICL TECHNICAL JOURNAL

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

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

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

1990 subscription rates: annual subscription £40 UK, £44 EEC, £48 rest of world, US $95 N. America; single issues £22 UK, £25 EEC, £27 rest of world, US $52 N. America. Orders with remittances should be sent to the Journals Subscriptions Department, Oxford University Press, Pinkhill House, Southfield Road, Eynsham, Oxford OX8 1JJ.

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

ICL

TECHNICAL JOURNAL
Volume 7 Issue 2

Contents

French Translations of Abstracts **iii**

Dutch Translations of Abstracts **ix**

The New SX Models **193**

Foreword **195**

The SX Node Architecture
J.R. Eaton, G. Allt and K. Hughes **197**

SX Design Processes
G.P. Abraham, D.C. Freeth and H. Vosper **212**

Physical Design Concepts of the SX Mainframe
C. Shaw **233**

Papers arising from collaborative R & D **249**

Foreword **251**

The Development of Marketing to Design:
The Incorporation of Human Factors into Specification and Design
A.T.F. Hutt and F. Flower **253**

Foreword **270**

Advances in the Processing and Management of Multimedia Information
M.H. O'Docherty, P.J Crowther, C.N. Daskalakis, C.A. Gobel, M.A. Ireton, S. Kay and C.S. Xydeas **271**

An Overview of Multiworks
M.E. Morris and I. Cole **288**

RICHE-Rèseau d'Information et de Communication Hospitalier Européen (Healthcare Information and Communication Network for Europe)
T. Drahota **296**

E.S.F. – A European Programme for Evolutionary Introduction of Software Factories
R. Thomas, C. Fernstroem and O. Hesse **307**

A Spreadsheet with Visible Logic
V. West and E. Babb **319**

Intelligent Help – The Results of the EUROHELP Project
M. Smith and C. Tattersall **328**

How to use Colour in Displays – Coding, Cognition and Comprehension
D. Van Laar and R. Flavell **362**

Further papers

Eye Movements for A Bidirectional Human Interface
R. Epworth **384**

Government IT Infrastructure for the Nineties (GIN); An Introduction to the Programme
P. Wiles **412**

Book Review **432**

Notes on Authors **435**

Guidance for Authors **444**

Résumés

J. R. Eaton, G. Allt *et* K. Hughes
Corporate Servere Group, ICL, W. Gorton, Manchester, UK
L'Architecture Nodale SX, p. 197

SX est la dernière génération de processeurs de Série 39 de haute performance. Elle est compatible avec les systèmes de la Série 39 existants mais offre de meilleures performances aussi bien au niveau des configurations simples que des configurations multi-nodales. SX offre également une connectivité E/S améliorée et de plus grandes performances pour répondre à l'augmentation de la puissance de traitement. Cet article décrit certaines des caractéristiques les plus importantes du point nodal.

G. P. Abraham, D. C. Freeth *et* H. Vosper
Corporate Servers Product Group, ICL, West Gorton, Manchester, UK
Processus de Conception SX, p. 212.

Cet article présente un survol des processus de conception appliqués lors de la conception du système SX.

Il couvre certaines des initiatives et innovations majeures et souligne les méthodes extensives utilisées dans la vérification de conception.

L'outillage décrit inclut les modèles de performance pour OCP, E/S et internode, techniques d'émulation de logiciel et simulation de matériel et logiciel d'essais.

Ce papier contient également une description des méthodes de mesure et de contrôle de conception.

C. Shaw
Product Servers Group, ICL, W. Gorton, UK
L'emballage du SX, p. 233.

Les améliorations spectaculaires au niveau des densités et de la vitesse d'intégration de circuits à puces de silicium, intervenues depuis plus de vingt ans, sont bien connues. L'exploitation efficace de la technologie LSI (Intégration à Grande Echelle) moderne, dans une unité centrale de hautes performances, implique des exigeances pour le matériel et l'emballage supportant les sous-systèmes électroniques.

Cet article décrit les implications de ces demandes sur la conception physique de l'unité centrale ESSEX et souligne certaines des solutions de conception résultantes. Le contenu des sous-systèmes de matériel fonctionnel majeurs au sein du Noeud ESSEX est résumé.

Dr. A. T. F. Hutt
ICL Systems Engineering, Systems Integration Division, Reading
Fiona Flower
ICL CPS Professional Services, Information Technology Services Division, Bristol
Le Développement du Marketing à la Conception, p. 253

Cet article décrit la façon dont les recherches ont été exploitées à partir des trois projets Alvey pour produire la méthodologie ICL "du Marketing à la Conception". Cette méthodologie incorpore des facteurs humains dans la spécification et la conception du produit, permettant aux équipes de conception de développer des produits qui sont plus petits, plus concentrés et donc plus acceptables pour leurs utilisateurs. ICL applique la méthodologie à une série de cinq ateliers, supportée par l'utilisation de manuels dans l'atelier et sur le lieu de travail. La méthodologie "du Marketing à la Conception" représente donc l'une des exploitations pratiques les plus considerables acquises dans la recherche aux facteurs humaine entrepris au travers du parrainage du programme Alvey.

M. H. O'Docherty, P. J. Crowther, C. N. Daskalakis, C. A. Goble, M. A. Ireton, S. Kay *et* C. S. Xydeas
Multimedia Information Systems Research Group (Groupe de Recherche sur Systèmes d'Informations Multimédias), Victoria University of Manchester, UK
Progrès dans le Traitement et la Gestion des Informations Multimédias, p. 271.

Les systèmes de base de données courants sont efficaces pour le traitement de données simples telles que les attributs numériques ou textuels. Mais ceux-ci ne sont pas bien adaptés aux autres médias sauf pour les systèmes d'archivage utilisant des étiquettes générées manuellement. Au contraire, un Système d'Informations Multimédias est destiné à traiter tous les types de données électroniques d'une manière uniforme. Des exemples de données multimédias sont le texte, les images et la voix. De nombreuses applications de traitement de l'information sont de nature multimédia mais les technologies nécessaires à la mise en oeuvre de systèmes multimédias pour de telles applications n'ont commencé à faire leur apparition que récemment.

Cet article examine tout d'abord les caracteristiques nécessaires à un système d'informations multimédias idéal. Celles-ci incluent un environnement orienté vers l'objet pour traiter les données multimédias, la récupération de toutes les données par contenu utilisant des techniques d'analyse sémantique qui peuvent être adaptées à chaque nouvelle application, un modèle convenable de données orienteés vers l'objet tel que ODA, une stratégie d'interrogation soigneusement conçue avec des degrés de confiance permettant l'appariement inexact et enfin des facilités de passage en revue et d'édition vastes.

Les prototypes, qui d'une certaine façon répondent à ces objectifs, sont décrits. Leurs limitations sont spécifiées et celles-ci sont suivies par une description du système prototype développé à l'Université de Manchester qui a pour objectif de répondre aux exigences du système idéal.

M. B. Morris *et* I. Cole
STC Technology Ltd., London Road, Harlow, Essex, CM17 9NA, UK
Un Survol de Multiworks, p. 288

Cet article offre un simple survol de la fonctionnalité prévue de Multiworks (Poste de travail intégré multimédia) un Projet d'Intégration de Technologie (No. 2105) supporté par ESPRIT II.

L'objectif de Multiworks consiste à développer un poste de travail pour le professionnel, traitant la voix, le son, les graphiques à 2 et 3 dimensions, la vidéo et le texte. Le but est de fournix aux multimédias comparables à ceux couramment disponibles des facilités pour les médias traditionnels de textes et graphiques.

Le projet est élaboré selon une approche par étape pour la conception et le développement, avec des progiciels de travail dédiés à la conception d'une plateforme de configuration minimum, un environnement de programmation multimédia, un environnement d'applications et l'environnement d'utilisateur final. Les exigences de l'utilisateur sous la forme de scénarios ont été utilisés pour gérer la configuration et l'intégration des différentes étapes.

Tony Drahota
International Computers Ltd, Slough, Berks, UK
RICHE–Réseau d'Information et de Communication Hospitalier Européen (Healthcare Information and Communication Network for Europe). Esprit Project 2221, p. 296.

Le projet RICHE a été élaboré et organisé pour répondre aux exigences du secteur de la santé et obtenit une utilisation plus efficace de la technologie d'information. Un grand consortium de sociétés de technologie d'information et d'organisations de santé européennes ont établi, en 1989 pour définir et diriger les mises en application, une architecture pour les systèmes de santé publique qui s'appliquera, sur une base d'uniformité, dans tous les pays européens et qui reflètera le besoin d'intégration de fonctions de support pour les activités professionnelles, administratives, techniques et de gestion de la santé publique. La première phase d'activités de ce projet ambitieux s'étalant sur trois années, engagé actuellement dans sa seconde années, a été couronée de succes–l'analyse et l'étude des exigences globales et la définition initiale de l'architecture globale. Cet article resumé les résultats actuels du projet et les résultats escomptés à partir des travaux d'études et de recherche actuels et futurs.

Richard Thomas
Project Director–ICL
Christer Fernstroem
Project Technical Director–Cap Gemini Innovation
Olaf Hesse
Market Integration Manager for Project–University of Dortmund
ESF–Un Programme Europeen pour Introduction Evolutive D'usines de Logiciel, p. 307

Le logiciel est devenu un facteur primordial sur lequel repose le succès économique. L'aptitude à fournir des logiciels pour un nombre croissant de produits et de services déterminera le succès économique futur des sociétés et corporations. Ceci aura également des conséquences majeures sur le succès économique national.

Au sein de l'industrie IT (Information Technologie–Informatique), la demande en logiciels est un souci majeur pour les gros utilisateurs les intégrateurs de système et les fournisseurs de produits, et d'un intérêt énorme pour les groupes de techologie de pointe et les centres de recherche.

Cet article présente un survol de l'entreprise de coopération européenne de qui aborde le problème général fourniture de logiciel et élabore une stratégie de solution pour les années 1990 et le siècle prochain. L'essential de la solution ESF consiste en la construction d'usines spécialisées en logiciels, pour différentes industries, à partir d'un marché de composants à fournisseurs multiples; la spécialisation de l'usine étant obtenue par modelage du processus et support de processus actif. L'usine construite supportera tous les types d'utilisateur, par exemple: gestion, technique, administratif.

Vincent West *et* Edward Babb
Strategic System, ICL, Bracknell, UK
Un Tableau Financier avec Logique Visible, p. 319

Cet article présente le protype d'un tableau financier logique specimen développé au sein de Visilog, un projet EUREKA impliquant ICL et BULL. L'objectif de Visilog (Logique Visible) est de tirer la capacité maximale de la logique disponible pour l'utilisateur final, grâce à des graphiques d'une utilisation facile.

Ce papier traite des capacités du tableau financier, illustrant par des exemples les avantages par rapport à un tableau financier conventionnel, qui inclue des réponses généralisées, la recherche de l'objectif, la gestion des alternatives et des inégalités, et la programmabilité (logique). Cet article décrit également l'architecture interne, en particlier le moteur logique utilisé, le LPL (Langage de Logique Pure), qui résulta d'un projet ALVEY impliquant ICL, Imperial College et Edinburgh University.

Mick Smith
ICL, Strategic Systems, Manchester, UK
Colin Tattersall
University of Leeds, computer based learning unit
Prévision d'une aide intelligente et son impact sur le développement d'application, p. 328.

Une aide intelligente se distingue d'une aide conventionnelle par deux aspects importants. Premièrement, l'aide est générée par des techniques basées sur des études empiriques et scientifiques de la nature et des exigences de cette assistance. Deuxièmement, parce qu'elle est générative, elle peut être adaptée dynamiquement à la connaissance de l'utilisateur, la tâche en cours, et l'état de l'application. Ceci contraste avec l'aide conventionnelle où l'effort est porté sur les mécanismes d'accès aux textes déjà memorisés.

L'approche des dispositions pour une aide intelligente développée par le projet EUROHELP nécessite quatre éléments majeurs: un Noyau d'Aide Intelligente (IHSS) qui forme le système de déroulement générique, un formalisme de Modelage d'Application pour représentation d'une application au IHSS, un Système de Développement du Système d'Aide (HSDS) comme support de la construction des Modèles d'Application, et finalement une Méthodologie de Développement du Système d'Aide (HSDM) comme guide du processus entier.

Cette approche nécessite certaines interfaces avec l'application elle-même, afin que l'IHSS puisse avoir accès aux informations contextuelles et de l'utilisateur. De plus, et pour une exploitation efficace, ceci implique certains changements dans les méthodes, l'équipement, la conception et la réalisation du développement d'application.

L'article décrit les leçons tirées du projet EUROHELP concernant la nature de l'aide intelligente, les bénéfices potentiels à tirer d'une telle assistance et les techniques pratiques pour prendre de telles dispositions. Sur un plan général, il traite de l'impact de ces dispositions pour une aide intelligente sur le développement d'applications.

Darren Van Laar *et* Richard Flavell
The Management School, Imperial College, London, UK
Comment Utiliser la Couleur dans les Affichages. II. Codage, Connaissance & Compréhension, p. 362.

La couleur est souvent utilisée sur les écrans visuels pour améliorer les interactions homme-ordinateur et pour obtenir un affichage plus naturel et plus représentatif que le simple codage monochrome. Afin d'utiliser efficacement la couleur, il est important de comprendre comment elle est perçue, comment le codage couleur peut être utilisé comme support de tâches et de savoir combien de couleurs doivent être utilisées sur un affichage et quand la couleur devient utile.

Il s'agit du second de deux articles consacrés à l'utilisation efficace des couleurs dans les affichages à tube cathodique et couvre les facteurs de tâche dans les affichages couleurs, explique ce qui est entendu par codage couleur, quand utiliser les différentes formes de codage couleur, et l'effet de l'utilisation de la couleur sur les performances et les préférences. Cet article couvre également les facteurs d'esthétique et d'environnement applicables à l'emploi de couleur sur les affichages.

L'article se conclut par un résumé des facteurs physiques et cognitifs qui ont été soulignés dans les deux articles, ainsi que par quelques indications pour mettre ces recommandations en pratique. Un glossaire des termes fréquemment utilisés est également inclus.

Richard Epworth
Technology Strategy Manager, STC Technology Ltd., London Road, Harlow, Essex CM17 9NA, UK
Mouvements Occulaires, pour une Interface Humaine Bidirective, p. 384

Lorsque nous regardons un écran, nous ne savons pas précisément où nos yeux se posent et pourtant nos mouvements occulaires contiennent des informations importantes. Un système opto-électronique, qui surveille nos mouvements occulaires, peut être utilisé pour communiquer avec une machine. Les articles sur un écran peuvent être sélectionnés simplement en les regardant et les actions peuvent être lancées par des mouvements occulaires volontaires. Un tel système peut également être conscient de nos intérêts et difficultés et peut répondre à nos besoins, même ceux dont nous ne sommes pas conscients. Ce document décrit les techniques de mesure du mouvement occulaire, les applications possibles des ces informations, et plusieurs démonstrations pratiques d'une interface contrôlée par l'oeil.

Peter Wiles
ICL Central Government and Defence Business Unit, Slough, Berks, UK
Une introduction à la programme GIN (Government Infrastructure for the Nineties), p. 412.

Cet article décrit le raisonnement qui a mené à la création d'un projet ICL majeur, le Programme GIN. GIN (Government Infrastructure for the Nineties) signifie Infrastructure Gouvernementale pour les années 90. Il s'agit d'un programme visant à la livraison d'un ensemble intégré, interactif de produits UNIX et PC gérés via une gamme de produits de Gestion de Système.

Le programme GIN répond aux stratégies IT de plusieurs clients majeurs, en particulier certains des plus grands Administrations. Le Gouvernement ne dispose pas de nombreuses applications industrielles, mais exige toute une infrastructure cohérente, avec une ligne intégrée de produits qui soit en avance par rapport a l'ensemble du marché. L'article décrit la façon dont l'infrastructure a été déterminée et la manière dont ICL a l'intention de poursuivre le développement des produits appropriés.

Samenvattingen

J. R. Eaton, G. Allt *en* K. Hughes
Corporate Services Group, ICL, W. Gorton, Manchester.
De SX Node Architectuur, p. 197

SX is de jongste generatie hoog presterende Serie 39 processors. Deze zijn compatibel met bestaande systemen uit Serie 39 maar bieden hogere prestaties bij zowel enkele als meervoudige knooppunt en (nodes). Tevens biedt SX betere I/O aansluitbaarheid en hoger prestatievermogen om met het verhoogde verwerkingsvermogen gelijke tred te kunnen houden. Dit artikel beschrijft enige van de belangrijkere kenmerken van het knooppunt.

G. P. Abraham, D. C. Freeth *en* H. Vosper
Corporate Servers Product Group, ICL, West Gorton, Manchester.
SX Ontwerp Processen, p. 212.

Dit artikel geeft een overzicht van de ontwerpprocessen die bij het ontwerp van het SX systeem gevolgd werden.

Het bestrijkt sommige van de voornaamste initiatieven en innovaties en legt speciale nadruk op de uitgebreide methoden welke in ontwerpverificatie toegepast worden.

De omschreven gereedschappen zijn o.m. prestatiemodellen voor OCP, I/O en inter-knooppunt software emulatietechnieken, hardwaresimulatie en test software.

Het artikel bevat tevens een beschrijving van de ontwerpbesturing en meetmethoden.

C. Shaw
Product Services Group, ICL, West Gorton, UK
Het ontwerp de SX-mainframe, p. 233.

De spectaculaire verbeteringen de dichtheid von circuitintegratie en snelheid van de silicon chip, die nu reeds twintig jaar aan de gang zijn, zijn alom bekend. De effectieve exploitatie van moderne LSI technologie in een mainframe computer met hoog prestatievermogen stelt hoge eisen aan de hardware en de packaging die de elektronische subsystemen ondersteunen.

Dit artikel beschrijft de implicaties van deze eisen op het fysieke ontwerp van de SX mainframe en geeft enige van de ontwerpoplossingen in grote lijnen weer. De inhoud van de voornaamste functionele hardware subsystemen binnen het SX knooppunt worden in het kort weergegeven.

Dr. A. T. F. Hutt
ICL Systems Engineering, Systems Integration Division, Reading
Fiona Flower
ICL CPS Professional Services, Information Technology Services Division, Bristol
De Ontwikkeling van 'Marketing to Design', p. 253

Dit artikel beschrijft hoe het research van drie Alvey projectan gebruikt werd bij het ontwerp van de 'Marketing to Design' methodologie van ICL. Bij deze methodologie worden menselijke factoren in produktspecificatie en ontwerp opgenomen, zodat ontwerpteams producten kunnen ontwikkelen die kleiner en beter afgestemd zijn, en zodoende beter geschikt voor hun gebruikers. ICL presenteert de methodologie op het ogenblik d.m.v. een serie workshops, begeleid door werkboeken en handleidingen voor gebruik in de werkplaats en op de werkplek. 'Marketing to Design' vertegenwoordigt zodoende een van de belangrijkste praktische toepassingen die voortgekomen zijn uit het research naar menselijke factoren dat onder het sponsorschap van het Alvey programma uitgevoerd is.

M. H. O'Docherty, P. J. Crowther, C. N. Daskalakis, C. A. Goble, M. A. Ireton, S. Kay *en* C. S. Xydeas
Multimedia Information Systems Research Group, Victoria University of Manchester
Ontwikkelingen in de Verwerking en Management van Multimedia Informatie, p. 271.

Huidige databasesystemen zijn efficient in de verwerking van eenvoudige data zoals numerieke of tekstattributen. Zij zijn niet erg geschikt voor andere media, uitgezonderd voor gebruik als archiefsystemen met behulp van handmatig gegenereerde labels. Daartegenover is een Multimedia Informatie Systeem bedoeld om alle typen elektronisch gemodelleerde data op uniforme wijze te verwerken. Voorbeelden van multimedia data zijn tekst, beelden en spraak. Vele informatieverwerkende toepassingen zijn multimedial van aard, maar de technologiën die nodig zijn om van multimedia systemen voor zulke toepassingen te implementeren hebben slechts kort geleden hun intrede gemaakt.

Dit artikel onderzoekt eerst de kenmerken die een ideaal multimedia informatie system moet bezitten. Dit zijn o.m. een object-georienteerd omgeving voor verwerking van multimedia data, de opvraging van alle data volgens inhoud m.b.v. semantische analysetechnieken die aan elke nieuwe toepassing aangepast kunnen worden, een geschikt model van object-georienteerde data zoals ODA, een zorgvuldig ontworpen opvraag-strategie met betrouwbaarheids niveaus met de mogelijkheid tot inexact selectief collationeren, en tenslotte uitgebreide dourblader en opmaakfaciliteiten.

Verder worden prototypen die enigermate aan deze doelen voldoen beschreven. De beperkingen hiervan worden aangegeven en dit wordt gevolgd door een beschrijving van het prototype dat op Manchester University ontwikkeld wordt met het doel aan de vereisten van het ideale systeem te voldoen.

M. E. Morris *en* I. Cole
STC Technology Ltd. Harlow, Essex, UK
Een Overzicht van Multiworks, p. 288.

Dit artikel geeft een eenvoudig overzicht van de bedoelde functionaliteit van Multiworks (MULTI-media intergrated WORKStation) een Technologie Integratie Project (Nr. 2105) ondersteund door ESPRIT II.

Het doel van Multiworks is de ontwikkeling van een werkstation voor een op kantoor werkende professional, dat spraak, geluid, 2D en 3D graphics, video en tekst manipulieren kan. Dit project hoopt dezelfde voorzieningen voor multi-media beschikbaar te maken die nu reeds bij de traditionele media voor tekst en graphics in gebruik zijn.

Het project volgt een gelaagde ontwerpen ontwikkelingsbenadering, met werkpakketten speciaal voor het ontwerp on van een fundamenteel hardware platform, een multi-media programmeeromgeving milieu, een toepassings omgeving en het eingebruikersomgering. Eisen in de vorm van scenario's worden gebruikt voor het aansturen van de configuratie en integratie van de verschillende lagen.

Tony Drahota
ICL Slough, UK
RICHE–Reseau d'Information et de Communication Hospitalier Europeen. Esprit Project 2221 (Informatie en communication et werk voor de Europese gezondheidszorg). p. 296.

Het RICHE project werd in het leven geroepen en georganiseerd als antwoord op de eisen van de gezondheidszorgsector voor effectiever gebruik van informatietechnologie. Een groot consortium van Europese IT ondernemingen en gezondheidszorg organisaties begon in 1989 met het omschrijven en opzetten van pilotimplementaties en een architectuur voor gezondheidszorgsystemen voor uniforme toepassing in alle Europese landen, waarin de behoefte aan standaardisatie van geïntegreerde, begeleidende functies voor professionele, administratieve, technische en management activiteiten te standaardiseren tot uitdrukking komen. De eerste fasede analyse en modellering van de algemene vereisten en de aanvankelijke definitie van de globale architectuur – van dit ambitieuze driejarenproject, dat nu in zijn tweede jaar is, is nu met succes voltooid. Dit artikel bevat een overzicht van wat de project resultaten tot heden en kijkt vooruit naar sommige van de resultaten die van de huidige en toekomstige werkzaamheden verwacht kunnen worden.

Richard Thomas
ICL, Project Director
Christer Fernstroem
Cap. Gemini Innovation, Project Technical Director
Olaf Hesse
University of Dortmund, Market Integration Manager for Project
ESF–Een Europees Programma voor evolutionaire introductie van Software Fabrieken, p. 307.

Software is een belangrijke en onontbeerlijke factor geworden voor economisch succes. Het vermogen om software te kunnen leveren voor het toenemende aantal produkten en diensten die van software afhenkelijk zijn, is van doorslaggevend belang voor het toekomstige commerciële succes van ondernemingen. Dit heeft tevens grote invloed op het economische succes van de betreffende landen.

Binnen de IT industrie is de vraag naar software een reden voor grote bezorgdheid onder grootschalige gebruikers, systeemintegrators en leveranciers van produkten en van enorm belang voor geavanceerde technologiegroepen en researchcentra.

Dit artikel geeft een overzicht van een Europees samenwerkingsverband om het probleem van softwareaanbod aan te pakken en met een strategie. De essentie van deze ESF-oplossing is de oprichting van gespecialiseerde software fabrieken, voor een aantal industrieën uit een componentenmarkt met vele leveranciers, waarbij de fabrieksspecialisatie bereikt wordt door processmodellering en actieve processondersteuning. De fabriek onderskunt alle typen gebruiker, d.w.z. management, technisch, administratief.

Vincent West *en* Edward Babb
Strategic Systems, ICL Bradmell, UK
Een Spreadsheet met Zichtbare Logica, p. 319.

Dit artikel beschrijft een prototype van een logische spreadsheet dat ontwikkeld werd als onderdeel van VisiLog, een EUREKA project waarbij ICL en BULL betrokken zijn. De bedoeling van VisiLog (Visible Logic) is het volledige vermogen van logica ter beschikking te stellen aan de eindgebruiker d.m.v. gemakkelijk te gebruiken graphics.

Het artikel bespreekt de mogelijkheden van het spreadsheet en illustreert de voordelen hiervan t.o.v. een conventionele spreadsheet d.m.v. voorbeelden zoals gegeneraliseerde antwoorden, doelzoeken, hanteren van alternatieven en ongelijkheden en (logische) programmeerbaarheid. Tevens beschrijft het de interne architectuur, speciaal de gebruikte 'logical engine' PLL (Pure Logic Language), die het resultaat was van een ALVEY project waarbij ICL, Imperial College en Edinburgh University bij betrokken waren.

Mick Smith
ICL Strategic Systems
Colin Tattersall
University of Leeds, Computer based learning unit
De voorziening van intelligente hulp en de uitwerking hiervan op de ontwikkeling van toepassingen, p. 328

Intelligente hulp kan in twee belangrijke opzichten van conventionele hulp onderscheiden worden. Ten eerste, hulp wordt gegenereerd door technieken die gebaseerd zijn op empirische en wetenschappelijke studies van de aard van en vereisten voor hulp. Ten tweede, en tevens vanwege het feit dat het gegenerend is, kan het dynamisch aangepast worden aan de kennis van de gebruiker, de onderhanden zijnde taak, en de toestand van de toepassing. Dit in tegenstelling tot conventionele hulp, waar de nadruk lag op toegangsmechanismen tot eerder opgeslagen teksten.

De benadering tot het voorzien in intelligent hulp die door het EUROHELP project ontwikkeld werd, vereist vier belangrijke onderdelen: een Intelligent Help System Shell (IHSS) dat het generische run-tijd systeem vormt, een Toepassing Modellering formalisme om een toepassing aan het IHSS voor te leggen, een Help System Development System (HSDS) ter begeleiding van de constructie van Toepassingsmodellen en tot slot een Help System Development Methodology (HSDM) ter sturing van het gehele proces.

Deze benadering vereist bepaalde interfaces met de toepassing zelf, zodat de IHSS toegang tot contextuële en gebruiker informatie kan krijgen. Daarnaast, en voor effectieve exploitatie, brengt dit bepaalde veranderingen met zich mee in toepassings ontwikkelingsmethoden, outillering, ontwerp en implementatie.

Dit artikel beschrijft de lessen die van het EUROHELP project geleerd zijn m.b.t. de aard van intelligente hulp, de potentiële voordelen die van zulke hulp verwacht kunnen worden, en practische technieken voor het verlenen hiervan. Globaal bespreekt het artikel de uitwerking die het verlenen van intelligent hulp heeft op topassingsontwikkeling.

Darren van Laar *en* Richard Flavell
The Management School, Imperial College, London
Gebruik van Kleur op beeldschermen. II. Codering, herkenning en toepassingbereik, p. 362.

Kleur wordt dikwijls op Beeldschermen gebruikt om de interactive tussen mens en computer te verbeteren en een natuurlijker en representatiever beeld te krijgen dan dat met monochroomcodering alleen bereikt kan worden. Teneinde effectief gebruik te maken van kleur, is het van belang dat men begrijpt hoe kleur waargenomen en er op gereageerd wordt, hoe kleurcodering gebruikt kan worden als hulp bij bepaalde taken, hoeveel kleuren in een beeld gebruikt kunnen werden, en wanneer kleur nuttig kan zijn.

Dit is het tweede van twee artikelen waarin het effectief gebruik van kleuren op beeldschermen besproken word. Het behandelt speciale factoren van kleuren beeldschermen, verklaart wat met kleurcodering bedoelt wordt, wanneer de verschillende vormen van kleurcodering en gebruikt worden, en het effect van het gebruik van kleuren op prestaties en voorkeur. Ook de op het gebruik van kleur in beeld-schermen betrekking hebbende esthetische en omgeringsfactoren worden besproken.

Het artikel besluit met een samenvatting van de fysieke en cognitieve factoren die door deze twee artikelen naar voren gebracht werden en tevens enige raadgevingen over hoe dit advies in praktijk gebracht kan worden. Ook bevat het een verklarende woordenlijst van de meestgebruikte termen.

Richard Epworth
Technology Strategy Manager, STC Technology Ltd., Harlow, Essex CM17 9NA, UK
Oogbewegingen voor Een Bidirectioneel Mens-interface, p. 384.

Wanneer wij naar een beeldscherm kijken, zijn wij ons er niet van bewust waar onze ogen precies naar kijken; toch bevatten onze oogbewegingen waardevolle informatie. Een opto-electronisch systeem dat onze oogbewegingen bewaakt, kan gebruikt worden om met een machine te communiceren. Gegevens op een beeldscherm kunnen geselecteerd worden eenvoudig door er naar te kijken, en bewerkingen kunnen gestart worden door bewuste oogbewegingen. Ook kan zo'n systeem weten waarin wij belangstellen of waarmee wij moeilijkheden hebben, en op onze behoeften reageren zelfs als wij ons die zelf niet bewust zijn. Dit artikel beschrijft de technieken voor het meten van oogbewegingen, mogelijke toepassingen van deze informatie en verschillende praktische demonstraties van een door de ogen bestuurde interface.

Peter Wiles
ICL Central Government & Defence Business Unit, Slough, Berks, UK
In leiding tot het GIN programma (Government-IT-Infrastructure for the nineties), p. 412.

Dit artikel beschrijft de redenering welke leidde tot het opzetten van een belangrijk ICL projet, het GIN Programma. GIN betekent Government Infrastructure for the Nineties. Dit is een programma dat bedoeld is om een geïntegreerde, samenwerkende set UNIX en PC producten te leveren, beheerd via een range Systems Management producten.

Het GIN programma vormt een antwoord op de opvattingen van verschillende b elangrijke klanten m.b.t. hun toekomstige IT strategie, in het bijzonder van dat van sommige van de grootste overheidsinstanties. Hoewel toepassingen bij de overheid niet veel gemeen hebben met die van de industrie in het algemeen, hebben deze toch een set infrastructuur vereisten gemeen, in het bijzonder voor een geïntegreerde product basislijn welke vooruitloopt op de markt in zijn geheel. Dit artikel beschrijft hoe die infrastructuur bepaald werd en hoe ICL van plan is de betreffende producten te ontwikkelen.

THE NEW SX MODELS

Foreword

Having been involved in the development of ICL's Series 39 range throughout its entire history, it was my particular pleasure to be asked to introduce these three papers on SX Systems. These articles cover the development of the SX Systems node – which was designed by a core team of 120 increasing to 200 at the peak of its activity. However this number would be at least doubled if one were to take into account those who designed the accompanying Macrolan, VME developments, Node Support Computer, Macrolan/Oslan Gateway, and the Design Services infrastructure. The commitment of all these people was required to see the project through to completion.

I have been fortunate enough to lead the development of Series 39 SX Systems from their inception in 1986 right through to the present day. From the very beginning I was aware that the technological advances incorporated in this new product could return ICL to the forefront of mainframe design. In fact the resulting processor has now been announced as the most powerful uniprocessor in the world. The unique architecture of this processor forms the subject of the first paper, 'The SX Node Architecture'.

Behind this architecture lies a lengthy design process. As Director, Mainframe Systems it was my responsibility to provide the team of designers who worked on the Essex project with the best silicon technology possible. Accordingly, in 1985, we extended out collaboration with Fujitsu to include the Hawk technology which gave the ICL designers access to the most advanced chip technology in the world. However, we were aware from the start that our competitors would also have access to the same basic technology. Accordingly, it was up to our own designers to ensure that their processor and associated hardware designs exploited the new technology to its greatest advantage, and thereby gained true superiority for the SX Systems range. The second article – 'SX Design Processes' – describes how design objectives were set and met through the most advanced methodologies.

Their success can be measured by a comparison between SX Systems and a mainframe using the same Fujitsu technology manufactured by one of our competitors: Amdahl's 5990. The SX 580–20 is less than a sixth of the 5990/1100's weight. It takes up less than a quarter of the floor space. It uses less than half the power, and it puts out less than half the heat. And yet it gives over 50% more power per processing node. The story of how these particular advantages were gained forms the basis of the final article: 'Essex Packaging Concepts'.

While the lower end of the computer market has become increasingly commodity led, the superiority of Series 39 SX Systems is a clear demonstration that design excellence can produce true differentiation for a mainframe range. It is this last point which I hope that you will bear in mind as you read the following articles. ICL has long had the reputation for breaking new ground. By its pioneering adoption of such advances as optical fibre cabling, and its decision to stand out against older thinking by developing an operating system better suited to the information needs of the current marketplace, ICL has always endeavoured to create products which pass on unique benefits to our customers.

T.A. Hinchliffe
Director, Computer Products Division

The SX Node Architecture

J.R. Eaton, G. Allt and K. Hughes
Corporate Servers Product Group, ICL, West Gorton, Manchester, UK

Abstract

SX is the latest generation of high performance Series 39 processors. It is compatible with existing Series 39 systems but offers much higher performance both in single and multi-node configurations. SX also offers increased I/O connectivity and performance to balance the increase in processing power. This paper describes some of the more important features of the node.

1 Introduction

SX is the latest member of the successful Series 39 systems. It is introduced to provide increased levels of performance for users of the ICL VME operating system. As a member of an existing family it must be completely backwards compatible; all current user programs will be capable of running on SX without the need for recompilation.

Many design and technology innovations have been combined to give the SX the increase in performance it has achieved. In many respects this has only been possible because both the technology and the design capability came together at the right time.

SX is built from large ECL gate-arrays manufactured by Fujitsu. These are significantly larger and faster than those used in the Series 39 level 80. However the technology alone could not have delivered the required performance increase over the Series 39 L80. SX has relied heavily on design innovation which has lead to the development of a very sophisticated pipelined processor – possibly the ultimate pipeline for the 2900 order code. The result is a design which uses three times as much logic as the Series 39 L80, has a clock beat less than half that of the Series 39 L80 *but* has increased the performance by a factor of up to four. We have obtained a performance boost of two by technology and two by design.

In the early design of SX it was recognised that the main business of the users of VME based systems was to be Online Teleprocessing – OLTP. It was decided that this was one area where the node would have to be optimised but not at the expense of other workloads. A great deal of design effort –

particularly in the Order Code Processor (OCP) slave has been done to achieve this.

The Series 39 L80 was provided with optical fibre Input/Output, Macrolan 50, for high speed peripherals and OSLAN (Ethernet) for slow speed peripherals. For SX, in order not to compromise the design of the high speed I/O system, it was decided not to have slow connections directly to the node but to provide OSLAN connection via the Macrolan to OSLAN Gateway – MOG.

As part of the Series 39 family, SX must provide a multi-node capability. A huge amount of effort has been invested to provide a highly effective inter-node service which is technically capable of supporting a multi-node system with up to 8 nodes.

1.1 Existing System Behaviour

Extensive investigation into existing Series 39 L80 was carried out in order to understand the behaviour of the current systems. A processor was extensively modified in order to capture large amounts of trace data relating to instruction sequences and data addresses. Many monitoring runs were carried out using different workloads, from teleprocessing to scientific batch. This information gave the designers an understanding of the current use of the Series 39 systems. Together with awareness of the expected trends, this information allowed the optimisation of the SX design for the workloads which were actually being and going to be run. The trace data was used during the design and development process to predict performance for a number of workloads and used for exercising high level models in order to identify bottlenecks in proposed designs.

1.2 Outline Node Description

The SX node consists of four units

Order Code Processor	OCP
Input Output Processor	IOP
Inter-Node Processor	INP
Store	

Although these units are logically separate, much of the functionality of the INP and IOP is provided by the OCP.

The basic information flow – data and control can be represented by the following diagram:-

2 The SX Order Code Processor

The SX OCP is a heavily pipelined design. At the highest level it consists of four asynchronous units.

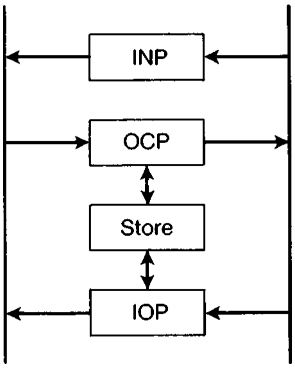


Fig. 1 Conceptual data and control flow in the SX node

Scheduler
Upper pipe – address generation
Data slave
Lower pipe

The Upper Pipe, Data Slave and Lower Pipe are often considered as a single unit – the engine.

2900 order code is fetched by the scheduler and then decoded to give a sequence of Picode which the engine then obeys. Picode is the implementation order code of the SX OCP – it takes the place of microcode on previous machines but is at a higher level, much closer to the 2900 order code. Picode is executed by the engine.

Each unit is internally pipelined with typically four stages. The internal stages of different units are effectively overlapped. This allows an output

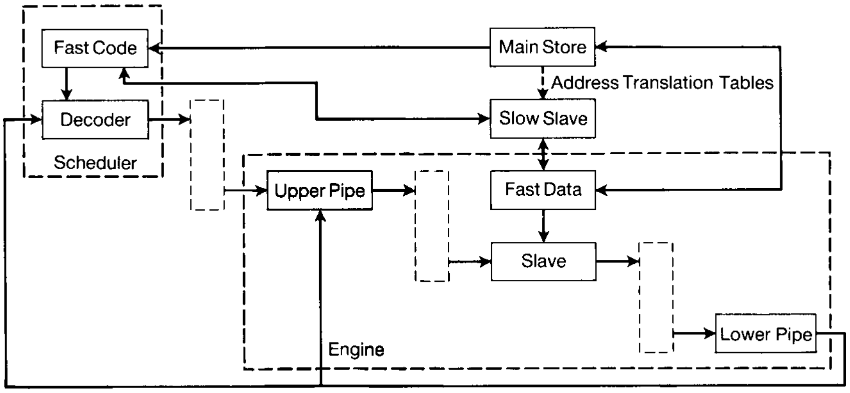


Fig. 2 The SX pipeline

from a unit to be generated before that unit reaches its last internal stage. In the engine there are a total of 13 stages but after accounting for stage overlap only 10 are visible due to the overlap between stages. In the total pipeline there are 17 stages with 14 visible.

Each of the four units operates asynchronously. There is a queue of up to 16 instructions between stages – the parameter files. This allows the pipeline utilisation to be maximised. Instructions are executed in strict chronological order – this is a rule imposed by the Series 39 architecture – and completed at a maximum rate of one per beat. Although completed in strict chronological order the slave will handle instructions in any valid order. This allows store accesses to overtake one another subject to strict rules and hence allows the possibility for two read instructions to be completed in reverse chronological order provided that the destination registers were different and that no conflicting writes appear in between. The lower pipe will ensure that the final terminations are in the chronological order.

The pipeline handles two separate streams, the A and the B stream. The A stream is used to execute 2900 order code, the B stream to provide system level support. There is communication between the two streams in order that certain activities may be co-ordinated. This is handled by two First In First Out buffers, an A to B and a B to A. These cause events to be generated on the target stream. The use of two streams has the effect of maximising the pipeline utilisation by absorbing the slack caused by store latency etc.

All registers are held and maintained by the lower pipe, for both streams – each stream has its own copies of the register set. Copies of certain registers are held in the upper pipe and the scheduler in order to improve performance by reducing the latency of access. The upper pipe holds copies of all registers which can be used in address generation and attempts to maintain them, but in certain circumstances it cannot. A prime example of this is when the register is loaded from store. In these cases the register will be updated from lower down the pipe. In general this update will come from the lower pipe; however in certain circumstances the slave will provide the data directly. This has the advantage of significantly reducing the latency for certain updates. This leads directly to a performance improvement for the OCP.

Like any other pipeline the SX pipeline needs to be reset after certain events such as an incorrect jump prediction. In this case the current instructions being executed by the pipe must be abandoned. This is handled by resetting the pipe and then updating any slaved copy of a register with the current master value from the lower pipe. This is an expensive event and the frequency must be minimised in order to maintain the performance of the pipe. This was achieved by developing a very effective jump prediction mechanism.

Interlocks between units are handled using a scoreboard-like technique. This is intentionally "imprecise" in that instructions in any unit will start regardless of the state of the scoreboard. At the end the scoreboard is used to determine if that instruction has been completed successfully. This allows many of the bypasses between stages – where data is fed back up the pipe – to be used. The effect is a net gain to the performance of the pipe.

2.1 Scheduler

The scheduler is responsible for fetching the 2900 instructions (PLI = Primitive Level Interface) and for translating them into sequences of Picode which will be executed by the engine.

PLI fetches are satisfied either by the fast code slave, the slow slave or by the store. Picode fetches are satisfied by the Picode RAMS (a special case) and also by the fast slave, slow slave or main store.

A sophisticated jump prediction mechanism is implemented for the 2900 order code. It is essential in order to maximise the performance of the pipe. Without this feature, the number of jumps in normal instruction streams would disrupt the pipeline and reduce the performance dramatically. Various jump prediction mechanisms have been investigated in the past giving various results depending on the workload. The method chosen for 2900 order code on SX is to access an indicator, using a hash of the jump instruction address, which predicts if the jump will be taken. The indicator is set based on the behaviour of the jump instruction in the past. A simple by 1K by 1-bit RAM is used to hold the jump prediction table. This gives a jump prediction accuracy of over 85% on typical workloads.

Picode has a different jump prediction mechanism. All Picode jumps contain an indicator as to whether the jump is likely to be taken or not. In the design of the Picode it was easy for the programmer to determine which way the jump was likely to go, setting the indicator accordingly. This has resulted in very high prediction accuracy for Picode – in excess of 95%.

2.2 Upper pipe

The upper pipe is responsible for address generation and for the execution of simple instructions.

Address generation for Picode, which maps very closely to 2900 PLI, is a relatively simple task BUT is complicated by the architectural rules associated with 2900. The upper pipe handles all these cases. It will check stack addresses to ensure that they do not fall outside the stack frame and will check that code area addresses are within the current code area.

Some simple instructions are executed directly by the upper pipe. These are restricted to simple register/literal operations or register/register operations. This has a significant effect on the performance of the pipeline by reducing the effect of pipeline stalls introduced by register dependencies. If the unit did not execute the instructions directly, the latency of a register update would be the complete length of the engine pipe; this would clearly be unacceptable.

2.3 Slave

The slave is essential to the OCP design as it provides fast access for both code and for data. A multi-level slaving system is provided with fast code and data slaves supported by a slow slave and finally the main store. I/O is always direct to main store. A logical view of the slaving system is shown in Fig. 3.

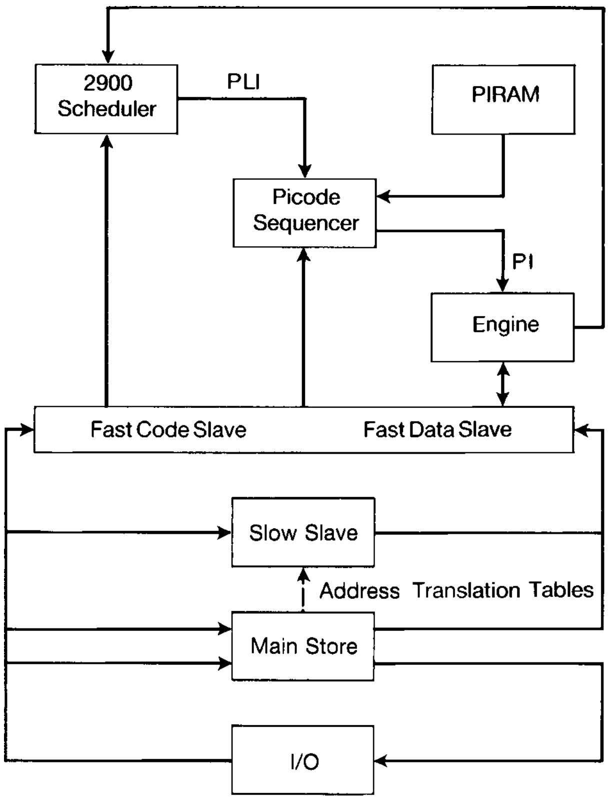


Fig. 3 The SX Slave Hierarchy

The slave handles access to all operands, those in virtual store, real store, image store (control areas) and working store. Even operands which do not require any slave access, such as literals, are passed through the slave. Image store and working store are special areas within the fast data slave. Image store is used for machine control; typically it is used for communication between the independent units of the machine such as the I/O system.

Working store is special store dedicated to the Picode and will typically hold architectural and process control information.

2.3.1 Slave hierarchy The slave has a number of specific slaves used for different purposes which together can be viewed as a hierarchy.

fast code slave	128 lines of 32 bytes	– fully associative
fast data slave	32 lines of 32 bytes	– fully associative
slow slave	64Kbytes, 32 byte cells, 4 way semi-associative	

The fast code slave exists within the scheduler. It is used to hold code for both 2900 PLI and for Picode. It is directly in the internal pipe of the scheduler and as such costs one pipeline step to access.

The fast data slave exists within the engine. It holds all data read from the store. Associated with each line of the fast slave is a set of byte-valid markers. These indicate the validity of the bytes within the cells and are used to eliminate the need for store access on cell creation. Access to the fast slave is in the engine pipeline and as such costs 1 beat to access, which is part of the pipeline delay. The fast data slave is a delayed write-through slave. Only on an explicit prompt or when the cell needs to be re-used is the cell written to the slow slave and/or store. This has the effect of greatly reducing the store bandwidth required for write access; the fast data slave behaves as 32 store buffers.

For the fast code and the fast data slave when a cell is to be re-used the current data is unloaded into the slow slave. This is the only route by which data can get into the slow slave (except for address translation tables – see below).

The slow slave supports both the fast code and data slaves. It is split into four parts of 16Kbytes each, one holding code, one holding data and two holding address translation tables. The slow slave is probably mis-named, the penalty for access to the slow slave being only 2 beats, slower than the fast slave but as fast as (most) slaves in previous machines.

Slaving the address translation tables is an innovation on SX. Previous machines have never slaved address translation tables, only "Current Page Registers" (CPRs) which are manufactured from the address translation tables. Slaving the address translation tables provides fast access to the translation data which results in fast creation of the address translation slave entries. This is important as it allows the number of address translation slaves (CPRs) to be relatively small.

2.3.3 Slave support of address translation A number of Current Page Registers (CPRs) are held for both code and data. These map the virtual address to the real address and provide information on the protection levels of the page. In addition separate CPRs (global CPRs) are used to map virtual addresses which are indirect virtual addresses which map to a global virtual address. The CPRs are fully associative with a least recently used (LRU) replacement algorithm. The CPRs are allocated as follows:

Code	6 Global
	6 direct
Data	6 Global
	12 direct

The address translation process relies on both the global and direct CPRs. A virtual address is first translated using the global CPR. If the address is indirect then the global CPR recognises it and replaces it with the global virtual address. The address is then presented to the direct CPR which will generate the real address and all the protection information required. Both the global and direct CPRs are directly accessible in the slave pipeline. In this way the cost of global translation is eliminated. On previous machines the cost of indirect address translation has been very high, up to 50% greater than direct address translation. Global areas are used to share data between virtual machines which is a facility heavily used in OLTP systems. The substantial improvement in the cost of address translation has the effect of significantly improving the performance of TP systems without impacting other workloads.

2.3.3 Slave Control Interface On SX the operating system has a procedural interface to the OCP which is used to manipulate the address translation tables. The information passed is used by the OCP to determine what action must be taken in order to maintain the slave. This is another example of the tendency to offer higher level interfaces to the operating system in order to insulate it from changes in the underlying hardware. The effect on SX is to minimise the cost of slave clearances leading to increased slave retention. This has a direct influence on the machine performance.

2.3.4 Data alignment The slave is responsible for operand alignment for both read and write data. When store data is read – whether it is supplied by the fast slave, slow slave or main store it is aligned before being presented to the lower pipe. Similarly when data is being written it will be byte-aligned before being written into the slave cells. This gives the machine access from 0 to 8 bytes on any boundary, including page straddles. It is the first machine developed by ICL which is capable of doing this and makes SX the first truly byte oriented VME machine. Byte oriented access simplifies the Picode in that it never has to handle complex cases explicity such as when a six-byte field is contained in three words. It also simplifies the validation process for the Picode as the special cases do not have to be tested in every sequence.

2.3.5 Execution sequence The slave is not constrained to operating on its input request stream in a chronological order. The slave will action VALID requests in the "oldest first" order. This has the effect of allowing the slave to do operations which are not in sequence. The slave continues to fetch data and often will fetch data ahead of when the lower pipe will require it. This has the effect of improving the overall pipe performance.

2.3.6 RAM protection Hamming correction is provided on all data areas for improved reliability and integrity. This is an improvement over previous designs where Hamming correction was provided on the main store but only parity error detection on other data stores.

2.4 Lower pipe

The lower pipe includes the main execute mill. It supports functions for all the Series 39 data formats: integer, decimal, floating point and logical.

The lower pipe provides complex functions such as floating point multiply and divide. These are micro-sequenced using a soft control program held in very fast RAM – the execute mill microcode. The lower pipe operates on a 64 bit data flow. This improves performance as 2900 uses 32-, 64- and 128-bit operations although 128 is infrequent.

In addition to the mill function the lower pipe updates the registers from a mill operation. This will result in copies of that register held in the earlier stages on the pipe (typically the upper pipe) being updated.

Only instructions which successfully terminate can result in register updates. Instructions which terminate unsuccessfully – Virtual Store Interrupt, Program Error etc. cause an exception which results in the appropriate exception handling code being run.

The sequence of execution of the lower pipe is strictly chronological. Although the proceeding stage (the slave) executes instructions in any valid order, the instructions must be terminated chronologically. If this was not the case, restart after a virtual store interrupt etc. would be extremely difficult!

3 Picode

Picode is the native order code of the SX machine. It is fetched by the scheduler and processed by the upper pipe/slave/lower pipe. Picode has been designed explicitly for the SX machine which is a 2900 execution machine. As such many of the 2900 concepts, such as descriptors, are built in.

Picode is unlike the previous low level order codes on Series 39 machines in that it is not a microcode. Picode is at a much higher level than microcode but slightly lower than the 2900 order code. Unlike microcode there are no timing rules associated with Picode. Each instruction executed is independent: interactions are only through the operands addressed by the intstruction. This is enforced by the hardware. Picode execution is atomic in that it is either completed or not, it cannot be interrupted part way through.

Picode has access to a larger register set than the 2900 order code. Nominally the 2900 register set is duplicated BUT the individual 32-bit components of larger registers are accessible. Each instruction operates on two operands, one of which can be in store. Both are used as source operands and one is used as the destination.

Picode is held in virtual store and is accessed through slaves in the scheduler. Three levels of slave exist for Picode, a very fast Picode RAM, the fast code slave and the slow slave. Separate Picode stores are used for the A and B streams with independent Picode RAMs for each. Each Picode RAM holds 1K instructions which correspond to the first 1K instructions of the Picode in store. The Picode RAM holds performance critical code – code which if held in the slave would be "thrashed out" by other less critical code.

PLI Execution is handled by translating the PLI into a Picode Sequence which is conceptually fetched from store and fed down the pipe. Since Picode is slaved the cost of fetching it from store is virtually eliminated. The basic Series 39 PLI is implemented in a very small amount of Picode – about 1K, the complete Picode for all PLI is less than 4K. This is achieved by providing a mechanism in the Scheduler which maps the Series 39 PLI onto a Picode template. Functions, registers and operands are replaced by changing the template into the real Picode instruction. The majority of executed PLI will map directly onto 1 Picode.

The Picode and hardware combine to optimise for the normal case. The normal case is executed as fast as possible. In order to support all other cases a special facility is provided called "warp". This causes the abnormal case to generate an exception event which causes entry into a special Picode sequence which will then perform the required action. "Warps" are vectored in that a specific sequence is provided for each. The "warp" behaves like an escape sequence in that at the end the originating sequence is returned to. In many cases the "warp" is generated by the slave as a result of an operand access. The operand access is converted to an access to the special working store where a picode address is found. This is then used to force a jump to the appropriate sequence.

Optimising for the normal case is also applied to the detection of program errors. In order to improve performance of the PLI implementation certain architectural checks which could result in a program error are handled imprecisely. Effectively the check is applied BUT the resolution is only to the instruction. A special Picode module – Resolution of Program Errors (ROPE) – is entered which emulates the instruction in order to determine the error precisely.

In addition to PLI support, Picode is used for all system level functions, which include

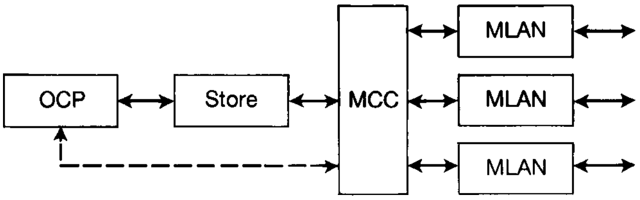
Multi-node support
IOP support
I/O protocol support (VME sub-system)
Node support (diagnostics etc.)
Etc.

This accounts for the majority of the Picode. In excess of 40K instructions are used to support the system level functionality. Such a large amount of storage would have been prohibitively expensive and could not have been implemented within the constraints imposed by the technology. Holding the Picode in main store and relying on very good multi-level slaving has allowed the use of large amounts of control function to be implemented in Picode without significantly degrading system performance.

4 Input/Output system

The input/output system on SX provides up to 16 Macrolan channels. These are used for all peripheral access to both fast and slow devices.

The I/O hardware is made up of a single Macrolan Channel Concentrator and up to 16 macrolan daughter boards.



MCC=Macrolan Channel Concentrator

Fig. 4 The SX I/O Interfaces

High speed I/O is driven using the same VME interface as was used on previous Series 39 machines – this has removed the need to change VME for all disc and tape driving.

For low speed devices the OSLAN protocol is handled over the Macrolan. A new subsystem has been introduced to VME – Remote Coupler Handling – to support this. This packages up the OSLAN data and passes it over the Macrolan to the Macrolan to Oslan Gateway (MOG) which then drives the OSLAN.

As on previous Series 39 machines the OCP supports the VME operating system by providing high level functionality for I/O driving. Logically the I/O driving software exists as part of VME; however the performance critical paths of the I/O interface have been implemented in Picode and are called directly (and indirectly on interrupts) from VME. On SX this code has been extended to support the Remote Coupler Handling software.

The OCP supports the I/O by use of special stream B Picode. This is assisted by the use of special hardware entry points which direct the Picode to the appropriate sequence.

5 Multi-node

Multi-node systems are an important part of Series 39 strategy and hence essential for the SX system.

Series 39 multi-node systems are built around a number of nodes running a single coherent copy of the VME operating system. These nodes may be physically separated by up to 2 kilometres. Nodes may share virtual store by replicating that store on the different nodes. The replicated virtual store is maintained by the inter-node service without the intervention (in normal operation) of the operating system.

Experience of past machines has indicated that the performance of the inter-node service is highly dependent on the amount and type of inter-node traffic. In order to ensure that the SX inter-node service could handle all levels of workload, extensive simulation, using trace data from the L80, was carried out.

All previous implementations of the inter-node service have been significantly different, resulting in different VME sub-systems for the control of each. SX has introduced a new high level procedural interface for inter-node service control allowing VME to be insulated from the implementation of the inter-node service. It is now possible to replace completely the inter-node service below the VME interface without changing VME. This is a direction more of the VME interfaces will take in the future giving greater flexibility to the hardware designers while reducing the demands on VME to "box follow".

Due to the high performance of the individual nodes in an SX multi-node system, a high performance inter-node service is required. This is because an increase in node performance increases the amount of multi-node traffic. Handling this becomes a factor limiting the overall system performance. Nodes may be separated by up to 2 kilometres but this has an impact on the multi-node performance due to the message latency, primarily for synchronisation events.

The SX inter-node service is based on multiple glass fibre links – as have all previous services. Four optical fibres may be used as data links and one for chronology. For resilience and system partitioning purposes a second token ring is provided.

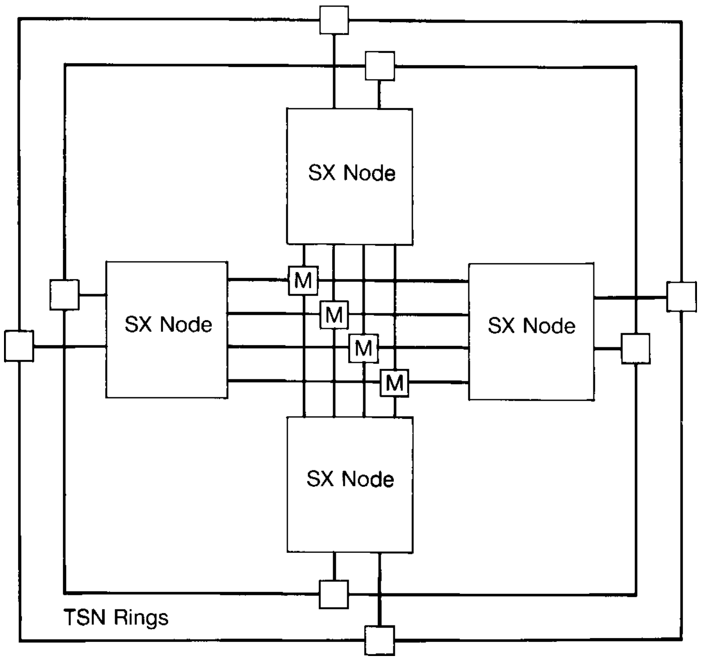


Fig. 5 The SX Multi-node Interface

-□- This is a Macrolan Port Switch Unit (MPSU) which connects the Lan into a 'ring structure'.

All Macrolans use the same optical fibre technology-Macrolan 200.

Note: only one TSN ring is active in a four node system. When the nodes are partitioned one TSN ring is allocated to each partition together with a number of data LANs.

It is a requirement of the Series 39 multi-node system that a coherent store image is maintained. On the current multi-node systems, store can be replicated on different nodes, and this replicated store image must also be maintained coherent. This is achieved by applying a strict chronology to all writes to this store. On SX the chronology is provided by means of a Token Serial Number – TSN – which is carried by the single token on the TSN ring.

A node wishing to transmit a public message waits until the token arrives on the token network. In addition to a serial number the token provides information on which of the data LANs are currently in use. Providing a LAN is free the node will acquire dedicated access to a data LAN and indicate this in the token. The token is not stopped; it is changed as it passes through the node. The node now transmits the message on the data LAN has acquired. Nodes which receive the message all "acknowledge" the message.

Once all nodes have acknowledged the message – a protocol within the data LAN – the message is deemed complete. The OCP is then informed. The sending node clears the information indicating the LAN is in use when the token next passes.

By separating the implementation of chronology and of data transmission, the latency for messages is much reduced. The circulation time of the token is fixed; it is independent of the traffic on the data LANs. This reduces the latency for semaphore operations which are *critical* to the performance of the system.

SX has introduced blocking to the transmission of data on the inter-node network. All outstanding messages are transmitted when a data LAN is acquired as a single "block". In a similar way multi-node traffic as a result of moves to public areas is "blocked". A string order which writes to an area which would result in inter-node traffic has the writes buffered. Only when this buffer is full or when the string order completes is the data actually transmitted. This significantly reduces the data LAN utilisation with corresponding improvements in performance.

As indicated above all replicated store must be maintained coherent across all nodes. It is possible that a write from one node clashes with a write from another – the same store area is being written concurrently from two processes. To avoid this resulting in different values in the stores a clash map is implemented in each node. If a write from another node addresses the same store area before a write from the current node is completed, ie before the OCP is informed that all nodes have acknowledged the message – then a clash has occurred. If a clash has occurred then the nodes own writes are re-applied as they return. The clash map – implemented in the slave – detects the condition and inhibits all reads from this area, otherwise the OCP would have to be stopped until its outstanding writes were completed. This has the effect of maintaining performance as the resolution on clashing addresses is so high that clashes rarely occur.

6 Conclusion

SX is the most complex and powerful machine ever built by ICL. It has combined technology provided by Fujitsu with design skill and experience from ICL. SX has been a challenge to the designers and implementors. Many new and complex features have been introduced into the node, features which will be exploited in future designs.

The result is a high performance, highly reliable system capable of satisfying a large range of customers' computing needs.

References

1 BROUGHTON, P.: Input/output controller and local area networks of the ICL Series 39 Level 30 System, 1985 ICL Tech. J *4* (3), 260–269.
2 WARBOYS, B.C.: VME nodal architecture: a model for the realisation of a distributed system concept, 1985, ICL Tech. J. *4* (3), 236–247.

European (EP) and British (GB) Patent applications relating to SX

EP 262 782
Multi-node data processing system
D. Bell

EP 352 935
Data processing apparatus
J.R. Eaton, P.V. Rose

EP 357 188
Pipelined processor
J.R. Eaton, C.M. Duxbury

GB 2 215 887
Data memory system
J.R. Eaton, R.M. Lister, K. Turley

EP 320 098
Jump prediction
S. Hodges

EP 372 751
Pipelined data processing apparatus
C.M. Duxbury, P.V. Rose

GB 2 227 584
Data processing system
D. Bell, C. Lomas

SX Design Processes

G.P. Abraham, D.C. Freeth and H. Vosper
Future Products Development, Corporate Servers Product Group, ICL Computer Products Division, West Gorton, Manchester, UK

Abstract

This paper gives an overview of the design processes followed during the design of the SX system.

It covers some of the major initiatives and innovations and specifically highlights the extensive methods employed in design verification.

Tools described include performance models for OCP, I/O and inter-node, software emulation techniques, hardware simulation and test software.

The paper also contains a description of the design control and measurement methods.

1 Introduction

The continued improvements in silicon technology and the increases in complexity have necessitated that the design of the SX system uses design methodologies which are innovative, rigorous and give orders of magnitude improvement in the quality of the design compared with previous processor designs.

The methodologies and associated development routes described here cover the design of the hardware, the associated microcode known as PICODE and the node support computer (NSX).

There are many examples within these areas where a high level of innovation can be shown and this paper focusses on some which although particular to SX may be of interest and applicability to many areas of system design.

All of the initiatives are a direct result of analysing the problems encountered on previous projects and looking at areas for improvement whilst at the same time taking into account the advances made in the hardware and software tools that are now available.

To set the context for the following sections, reference to a generic development route is helpful – see Fig. 1. This 'V' diagram illustrates the stages of design from system down to implementation on the left hand side and stages of test from unit up to system on the right.

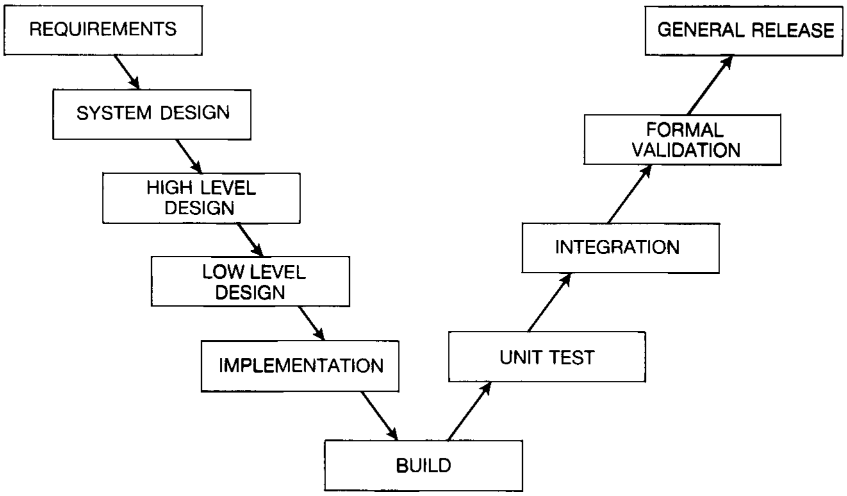


Fig. 1 Generic development route

The aspects of this development route which will be covered are the design and development activities, i.e. from Requirements through to Implementation. Particular emphasis has been placed on the System Verification, Simulation and Evaluation and Test Bed activities which have demonstrated considerable success in the System, Picode and Hardware Design processes.

2 Requirements/system design

Fig. 2 shows an outline of the overall design and validation process in terms of the design data decomposition and the verification method for each level. For example, the logic description is validated by the use of the MSIM logic simulator.

This section will describe the requirement setting process and two of the most innovative and valuable verification techniques used at the System Design stage. These are:-

HAM – High Level Architecture Model.

and

SIMULA – A model of I/O and multi node capability.

The other design validation methods will be described in Sections 3, 4 and 5.

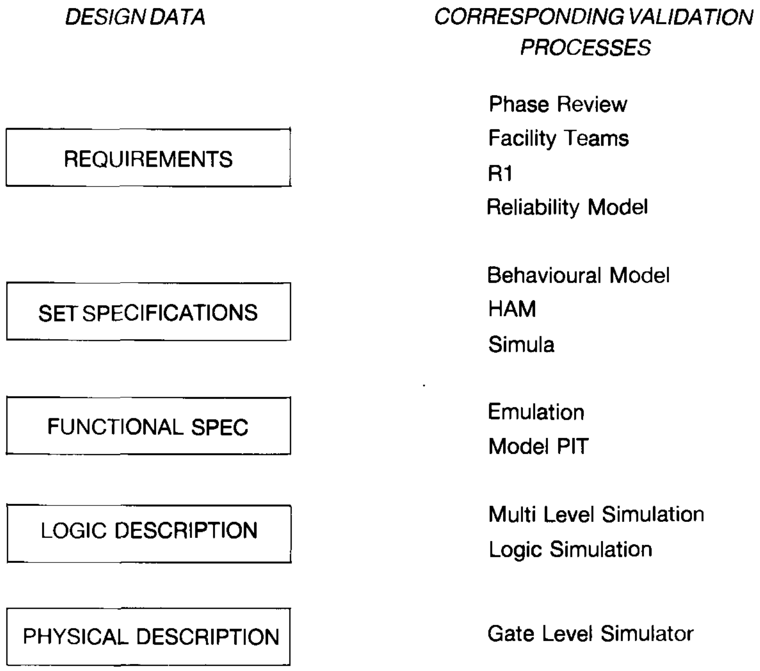


Fig. 2 Overall design and validation process

2.1 Requirements

A top level Statement of Marketing Requirements is formed reflecting the customer requirements. This is mapped to a System Definition taking into account technology information and the Business Case. Having established an issued System Definition containing targets for the various system attributes, the mapping and decomposition of these requirements is an important process.

In the case of SX, system attributes were managed by setting up orthogonal review mechanisms which were led by Systems Architecture and drew together expert areas of the Division. Thus, attributes such as performance, reliability and maintainability could be monitored and assessed. Component and system reliability models were constructed and updated with lower level design data at appropriate times during the development life cycle.

Performance models will be discussed in more detail later.

Facility teams were also created within the Development teams and a three level Requirements mapping and review mechanism established. These multi disciplinary teams each investigated specific areas of the design and agreed the definitive set of requirements for that area. These included, for example, Memory, Order Code Processor, Diagnostic Control and Engineers Facilities. These high level requirements were then reviewed at three stages:-

The definition of *what* facilities were to be implemented (R0), *how* they were to be implemented (R1) and then the *mapping* of these facilities to functional partitions of the design (R2). The completion of this stage allowed detailed design implementation and verification to proceed.

2.2 HAM

The High Level Architectural Model – HAM (patent pending) is a model used to predict the performance of the SX system. It is a behavioural synthesiser – that is to say, it does not execute operations. It builds up a behavioural picture of the node by using information derived directly from a Level 80 which is monitored whilst running standard customer benchmarks. This is combined with architectural functional patterns extracted from the design state of the SX node.

HAM is a model of the pipeline and slaves in an SX node which simulates their behaviour and interaction.

This modelling is achieved by decomposing the 2900 PLI [PLI = Primitive Level Instruction – the range compatible order code for Series 39] into individual pipeline relationships which are used to determine the behaviour of the model. These relationships identify the functional interactions which enable designers to calculate the performance costs of pipeline hold ups.

Virtual addressing information is also supplied to the model which permits slave algorithms to be assessed.

This meant that both "what if" scenarios in the design and changes resulting from practical design implementation could be modelled, giving accurate predictions of system performance.

One of the major advantages of HAM is its speed of operation which is one thousand times faster than conventional simulators. The resultant power advantage enabled realistic benchmarks and workloads to be run.

Significant effort was put into the extraction of real time paths through the running of benchmarks on Series 39 to capture instruction streams and addresses. This ensured that the model was not working on a statistical basis but on real PLI streams. The major benefit of this was that the pipeline operation and slaving were verified and established early in the design life cycle.

The model was used throughout the SX Development to monitor system and workload trends and give early and accurate performance assessments which were fed back to the corporate review process and confirmed that the project was on track to meet customer requirements.

2.3 Simula

The design of the I/O and inter-node subsystems was modelled using a variant of the Simula [a derivative of ALGOL] programming language. The system, Discrete Event Modelling in Simula was derived from a development by G. Birtwistle in 1970 [Ref 1].

Simula is a behavioural model which predicts I/O and Internode performance using standard queueing theory and statistical methods.

During the early design stages, basic requirements must be thoroughly understood. For I/O these requirements are usually specified as throughput rate and for inter-node as the performance degradation seen as a result of adding more processing nodes.

The I/O subsystem was modelled with particular emphasis placed on buffer sizes and processor loading. The bulk of I/O traffic consists of packets of data flowing between the node and the disk subsystems. Analysis was made of standard benchmarks to understand traffic profiles and provide parameters for the Simula models. The model was run in varying I/O configurations and a check made for any non-linear behaviour, for example data buffer overflow on peak traffic loads.

The I/O model was run with various limit conditions to ensure that the design was stable, could meet requirements and proceed to the next design stage.

The inter-node system model was developed to enable performance predictions for various multi node configurations and differing physical separations between nodes. As with the I/O model, behavioural information from various benchmarks was analysed to establish statistical patterns of information, in this case internode traffic.

In a multi node system, one of the prime causes of performance degradation is the loss of processing time when a node needs to resynchronise its local memory with that of other nodes. Synchronisation is achieved in 2900 architecture with PLI semaphore instructions. The distribution of these semaphores, together with node separations, was modelled to predict total system performance.

The model runs on a Series 39 mainframe and requires relatively small amounts of processing power. Graphical input is now available on SUN workstations which makes the Simula system an extremely flexible and useful tool in the System Design environment.

3 PICODE design process

One of the most notable process improvements and successful design implementations in the development of SX was in the area of PICODE design. PICODE, which is the native order code of the SX machine, is a derivative of Series 39 Level 80 microcode.

The delivered quality of the PICODE shows a twenty times improvement over its Level 80 counterpart in terms of faults found after prototype switch on. The two most significant contributors to this improvement were the design and implementation of a disciplined High Level Design process and an innovative module validation process, the Picode Test Bed (PIT).

3.1 High level design process

Fig 3 shows a simplified diagram of the overall PICODE design process. The High Level Design Process takes as its starting point the definition of the primitive level interface and the requirements from the R1 stage. PICODE specification documents are then generated for specific modules and each module is then flowcharted. PICODE source is then written and the source files are then linked into hierarchic structures and directly to their equivalent flow chart boxes using the SX design database. [Section 6].

It should be noted that one of the most fundamental aspects of the process is that design reviews, or desk checks where appropriate, are held after each stage. So for each specification, flow chart or source module generated, a formal review was carried out.

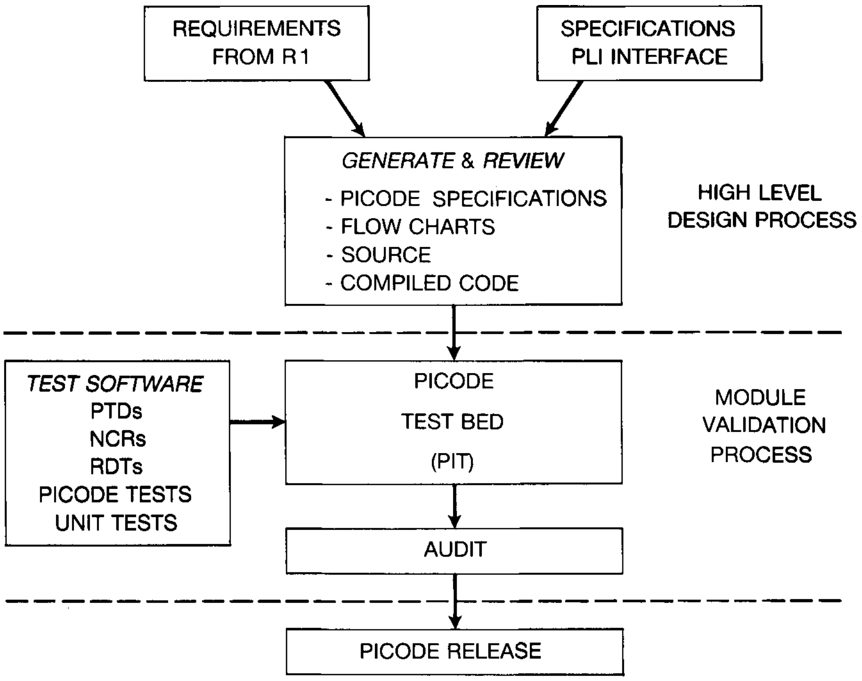


Fig. 3 PICODE development process

This review was fully defined in terms of procedure, input and output criteria and was performed and recorded by an independent member of the team.

Design control was performed via the SX Design Database which provided comprehensive build and control facilities. This precluded erroneous release or progress to next stage, and gave up-to-date and accurate audit trail information.

This highly disciplined approach meant that a very large proportion of faults were removed early.

A fault feedback mechanism was designed into the development process and fault information was centrally logged and analysed against targets for each design stage. A significant achievement was that 66% of all faults were found prior to the Module Validation stage.

3.2 Module validation process

Module validation on PICODE uses a particularly innovative environment – the PICODE Testbed or PIT, which allowed validation of the PICODE totally divorced from any hardware environment.

PIT is an emulator which permits PICODE to be tested, using specific and range defined standard tests. It provides a facility for testing sequences, executing PICODE and tracing execution paths. The definition of the hardware is provided by the hardware teams in the form of PICODE Procedures (PIPROCS) which are captured in the PIT environment. Compiled PICODE is input to PIT and then executed. One further facility is the ability to write test scripts to enable sequences to be initialised and executed in a controlled manner.

Tracing facilities are an integral part of the system.

The test software suite supported by PIT includes – PLI definition tests, Normal Commissioning Routines (NCRs) and Range Defined Tests (RDTs). These latter two test suites are identical to those run on Series 39 and SX prototypes, therefore maximising test cover and ensuring range compatibility.

All outputs from the PIT runs were audited prior to build and release. Design control was maintained by marking the correct status in the SX design database to enable the build stage to commence.

The prime benefit of PIT is that through earlier exposure to range defined test software, the faults are removed which would otherwise have been found on prototype hardware. In addition, early bug free and functionally proven PICODE (and test software) was available to the hardware simulation environment (MSIM) for hardware design verification.

4 Node support processor design process

4.1 High level/low level design

The Node Support Processor for SX has exploited structured design methods based on Ward & Mellor [Ref 2], supported by CASE (Computer Aided Software Engineering) tools. The software package is the design capture product EXCELERATOR/RTS and the supporting hardware a network of ICL professional work stations (DRS M60/M80), with links to dedicated VME service systems.

Adopting this approach, has provided a formalised design method that enforces a common way of expressing a design – a common language. This leads to improved communication within the project, consistency in the design and significant improvement to the review/audit process.

EXCELERATOR/RTS provides automatic checking of the validity of the design and improved ease of maintenance – any changes to one area can immediately be checked for impacts to other areas of the design.

Designs can be expressed through a combination of Context Diagrams, Transformation Graphs, State Transition Diagrams and Entity Relationships, all of which can be checked for consistency with each other at the various levels of definition.

The context diagram
This shows the external interfaces to the system at a high level and is used to ensure there is agreement on the original requirements.

The transformation graph
This considers what processes are needed to handle data and control flows. The flows of data are shown as solid lines and the flow of control as dotted lines. This is the first level of decomposition and will show a number of processes.

The state transition diagram
This allows the same set of processes to be represented in a different way showing how the system moves from one state to another.

The entity relationship
All the information captured in these diagrams is organised into entity types that represent the basic components of the system and many instances of an entity type will exist. Each instance is called an entity, and with large and complex systems it is easy to lose track of how these entities relate to each other. This diagram captures these relationships.

Figure 4 is an example of a context diagram for one module of the application firmware, in this case a top level representation of the application

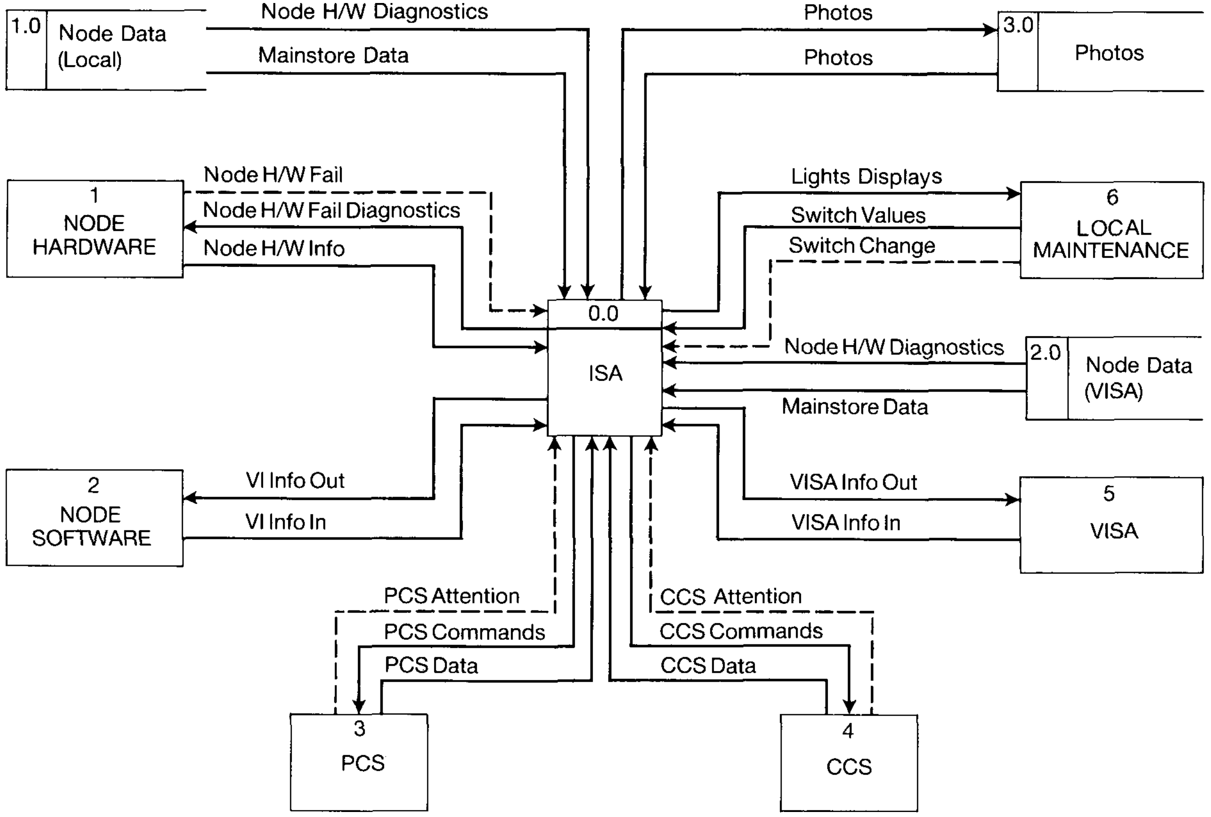


Fig. 4 Context diagram — application firmware (ISA)

firmware. It is typical in terms of showing a mixture of data and control flow and the level of information conveyed by such diagrams.

As part of the assessment of the technique, comparisons were made taking the original 'text only' design information for a particular module of design and the equivalent 'structured' design information. Critical analysis showed two thirds of the text was superfluous 'padding' that was not helping the understanding and the same level of understanding could be communicated in a few simple structured diagrams.

The benefits from the use of structured design methods and the particular implementation strategy adopted went well beyond the expectations of the team and showed quite clearly that it was a more cost effective process. A further benefit was the prevention of errors at the design stage, rather than trying to remove them in the later and far more costly integration and testing phases. This was borne out by the results of measures put in place across the complete development route.

4.2 NSX firmware test bed

The test bed phase of firmware development has proved to be a major area of improvement to the development route. It has allowed extraction of a large proportion of the residual faults post compilation of the code. This is in line with the fault removal targets of 40% in desk checking and 40% in Test Bedding, leaving only 20% to be found on test rigs or prototypes prior to full scale SX system integration. This compares with previous developments where only 25% of the faults were extracted in the Test Bed phase.

The NSX incorporates an INTEL 286 microprocessor that runs PL/M-86 programs. The project already had in place a network of DRS M60 workstations which allowed the PL/M-86 code to be tested in a proven test environment before being exposed to the NSX hardware.

However, it was noted that there had been no method of analysing the extent of test cover provided by the test bed, so the development team searched for software tools that could be used to do this.

A PL/M-86 version of the LDRA TESTBED software package was selected and installed. This provided not just a check of the testing level of the code modules, but also checks on the structure and use of non-standard code. As part of the assessment of the tool, existing Series 39 firmware was analysed and this revealed that the test cover had averaged only 60% – showing considerable room for improvement.

The LDRA TESTBED is a software testing and assessment product with built in quality management facilities for providing report and summary information, and it can run on most operating systems; in our case, VME.

Two types of code analysis can be carried out – Static and Dynamic.

STATIC analysis looks at three areas – *Standards*, *Complexity* and *Structure* and Fig. 5 shows a typical set of results from a run on some modules of the SSP firmware.

Module Name	Version No.	Version Lines	Executable Lines	Procedures	Standards	Complexity	Structured
CCST	1.18	2225	1280	10	4%	5	Yes
COMJOB	1.05	1586	648	38	4%	0	Yes
COMMON	1.66	2989	1227	35	5%	2	No (1
COMNMU	1.24	2176	789	19	1%	2	Yes
CPLRT	1.23	1675	746	18	0%	1	Yes
ILRT	1.11	1312	574	17	1%	1	Yes
LIGHTT	1.20	1183	473	18	4%	1	Yes
NODCHG	2.10	1541	707	12	0%	4	Yes
NODCOM	1.69	3170	1535	27	4%	8	No (2
NODDED	1.32	1826	865	15	1%	5	Yes
NODDES	1.08	789	346	7	2%	2	Yes
NODDFS	1.17	1492	693	11	1%	4	Yes
NODDPR	1.08	756	295	5	1%	2	Yes
NODDSS	1.00	249	41	3	0%	0	Yes
		110	149	[illegible]			

Fig. 5 Static analysis results

The *standards penalty* is shown as a percentage and indicates the percentage of the code that contravenes the standard. These standards are customised from a standard list that is incorporated into the program. Experience has shown that any modules showing with >5% violation should be reworked.

The *complexity penalty* is measured in a scale of 0 to 10 and indicates parameters such as understandability and maintainability. Basically, the more complex the code, the more likely that bugs will have been introduced.

The *structure* check gives a straight Yes/No result based on matching templates of acceptable structures with the connection matrix of the basic blocks on a module-by-module basis.

DYNAMIC analysis uses regression and development test sequences to check paths and sequencing in a running environment and again has three areas of assessment of test cover – *Code*, *Jumps* (Explicit & Implicit) and *LCSAJ's* (Linear Code Sequence and Jump).

Each is shown as a percentage test cover. *Code* is straightforward executable statements obeyed in sequence, *Jumps* covers control flow branches and *LCSAJ* covers the more complex situations of executable statements sequences followed by control flow branches. A set of results is shown in Fig. 6.

Module Name	Test Files	Code %	Jumps %	LCSAJs %
CPLRT	5	91	89	72
ILRT	3	83	82	69
LIGHTT	1	79	76	50
OVACRT	2	91	83	70
PCST	13	92	87	86
PIT	4	91	85	67

Fig. 6 Dynamic analysis results

Experience has shown that there is considerable variation in the figures across the modules and the breakpoint at which it is worthwhile writing further test sequences varies both with the module and the type of check. Analysis has indicated that realistic targets are 90%, 80% and 70% for code, jumps and LCSAJ respectively, and any divergence from these figures results in an assessment of the test sequences to ascertain whether it is cost effective to extend the testing to give better cover for that particular module.

Overall, the Test Bed phase now pulls out 37% of the post compilation errors and the desk checking 43%.

5 Processor hardware development route

5.1 Background

Range Compatibility. The ICL Series 39 range of machines, announced in 1985, comprised 2 processor designs – internally code named Estriel and DM1. The range was a "compatible evolution" from the previous 2900 series – it was required that all customer software would run without change, but improvements to multiprocessors were made by incorporating nodal architecture where nodes are interconnected via fibre optic cables. Also, I/O channel capacity was improved by the use of similar connections. The SX machine is a further compatible evolution – all customer software must still run without change but further improvements, e.g. higher bandwidth fibre optic technology, have been made to multi-node and I/O connections.

It is worth noting that compatibility can both help and hinder the designer, e.g. the freedom to design a new internal hardware architecture in order to extract the optimum from a new technology is considerably constrained – compared with the situation of SPARC RISC designers who had no such problem. However, one might speculate that difficulties could arise after a few generations of SPARC technological evolution have passed and eroded their original optimisations. The advantage of the SX case is that several generations of test software exist, with which to validate the new design and, indeed, *ensure* that it is compatible with previous machines.

Technology.
SX logic chips are faster (1.6 ×) and have higher integration density (3000 gates vs 400 gates), than those in Estriel. The effect of improved speed is self-evident, but other attributes also change:-

First, more gates/chip implies fewer chips/machine, hence reliability is improved. This improvement is compounded by silicon processing improvements which means that a single chip is now more reliable. In addition, the design incorporates careful management of soft and hard failures in RAM devices to provide resilience. The result is that a single node will only see a catastrophic fail once in 2 years. Multi node configurations ensure that even this low rate of intrinsic failure is not allowed to crash a customer system.

Second, the risk of chip reworks after prototype switch-on is increased. With only 400 gates/chip, the Estriel designers were able to minimise the degree of "intelligence" built into the silicon itself. Many of the chip designs were "super MSI" style with much design complexity built into the PCB wiring and microcode. This meant that most of the design errors found at prototype stage could be fixed without changing the silicon. At 3000 gates/chip, the SX style is very different. Chip partitions contain so much more logic (with only twice the pinout) that complexity has to migrate from the PCB level. Hardware design errors now *normally* result in chip rework, hence the design development route placed heavy emphasis on the quality of the design released to build.

Performance
The marketing requirement for SX was to provide an aggressive performance improvement over Estriel. This was to be achieved with a processor technology 1.6 times faster and a store technology 1.4 times faster. In order to bridge the "gap" between the technologies and the performance requirement, the SX design needed to be much more complex, and to encompass much more functionality [Ref. 3]. The use of Picode retained some "soft" control, however many functions previously executed by microcode had to be taken on by the hardware. This added significantly to the difficulty of validating the hardware prior to design release.

A final perspective is that SX was the most powerful general purpose uniprocessor then announced in the world. A large, complex, "hard", design was necessary to implement it.

5.2 Partitioning [see Fig. 7]

It is axiomatic that any large design task is partitioned in organisational terms as closely as possible along functional and physical implementational lines – it is a matter of managerial judgement to decide upon the best route to steer through the conflicts arising from this bland statement!

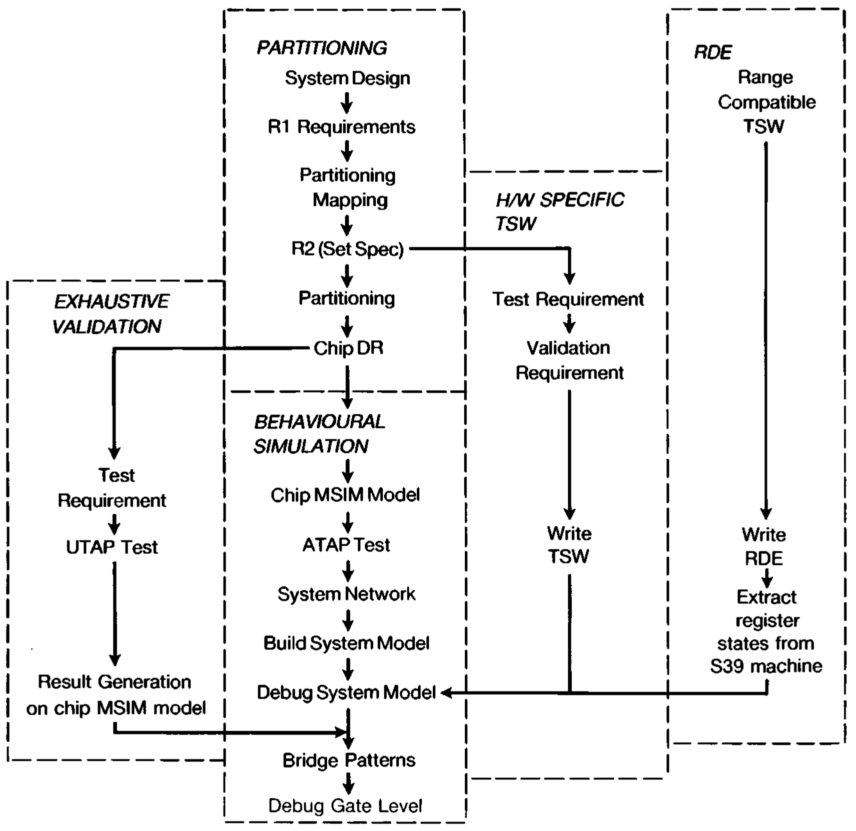


Fig. 7 SX hardware design validation

The SX hardware group was divided into implementation teams of five to six engineers each with a leader. Each team produced, and worked to, a "Set Specification" which encompassed the whole of the functional design unit (Set) for which they were responsible.

As described above [Section 2], the outputs of the system design process were documented statements of requirements – termed "R1". The implementation teams participated fully in the system design and they produced the Set Specification for their own design unit from the relevant R1 documents. This level of partitioning was finished when all R1 requirements had been mapped into implementation (Set) documents. The mapping was audited for completeness.

The Set Specification is the top level document for the team – it details the High Level Design (HLD) of the Set, i.e. the implementation of the R1 requirements and defines the chip partitioning. From it follow:-

Behavioural simulation.
Chip design requirement (CDR) – for each chip.
Test requirements – to allow software writers to test the Set.
Validation requirements – to allow simulation engineers and auditors to complete the validation of each individual chip prior to design release.

5.3 Behavioural simulation [see Fig. 7]

A two level approach to simulation was adopted for SX – behavioural and gate. The features of this approach, as opposed to gate level only, include the following:-

Advantages –
a) Behavioural code represents a verifiable specification of the design.
b) Early availability of a working model, enabling design faults to be fixed at lower cost.
c) The designer is able to visualise his design in functional rather than just physical terms, so providing an in-built "desk check".
d) Reduced processing power requirement enables a larger test set to be run.
e) Bridging of test patterns captured around a unit of design executed at behavioural level, into gate level, enables testing of gate level in isolation from the remainder of the design.

Disadvantages –
a) Designers require more tool skills.
b) Extra manpower required for coding.
c) Cumbersome route for design fixes later on.
d) Clock and diagnostic services only exercised at gate level.

The SX designers coded a model of each chip, using an in-house simulator – MSIM. The code was subjected to a style audit, mainly for understandability. Each model was then alpha tested with specific hand written test patterns (ATAP) – this process removed 85% of code faults and a few design faults. The individual chip models were networked together to build the complete system model – this came to 100K lines of code. The subsequent debugging identified 400 more code faults and 550 design faults.

5.4 Range definition exerciser – RDE [see Fig. 7]

Although simulation is now used in the verification of every significantly complex ASIC design, it is important to recognise the limitations of the technique. The most obvious of these is performance – one second of real time running of the SX processor is equivalent to one continuous month of behavioural simulation running using all the service machine processing power available to the design project!!

Given that fault situations in I/O, inter node, slave and pipleline, can sometimes take hours of real time to become manifest, it follows that the most test cover possible must be compacted into the least time. In practice, the limit on the size of the complete simulation test suite is a few million clock cycles.

Given the complex and "hard" nature of the SX design, the engineers wanted to run tests equivalent in cover to an existing range compatible test suite – known as Range Definition Tests (RDT). Unfortunately, RDT was, and is, well beyond our simulation capability – a billion clock cycles would take 10 years to run, without iteration for any fixes!!

However, it was estimated that the ratio of target test instructions to data generation and checking instructions was approximately 1 to 10 000. This suggested the possibility of removing most of this overhead by running only test instructions on the simulation, whilst generating data and checking results in real time on a Series 39 host machine.

The resulting tests became known as Range Definition Exercisers (RDEs). The method of checking the results relied on the ability of the Level 80 microcode to monitor register states at the completion of PLI instructions.

The RDEs were run on a dedicated Level 80 with the monitoring microcode enhanced to write the PLI visible registers, at the termination of each PLI instruction, to a dedicated area of the main store. This area of main store was then dumped to magnetic tape and transferred to the service machine for processing. Subsequently, this data could be used, together with the System simulation model, to compare Level 80 and SX visible register sets at the end of each PLI.

There are 17 RDEs, 8 testing PLI and 9 data dependent tests. The PLI tests took a total of 150K clock cycles whilst the data dependent tests took 600K cycles for single and double length arithmetic, or 7600K cycles if quad length was included as well.

5.5 Hardware specific test software [see Fig. 7]

Each Design team produced a Test Software Requirements document identifying areas of their part of the SX design which were not tested by the RDEs or other tests. The Test Software Group responded to this document with a specification of the tests, which was then reviewed with the project before low level program specifications for the Test Software were produced. As the tests were specific to SX they could not be validated fully prior to use on the System simulation.

5.6 Exhaustive chip validation [see Fig. 7]

A feature of running PLI level test software is that a homogeneous test cover of every hardware function is unachievable – for example, some gates in the arithmetic unit may be exercised in nearly every clock cycle in the test, whilst some gates in an error management function or slave control may only be exercised once after millions of clock cycles of testing. Any efficient test strategy must address these "sparse" areas. The hardware specific test software (section 5.5) allows for testing the design at a lower level of detail than PLI – using Picode. However, this still leaves some areas of the design with little cover.

These areas in SX were identified from the chip design requirement in a test requirement document which is chip specific. Bit patterns were applied to single chip MSIM models and gate level models, and the results compared. These tests were known as Unit TAP (UTAP).

5.7 The future

The foregoing was intended to give an insight into the complexity of a modern processor, and into the innovation required to design and validate it. Future designs will have even tougher targets to meet, especially reduced time to market. The ultimate challenge is to switch on a prototype free of any hardware deficiencies – but a more modest step would be to switch on a million gate design with only 10 faults!!

6 Configuration management

At the start of the SX Project, the need was envisaged for a system to allow a high degree of control in the manipulation of large volumes of design data through many overlapping development steps. Such a system had to provide facilities to allow effective control at both design team level, and at system integration level. In addition, the system needed to be capable of integrating a large number of different design tools into a uniform, interactive, and easily usable environment.

At that time no suitable commercial tool was available to meet the project requirements. The SX Design Database (SXDB) was, therefore, developed to support both hardware and PICODE development. SXDB now provides a high level of control, tools integration, data accessibility and performance. In most of these respects, SXDB is still ahead of commercially available systems.

The basic facilities of this system are listed below:

1) Product/sub-product version control
2) Audit trail facilities
3) Integrated documentation system
4) Tools invocation
5) Edit facilities
6) Interactive report/enquiry facilities

Each of these facilities is described below.

6.1 Product/sub-product version control

The versioning mechanism used within SXDB was initially invented for the CADES database, used in the VME development route [Ref. 4]. (The ICL DDS system has also borrowed some of these concepts). The basic philosophy is one of having a version state for an entire sub-product. The version state of any sub-product is an amalgamation of objects changed at that version, plus all other objects "inherited" from lower versions.

The creation of new version states for a sub-product are determined at team level as needed by the phased development strategy. Mechanisms are provided to lock and freeze version states, to stop further change at any particular version.

A "System Version" mechanism allows a single version of each sub-product to be related to form a single version of the total SX product. The combination of System Version and sub-product version facilities allows flexible control at both System and team leader level; in particular, System simulation models can be built and maintained for any development stage, unaffected by the later sub-product development.

6.2 Audit trail facilities

Each version of an object stored/modified in the SX database has a set of control details associated with that object. Such details include: date/time created, who created it, and intermediate freeze state. In addition, most objects are given a textual description for every change applied.

6.3 Integrated documentation system

All documents within the SXDB are assigned their own versions. Design documents are directly related to objects within the design hierarchy, and mechanisms are provided to record the impact of changes arising from either documents or objects.

6.4 Tools invocation

There are two basic classes of tools controlled via SXDB: "loosely"-coupled tools, such as the network editors running on graphics workstations, and "tightly"-coupled tools. The "tightly"-coupled tools are all invoked directly via the SXDB menu interface common to the majority of other facilities, including checks, reports, etc.

6.5 Edit facilities

Facilities for editing of file based design data are provided by SXDB. These facilities allow the current design state to be checked-out into a scratchpad filestore, edited and checked-in to the database.

For graphical data the file is transferred to a graphics workstation for edit. For textual data, VME edit facilities are utilised under direct control of SXDB.

6.6 Interactive report/enquiry facilities

A comprehensive set of interactive reports and enquiries is included within SXDB. Each of these is available via the SXDB menu interface, and provides fast, online response. Enquiries provide details for a single database object; reports provide similar details for multiple database objects selected via common attributes.

7 Measurement within the development process

There were two basic reasons why it was decided to apply extensive measurement to the development processes. First and foremost, it allowed assessment of whether the development was on schedule and whether the assumptions made about the many design parameters were still valid.

Second, measurement allows identification of the key factors that control the process and ensures they are properly applied at the planning stage of the next project. Many of the initiatives this time were based on measures taken on the previous development.

The prime measures throughout the development process were directly related to the delivered quality of the product. However, another key measure is the amount of effort expended in each phase of the development route. This directly affects the time to market and needs to be understood in detail if management is to make the right decisions on the best use of resources.

Taking a simple breakdown of the development route for NSX firmware, the synthetics used in the initial planning were as shown in the table below.

Devt Phase	HLD	LLD	CODE	DESK CHECK	TEST BED	M/C TEST
% Total effort	24	24	10	2	25	15

A user friendly database application was developed that would allow individuals to enter details of how they spend their time, and provided direct correlation with the planning parameters.

The WORK STATS database provided over 90 separate categories structured in a three level hierarchy. Input was extremely simple and checking facilities ensured all entries were valid.

The database has been in use since October 1987 and covers the full development cycle of several projects.

Extensive analysis facilities were built in allowing extraction of the data to be output for individuals or teams, for any category, and over any timeframe. Later, incorporation of spreadsheet facilities allowed output to be displayed in various graphical forms such as that shown in Figure 8.

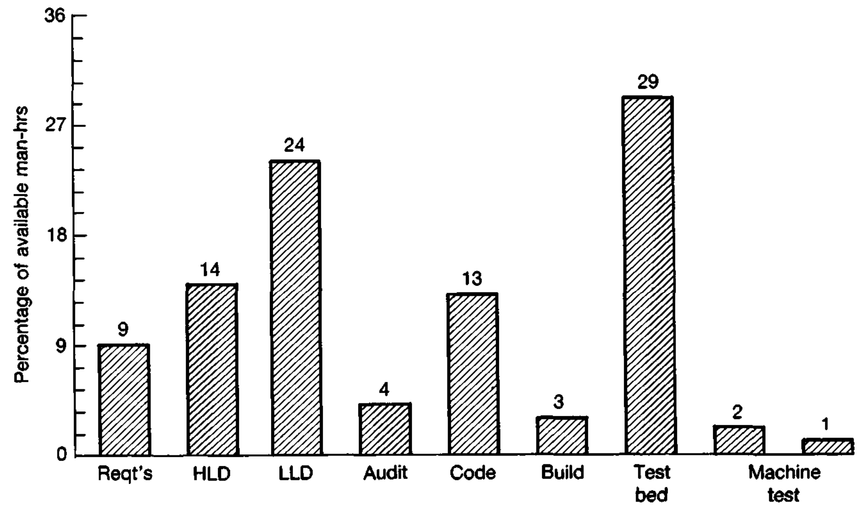


Fig. 8 Effort profile — application firmware development

This gave the breakdown of effort against each of the development phases, allowing the planning synthetics to be monitored directly. Comparison with the initial planning synthetics shown above highlighted variations that needed to be assessed.

Particular points to note are that the effort prior to coding accurately matched the synthetic and, more dramatically, the effort required for on-line machine validation was reduced from the expected 15% to less than 4%.

8 Concluding remarks

The development routes cover an amalgamation of tools, methods, checkpoints, metrication and design control. The successful implementation of this methodology however requires discipline and commitment from each and every member of the team. This commitment has resulted, to date, in an order of magnitude improvement in the delivered quality of the products.

The processes described are a few of the many involved in the development of SX. The challenge with each mainframe development is to understand our processes better and identify where changes to the methodology, techniques or tools can lead to the levels of improvement needed to maintain our position at the forefront of mainframe development.

The examples given clearly show that, by careful analysis of the previous development cycle and by assessment of the advances made in the software packages available, considerable improvement can be made in the next development cycle. It is this challenge that ensures designers, in whatever discipline, will have an interesting and exciting future.

Acknowledgements

The work described reflects the efforts of all members of the SX project team.

The authors would like to thank everyone involved for their efforts and their help in preparing this paper.

References

1 Birtwistle G.M. Discrete Event Modelling on Simula. Macmillan Publishers Ltd. ISBN 0-333-23881-8.
2 Ward P.T. and Mellor S.J. Structured Development for real time systems. Prentice Hall ISBN 0-13-854787-4.
3 Allt G, Eaton J.R. and Hughes K. The SX Node Architecture. ICL Technical Journal Vol. 7 Nov 1990. 197-211.
4 Snowdon R.A. "CADES and Software System Development". Software Engineering Environments, North-Holland Publishing Company, 1981.

Physical Design Concepts of the SX Mainframe

C. Shaw
Corporate Servers Product Group – ICL West Gorton, Manchester, UK

Abstract

The spectacular improvements in silicon chip circuit integration densities and speed, which have now been sustained for over two decades, are well known. The effective exploitation of modern LSI technology in a high performance mainframe computer places challenging demands on the hardware and packaging which supports the electronic subsystems.

This paper describes the implications of these demands on the physical design of the SX mainframe and outlines some of the resultant design solutions. Contents of the major functional hardware sub-systems within the SX node are summarised.

1 Physical design requirements

In order that the machine can be readily adopted by existing customers whose installations require increased computing capacity it is important that the new machine places no greater demand on its environment than its predecessor (typically Series 39[1] Level 80 systems in this case). The parameters of particular importance are:-

Overall Dimensions
Compactness brings numerous benefits: for the customer a reduction in the computer's footprint provides opportunities for more convenient siting of equipment as well as efficient use of premises. For the manufacturer delivery is made easier and transportation costs are minimised. In the case of the SX mainframe by restricting the overall height to 1500 mm it becomes possible to use standard air-freight containers for shipments to customers outside the UK.

Mainframe installation can also be simplified as the overall dimensions reduce. If the complete mainframe can be transported in one piece the need for on-site re-assembly is eliminated. Moving the equipment within the customer site, when purpose-built premises are not available; eg up liftshafts, through narrow corridors and door openings etc., is made more straightforward.

The overall dimensions of the SX mainframe are 1760 mm length, 770 mm width, and 1450 mm height.

Operating temperature range
Whilst component reliability benefits from maintaining low transistor junction temperatures, it is desirable to permit as wide a range of ambient operating temperature as possible. Imposing narrow limits on the operating range requires excessively tight controls on the computer room air-conditioning and results in increased running costs. The permitted range should ensure that at low room temperatures there is no risk of condensation forming within the mainframe when the humidity levels are high, and at the upper limit the temperatures of high dissipation components remain within their design targets. SX nodes are able to operate through an ambient temperature range of 15–29°C.

Acoustic noise emission
Our design objectives are to produce mainframes which have noise levels comparable to equipment, such as Workstations, designed for use in office environments (typically 55–60dBA sound pressure when measured at 1000 mm). This helps to ensure that overall computer room ambient noise, when many units such as disc sub-systems and printers are located together, is kept within comfortable limits.

Electrical power consumption
For a chosen computing performance the electrical power consumption is primarily dictated by the choice of circuit technology. Even so, careful design of the internal power supply and cooling systems for maximum efficiency can contribute significant energy savings which directly benefit the mainframe's lifetime operating costs.

2 Performance implications

The reduction in logic cycle times needed to achieve the performance of a modern mainframe has required not only the improvements in silicon chip processing but also comparable changes in the packaging of the logic circuit assemblies to reduce interconnection path lengths. The speed of light remains a constant impediment in the race for ever more MIPs!

Component densities on the logic assemblies have increased as a result of reductions in package size and finer trace geometries together with more layers on the printed circuits. The logic assemblies are more closely spaced to shorten signal paths. All these factors tend to result in increased heat density and dissipation within the logic units. In order to maximise the intrinsic component reliability it is imperative that these higher dissipations are managed without increasing the component operating temperatures.

For maximum mainframe peformance Emitter Coupled Logic (ECL) remains the silicon technology of choice. This is a power hungry logic family. The circuits used for the SX central processor unit (CPU) have a speed-power product of 0.4 picoJoule/gate[2] which when packaged at the gate densities achieved on the SSC exceed the heat transfer capacity of a forced-air cooling system.

3 Technology overview

The major functional components and interfaces of the SX Node are shown in Fig. 1 and a summary of each unit's relevant characteristics follows.

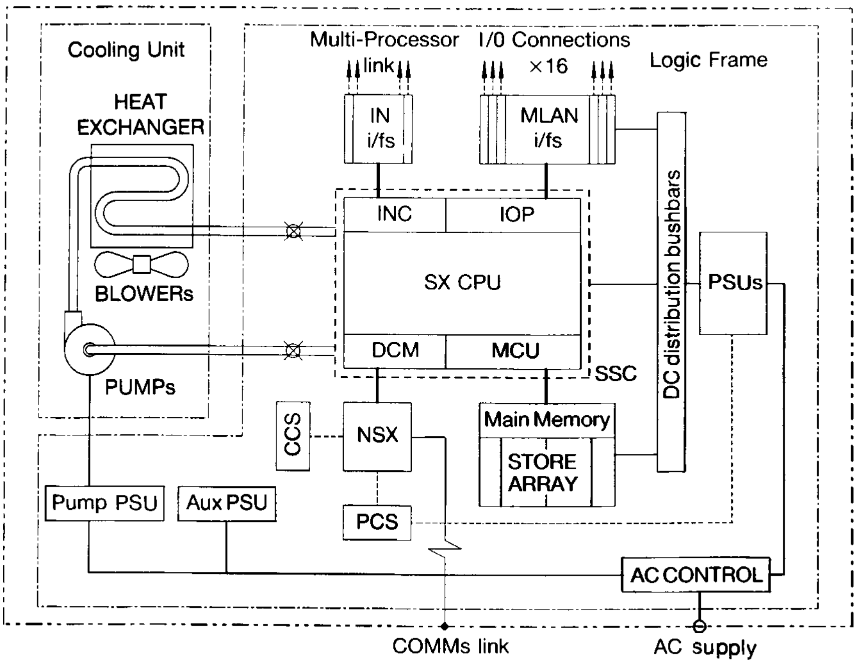


Fig. 1 SX Node Functional layout

3.1 CPU

In the SX design it has proved feasible to extend the "Single Board CPU"[3] concept adopted by Fujitsu and incorporate not only the CPU but also the performance-critical I/O and Memory Control sub-system logic on the same pcb. This has major advantages in reducing transfer time delays on the associated interfaces.

The Sub-System Carrier (SSC) forms the heart of the SX CPU. The SSC technology is a variant of one developed by Fujitsu Limited for use in their M780 range of mainframe computers. The logic and interconnections are entirely specific to the SX computer and wholly designed by ICL.

The SSC comprises a 540 mm × 488 mm 42 layer printed circuit assembly on which are mounted up to 336 ECL LSIs. This assembly is sandwiched between a pair of Conductive Cooling Modules (CCM)[4] which transfer the heat dissipated by the logic circuits. Each LSI has a nominal power dissipation of up to 9 watts and hence the CCM pair has a heat transfer capacity exceeding 3 Kwatt.

A maximum of 700 logic signals are provided to connect the SSC with other functional units in the SX Node. These signals are carried by miniature coaxial wiring to ensure low electrical noise and arranged into 14 multiway ribbon cable and connector assemblies.

3.2 Main Store Unit (MSU)

The MSU is designed to use Dynamic Random Access Memory (DRAM) components having typical access times an order of magnitude greater than the CPU clock cycle. From an SX logic designer's viewpoint the DRAM component timing tolerances (skew) arising from normal semiconductor manufacturing variations exceed the logic beat. Within the Main Store system, in order to minimise this skew, which represents wasted potential machine performance, it is desirable to both shorten and equalise the signal paths to and from the DRAM elements.

In the case of the SX main store skew has been optimised by the design of a double-sided motherboard assembly as shown in Fig. 2. All store dataflow

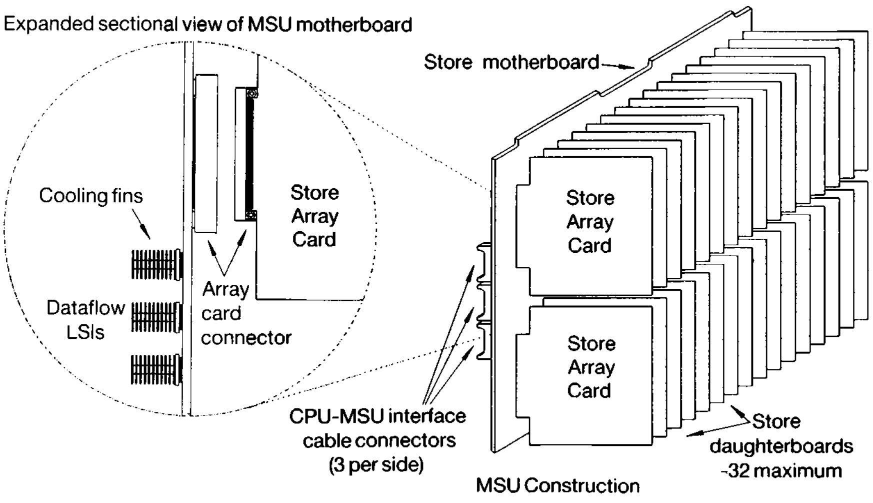


Fig. 2 MSU construction

circuits are arranged on one face of the motherboard and connect via short signal paths to the Array card connectors on the opposite face.

An "active" motherboard containing ECL dataflow multiplexing and direct drive to all Store Array cards was designed to maximise performance.

High storage capacity within the available volume is achieved by using double-sided surface-mount technology for the Array cards which contain the DRAM components. The fully configured SX Main Store contains 32 array cards.

By limiting the maximum utilisation of the logic circuits within each LSI on the MSU, and arranging these devices in only three rows across the full width of the motherboard, the power dissipation is restricted and it is possible to use an air-cooled variant of the CPU LSI technology.

Airflow through the MSU required careful design. The dataflow LSIs on the motherboard and some regions of the Array cards need high cooling capacities. Reduced air speeds arising from the additional volume created by the absent Array cards when the MSU is depopulated as in minimum configurations, must not degrade the motherboard cooling.

3.3 External interfaces

With the exception of the AC mains supply cable and the communications line linking to a conventional modem for remote maintenance (Telesupport), all interfaces between the Node and other SX system components (eg discs, printers, workstations) are provided using high speed serial fibre-optic connections. This greatly simplifies potential cable shielding and earth-loop problems and reduces the bulkiness of the interface cables and connectors.

Optical fibres (Macrolan)[5]
The proprietary ICL optical fibre connection system introduced when Series-39 was launched is also retained for SX systems. It enables processing nodes and the I/O controllers within a system to operate with separations of up to 2 Km, thereby facilitating remote operation and dispersed systems for "disaster proofing" etc..

3.4 I/O & Inter-Node Couplers

Macrolan is used both for I/O Controller $\leftrightarrow$ Processor connections and inter processor links in multi-node systems.

Inter processor data is transferred by a 200 Mbps variant of the Macrolan normally used for I/O traffic. A special purpose optical fibre link is also used to maintain data synchronism among each of the Nodes in a multiprocessor system. This "Transmission Sequence Number" (TSN) system employs the same optical interface components as does Macrolan.

3.5 Node Support Computer (NSX)

The NSX takes care of Initialisation, Self-test, Reconfiguration and Remote (diagnostic) access for maintenance. All development commissioning takes place using this mechanism – many of those customers seeing SX during visits to West Gorton have been surprised by the absence of engineers around the prototype machines, it being more efficient and comfortable to access the systems from terminals in the design offices.

The NSX and the Power and Cooling Control Systems (qv) are separate microcomputer units which operate independently from the main CPU. Power to these units is supplied by dedicated Auxiliary PSUs so that control functions are available whenever AC is applied at the node.

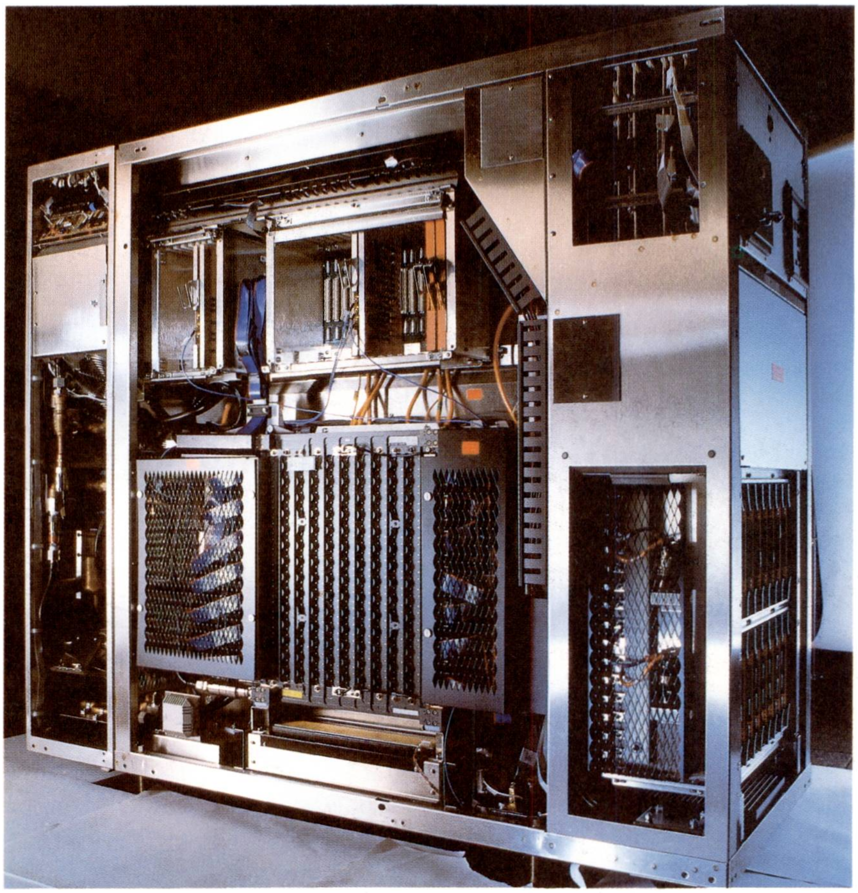


Photo 1 SX Node – internal view

4 Reliability

The advances in silicon technology have produced dramatic improvements in the reliability of computers. Electronic component failure rates are now typically better than 1 in 10^7 hours. Customers now expect their mainframes to function continuously for months without service breaks. It has become essential to ensure that the electro-mechanical sub-systems used for powering and cooling the CPU achieve comparable reliability.

4.1 Resilience

The basic strategies for ensuring high reliability of the node physical design sub-systems have been to eliminate electro-mechanical components as much as possible and, where this could not be achieved, to provide functional duplication. Examples of these strategies are detailed in subsequent sections.

5 Accessibility

Previous members of the Series 39 Processor family achieved very compact performance/volume ratios at the expense of ease-of-access to the individual logic assemblies. The high reliability of these designs meant that this was not a customer perceived problem, although for ICL increased system upgrade and enhancement times resulted.

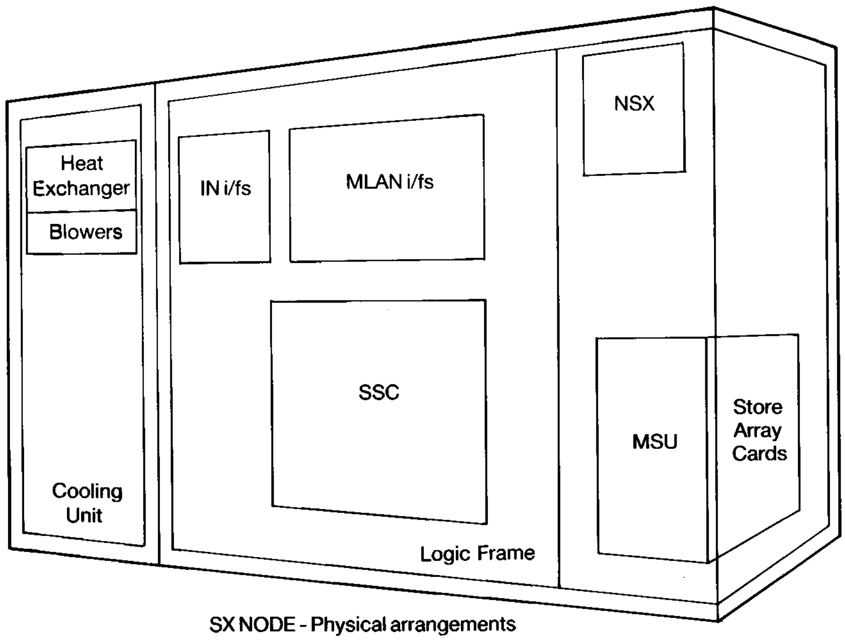


Fig. 3 SX Node – internal arrangement

With the SX node packaging design much care was applied to ensure that each field-enhanceable logic assembly could be accessed for upgrade/replacement without needing to remove other sub-systems. This accessibility is evident from Photo 1.

6 Power distribution

The major logic sub-systems in the SX mainframe are supplied from four low voltage DC rails. Rail currents of up to 750 amps are catered for. Voltage regulation has to be carefully controlled to eliminate noise pick-up on the logic interconnections.

As well as the large component timing variations which occur, process spreads in semiconductor manufacture result in substantial variations in circuit power consumption – $\pm 30\%$ of nominal is not uncommon. This poses a problem for the computer packaging designer who has to provision power to meet the worst case demand, whilst also seeking to minimise the overall machine size.

In order to minimise component dissipations and maximise logic gate switching speeds, semiconductor designers are progressively reducing both the logic switching levels and the operating voltages for the circuits. This trend further complicates the packaging design as high current PSU conversion efficiencies tend to diminish at low output voltages and noise susceptibility increases as logic switching levels reduce. Great care is needed to ensure that the DC distribution design minimises any voltage differences across the logic assemblies in order to preserve the noise immunity of the digital circuits.

6.1 Power Supply Units

All main DC rails in the SX node use the same switched-mode Power Supply design. Each unit can supply over 800 watts of regulated low voltage DC. Up to 15 PSUs are installed in a fully configured Node.

High AC to DC conversion efficiencies are achieved by the use of power Field Effect Transistors (FET) in the inverters. Switching frequencies are $\times 5$ higher than is feasible with conventional junction transistors. The very short FET on/off transition times greatly reduce the switching losses and the increased switching frequency allows the use of smaller magnetic cores in the wound components. In combination these factors allow the PSU volume to be reduced whilst maintaining adequate cooling airflows. The output power density achieved overall by the PSU approaches 0.2 Kwatts/litre.

PSU efficiency is an important physical design parameter as the conversion losses can account for a significant proportion of the total cabinet dissipation. As the DC rail voltages reduce, voltage drops across the PSU output rectifiers form an increasing proportion of the high current circuit and hence the efficiency decreases.

One complication arising from the higher switching frequency is the potential for increased Radio Frequency Interference (RFI). The high inverter voltages and very fast switch rise and fall times can result in a broad spectrum of electro-magnetic-radiation from the PSUs which need careful design and shielding to avoid RFI propagation.

All PSU connections, ie AC mains input, DC output and the control interface are pluggable. This reduces system assembly and replacement times and eliminates the need for tools in these processes. By placing all electrical connections on the inner face of the PSUs the usual protective cover plates are no longer required and effective RFI shielding becomes simpler.

The Power Supply Unit nominal DC output voltage can take one of four values (between 2 and 5.5 volts) according to the setting of link connections on its control interface. This allows any unit to be used on any voltage rail without the need for set-up adjustments, thus simplifying spare parts inventories and eliminating the risk of mis-connection. The links are hard wired at each interface connector in the PSU mounting rack to provide a "position selectable" voltage setting.

Three phase line-to-line AC is supplied to every PSU in order to ensure that the line currents in each phase are balanced and for increased resilience to transient supply disturbances affecting one or more phases.

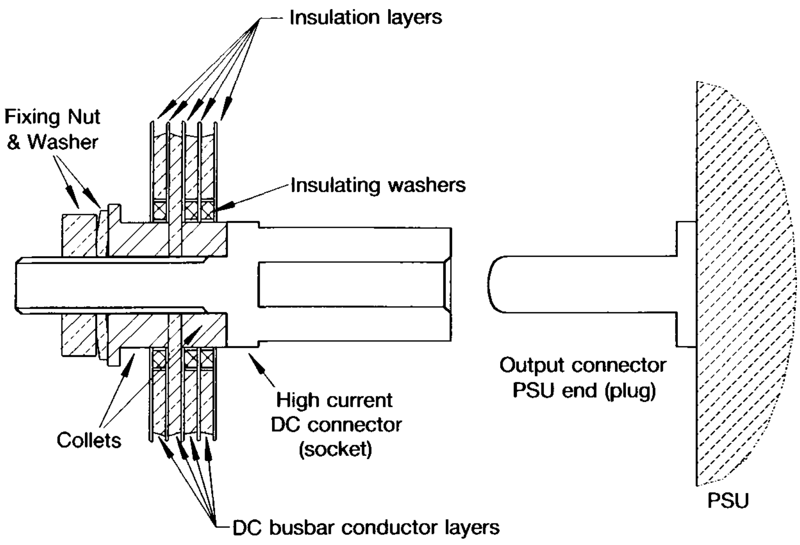
6.2 DC busbars

The DC distribution from main PSUs to SSC and MSU takes the form of a multilayer planar construction, rather like a heavy-duty pcb, which runs along the centre of the logic frame. Fig. 4 illustrates this with a sectional view through a PSU output connector. The conductor for each voltage rail is interspersed with an insulating laminate and the complete sandwich forms a very stable matrix in which the low impedance DC connector sockets can be accurately located. The configuration of insulating washers and conductive collets shown in figure 4 ensures that the connector socket is securely clamped to make good contact with the selected busbar layer whilst remaining well insulated from all other DC rails. Our measurements have shown that this method, in conjunction with suitable connectors, results in no greater DC voltage drops than would arise using conventional braided flexible links bolted to PSU terminals.

Every DC rail contains one more PSU than is required to supply the worst-case current demand. This "N + 1 Resilience" ensures that node operation is not disrupted should a PSU fail in service and so replacement of any faulty units can be deferred until a convenient maintenance opportunity occurs.

6.3 Power Control

The performance of the PSUs and the DC rail conditions are continuously monitored by a dedicated microcomputer sub-system. This unit also pro-



Insulation layers
Fixing Nut & Washer
Insulating washers
Collets
High current DC connector (socket)
Output connector PSU end (plug)
PSU
DC busbar conductor layers

Fig. 4 DC busbar construction

vides continuous control of "current sharing" – a technique which ensures that all active PSUs on any voltage rail are delivering the same proportion of the total current demand. When PSUs are connected in parallel such a technique is necessary to compensate for minor differences in the load impedance seen by the individual units which otherwise would result in imbalances driving some PSUs into current-limit and others into no-load conditions.

The power control system (PCS) also controls voltage sequencing during node switch-on and switch-off and has a communications link to the Node Support Computer. On successful completion of the power-on sequence the PCS uses this link to initiate logic startup of the SX node. Power system fault diagnostics are reported to the remote maintenance system via the NSX link.

In the event of a supply or power fault condition causing a node shut-down or preventing startup, the power system status and any diagnostic signatures are stored in a battery powered memory until normal reporting can be resumed.

7 Cooling

7.1 Cooling technology selection

The heat transfer capacity of any coolant is proportional to mass flow per unit time in contact with the items being cooled. For maximum heat transfer within a particular coolant volume liquids are greatly superior to gaseous coolants. However the simplicity of forced air cooling (eg no containment or insulation problems etc.) leads to its general use in preference to liquids but as the component dissipations rise the air speeds and component fin dimensions have to increase to unacceptable levels. Acoustic noise also increases with higher air speeds and design of suitable blowers becomes impractical.

7.2 Cooling Unit functions

Design considerations
Although the CPU technology dictated that liquid cooling was required, a key objective was to ensure that this did not make installation, maintenance or operation of the SX mainframe more difficult for our customers. Our aim was to achieve a design in which the customer's premises provide electrical power and normal computer room air-conditioning without any other awareness of the cooling technology used within the mainframe.

Most important was the desire to eliminate dependence on a direct connection to the building air-conditioning plant. Such a linkage, which is sometimes used in other designs of liquid-cooled mainframes, would constrain the installation of the node in buildings with minimal heating & ventilation controls and could drastically impact the computing system reliability because of plant failures/outages unrelated to the node.

Simplicity of operation
The cooling system should not impose a greater level of user skills or intervention than is required for the correct functioning or control of the mainframe's logic circuits. Unattended operation and deferrable maintenance of any defective components are therefore essential.

Reliability – Duplication of system-critical cooling components
Coolant pumps and their associated power supplies are duplicated to avoid system outages in the event of malfunction. Only one pump is in use at any time, but regular changeover of the pump in operation occurs under normal conditions. This ensures that latent faults are not hidden.

For each functional grouping of blowers in those areas of the node requiring forced-air cooling, "N + 1" resilience is provided so that adequate cooling air flow can be maintained should a blower fault occur.

For both the liquid and air circuits the design must prevent back-circulation of the coolant under pump or blower fault conditions.

Water quality
To achieve a reliable operating life (which could be as long as a decade) it is important to eliminate sources of contamination of the coolant. Prior to final assembly all the piping components undergo a rigorous cleansing process

which includes an extended period of rinsing in an ultrasonic cleaning bath. The water used in the coolant system is de-ionised and purified to pharmacological standards.

These measures minimise the inclusion of unwanted particulate debris and biological contaminants. In consequence there is no food supply and, after initial filling and system switch-on, any anaerobic bacteria are quickly starved and die. The growth of aerobic bacteria is inhibited because the closed-loop coolant recirculation does not provide an interface for re-oxygenation of the water.

Materials selection
All materials in the coolant circuit must be carefully selected to avoid corrosion. This consideration includes not only the pipes and pipe-fittings but also such items as the constituent metals used in any brazed joints so that electrochemical reactions are minimised. It is also important to control the composition of plastic materials such as flexible hoses, valve and pipe seals etc., as many of these items normally contain compounds from which undesirable ions such as halides can be leached into the coolant. Materials used in the piping manufacturing processes such as fluxes and acidic cleansers also require evaluation.

Initial SX Cooling Unit prototypes were manufactured using a stainless steel piping system, but considerations of materials availability, compatibility and manufacturing processes led to the subsequent choice of copper based systems.

Extensive testing
Validation of the cooling system design has involved extensive testing, including trials of materials compatibility, component performance, water quality, and mechanical integrity as elements of a program designed to determine the lifetime behaviour. Where possible, test conditions in excess of normal operating conditions have been used to accelerate the detection of long-term effects.

Periodic maintenance
To ensure that optimum coolant quality persists throughout the life of the node a simple annual maintenance procedure has been devised. The cooling circuit contains a pluggable cartridge filter which can be replaced without interruption to the normal operation of the node. Whilst the filter cartridge is being renewed an ion-exchange unit is installed in place of the filter. This purges any ionic built-up and restores the coolant to its initial condition.

Control – Pressure & Flow
Incorporation of all the liquid-cooled sub-systems into a single assembly has simplified the coolant circulation design by ensuring that only a fixed flow rate would need to be provided. During initial design the option of providing a free-standing cooling unit capable of supporting two or more nodes in a

multi-processor system was studied. Such an option would have reduced the cost of liquid cooling per node but was rejected because of the design and logistics complications that resulted.

The coolant circuit pressure and flow rates are factory preset during final assembly and test. Each pump is powered by its own AC-AC converter. This isolates the pump performance from the effects of changes in the mains supply voltage and/or frequency.

7.3 Cooling Control

In a similar method to the Power Control System an autonomous microprocessor based unit is used for control and monitoring of the cooling system functions. The Cooling Control System (CCS) continuously monitors coolant pressure, flow, and temperature. Abnormal readings can invoke blower speed changes to increase cooling airflows or pump changeovers as required. If the abnormal conditions persist and exceed preprogrammed limit values the CCS will shutdown the node to prevent possible damage.

Coolant conductivity is also monitored to check that corrosion or chemical contamination are not occurring.

Rotation speed sensing circuits are fitted on all blowers and monitored by the CCS. This provides for accurate resolution to the faulty unit when blowers are operating in parallel and ensures detection and fail reporting occurs before there is any risk of overtemperature conditions.

System protection
Whenever AC mains is reapplied to the mainframe the CCS performs a series of integrity tests to confirm that the all system cooling functions are operational. Only when these checks are completed and the node environment is satisfactory (eg the inlet air temperature from the computer room is within specification) does the CCS enable an interlock to allow DC power to be applied to the logic circuits. This interlock has "fail-safe" features so that any subsequent condition leading to loss of control will cause the DC to be removed.

Cooling Unit sensor data and operating status readings can be interrogated remotely via the NSX. Fault signatures are passed onwards via this route to the normal system diagnostics reporting processes. In this way, the same levels of maintenance information as exist for the CPU hardware and software, are made available for the physical design.

8 Construction

Photo 1 illustrates the internal arrangements of the SX node as seen looking onto the main logic bay. (The opposite side of the unit houses blowers, AC control, and the node power supplies). For clarification figure 3 identifies the main functional units visible in Photo 1.

8.1 Steel frame

The logic frame is made from welded stainless steel sheet. This permits ease of fabrication and provides great strength. Additional benefits from the use of stainless steel include the elimination of surface treatments and the provision of continuous electrical contact with the external covers for RFI screening is simplified. By suitable positioning, the frame members also function as air-cooling ducts and mounting brackets for attaching the logic units.

In contrast, a tubular welded frame construction is used for the cooling unit to give all-round accessibility during final assembly of the piping system. This construction was also dictated by the need to maximise the unobstructed cross-section for the heat-exchanger and the airflow through it.

Both frames are bolted together during the final stages of assembly and subsequently treated as an indivisible cabinet.

8.2 External covers

In addition to their obvious styling role, the cabinet covers perform important functions including attenuation of acoustic noise and RFI, inlet air filtration and the dispersion of exhaust air.

The panels around all sides of the node are hinged and open for maximum ease of access. Access "under the covers" is only required for engineering purposes and so in normal operation the panels provide isolation from any voltage and current hazards. All cooling air used by the node is drawn into the cabinet through low level grilles around all sides of the unit. This positioning of the inlets avoids the need for under-floor air feeds and hence the SX design can operate with much shallower raised floors than previous designs. The air-grilles are detachable and incorporate foam filter pads which can be exchanged or cleaned by an operator without needing to open hinged side panels in the node.

The transparency needed for drawing cooling air into the cabinet conflicts with the requirement for attenuating RFI efficiently. Prevention of RFI emission is achieved by providing a continuous electrically conducting screen enclosing all the electronics. Any holes in this screen, such as may be required for cable connections or display panels, must be kept to a minimum as they can behave like slot-feed transmitter aerials for HF signals. As logic switching speeds increase a broader spectrum of emission frequencies occurs and to minimise leakage all unavoidable apertures must be kept as small as possible.

8.3 Use of materials

In general the expected manufacturing volumes lead to a construction based on use of sheet metal piece parts. The high tooling costs for moulded plastic components are seldom amortized by the resultant reductions in parts costs. However there are areas where the component complexity or finish is best realised by plastic mouldings. One such example in the SX node is the inlet air grille. Fourteen air grille panels are fitted around the node and for this part the quantity involved in conjunction with the intricate surface shapes of the louvres made the development of a low temperature Noryl moulded part worthwhile. The use of a moulded component enabled styling features and retainers for the air-filter pads to be built into the basic part.

9 Conclusions

The technology demands for high dissipation densities and compact signal interconnects in SX have been reconciled with the needs for ease of manufacture, installation and maintenance to produce a mainframe having one of the best performance to volume ratios available. A system having more than four times the processing throughput of its predecessor has been packaged within the same overall volume.

The next challenge will be to maintain the advances in compact packaging for mainframes whilst meeting all the support needs for the next generation of very high performance digital circuit technologies.

10 Acknowledgements

The physical design solutions incorporated into the production model of the SX mainframe are the combination of the ideas from numerous ICL staff throughout Mainframe Systems and Manufacturing & Logistics divisions who have been involved with this project. The final result would have been much less satisfactory without their enthusiasm.

References

1 CAMPBELL-KELLY, M.: ICL Company Research and Development, Part 3: The New Range and other developments. ICL Technical Journal pp. 781–813 (Nov 1989)
2 OHNO, K. et al.: Semiconductor Technologies for FACOM M780. FUJITSU Scientific & Technical Journal, 23, 4 pp. 216–225 (December 1987)
3 NISHIHARA, M. et al.: Single Board CPU Packaging for the FACOM M780. FUJITSU Scientific & Technical Journal, 23, 4 pp. 226–235 (December 1987)
4 YAMAMOTO, H. et al.: Cooling System for the FACOM M780. FUJITSU Scientific & Technical Journal, 23, 4 pp. 243–254 (December 1987)
5 STEVENS, R.W.: Macrolan: A high performance network. ICL Technical Journal pp. 289–296 (May 1983)

PAPERS ARISING FROM COLLABORATIVE R & D

Foreword

It was in the early 1980's that the idea of R & D collaboration between European IT Companies, Research Institutes and Universities first became a serious topic for discussion. The IT community, the European Commission and various National Governments in the Community were increasingly concerned that the fragmented nature of the IT industry was making it more and more difficult for individual companies to devote sufficient resources to covering the ever-widening field of IT research. The deployment of effort in this area in both the United States and Japan seemed to be not only greater, but better focussed, at least in the latter case.

The result of this concern was on the one hand the ESPRIT I programme of collaborative Research and Development, launched by the European Commission in consultation with industry and academia, and the UK Government's Alvey Programme. Both were set in motion in 1984.

The ESPRIT I programme invited proposals for cross-border collaborations funded to 50% of cost by the Community; the Alvey scheme embarked upon a complementary programme of R & D within a UK context, again with 50% of the industrial costs met by public funds.

ICL was involved from the beginning in both of these ventures, and (together with STC) eventually participated in some 80 Alvey projects and 53 ESPRIT from 1984 to date. ICL also takes part in the CEC and EFTA-wide EUREKA scheme, in RACE (the CEC Telecommunications programme) and in the UK Government's Advanced Technology Programme.

It interesting now to remember the doubts that were expressed six years ago about the practicality of researchers from different organisations and different countries working together. Cultural differences, the language barrier, the 'not invented here' factor would, it was said, frustrate the best intentions of the schemes' planners. Six years on one can assert that the programmes have forged a true community of interest and mutual respect between the thousands of researchers from the community countries who have been involved in the projects.

The results achieved in the collaborations have considerably strengthened the base of IT technology available in Europe and in addition greatly furthered the creation and adoption of European and international standards which are vital to the successful interworking of the multi-vendor computer systems which are commonplace today.

ICL is proud to be playing a central role in this great effort and I am delighted that the Technical Journal is giving space to describe a small selection of the many successful projects in which we have been or are still involved.

V.V. Pasquali
Manager, External Technical Relations.

The Development of Marketing to Design: The Incorporation of Human Factors into Product Specification and Design

Dr. A.T.F. Hutt
ICL Systems Engineering, Systems Integration Division, Reading, UK
Fiona Flower
ICL CPS Professional Services, Information Technology Services Division, Bristol, UK

Abstract

This paper describes how the research from three Alvey projects has been exploited to produce ICL's *Marketing to Design* methodology. This methodology incorporates human factors into product specification and design, enabling design teams to develop products which are smaller, more tightly focussed and, as a result, more acceptable to their users. ICL currently presents the methodology through a series of five workshops, backed up by workbooks and manuals for use in the workshop and at the workplace. *Marketing to Design* therefore represents one of the most significant practical exploitations to emerge from the research into human factors undertaken through the sponsorship of the Alvey programme.

1 Introduction

The Marketing to Design methodology is a good example of how to achieve industrialisation of a process developed as a result of collaborative research by academic and industrial partners into a specific area of Information Technology.

This paper is in five parts. The first part describes how ICL established a series of three overlapping Alvey projects to develop a methodology for the capture of human factors-based requirements. The second part describes the finished methodology. The third part, which represents the bulk of the paper, describes the technical contributions which the three Alvey projects made to the development of the methodology while the last two parts contain an evaluation of the methodology and a report on the take-up of the methodology.

2 Background

In 1984 ICL identified that the issue of Human-Computer Interaction (HCI) was likely to become a major differentiator between competing Information Technology products. This decision was made at a time when the Alvey programme had also set aside funding to be devoted to research into human factors issues.

ICL accordingly set up an HCI Strategy unit to coordinate research in this area. The unit, headed by Dr. A.T.F. Hutt, began by determining the scope of the HCI problem, as illustrated in Figure 1.

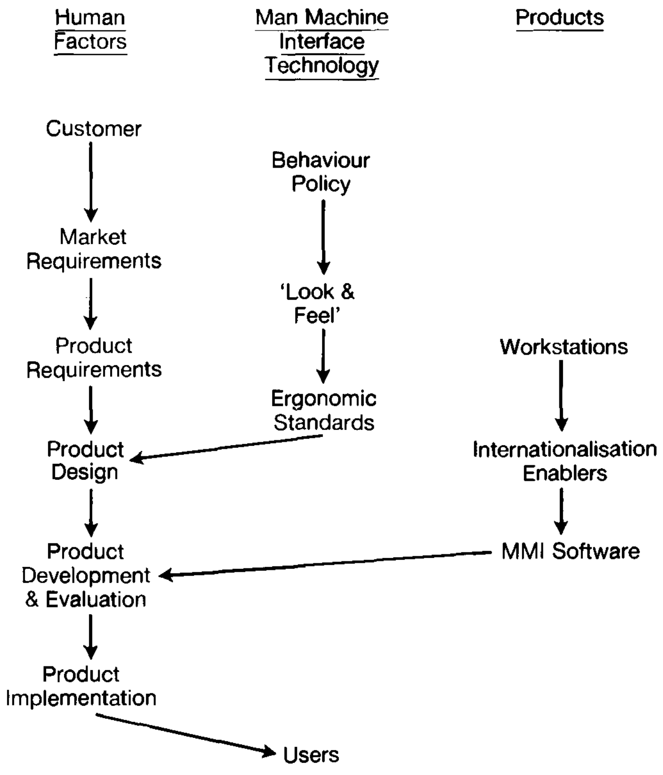


Fig. 1 The shape of ICL's HCI strategy in 1984

As the diagram illustrates, the preliminary study identified three basic starting points for the HCI Strategy:

- Human factors issues relating to the development process;
- The development of Man Machine Interface (MMI) technology;
- Products delivered by ICL which would need to be changed to incorporate the new MMI technology and to make them more usable.

The first and most important of these was considered to be the need to understand the product development process as a whole, and to identify the human factors issues which are relevant at each stage. The team decided to begin by studying the area of requirements capture. The reason for this choice arose from a feeling, confirmed with discussion with some of ICL's collaborators, that errors in usability arose during the requirements capture phase of a product development and that beyond that point a development team were unable to rectify these basic usability errors.

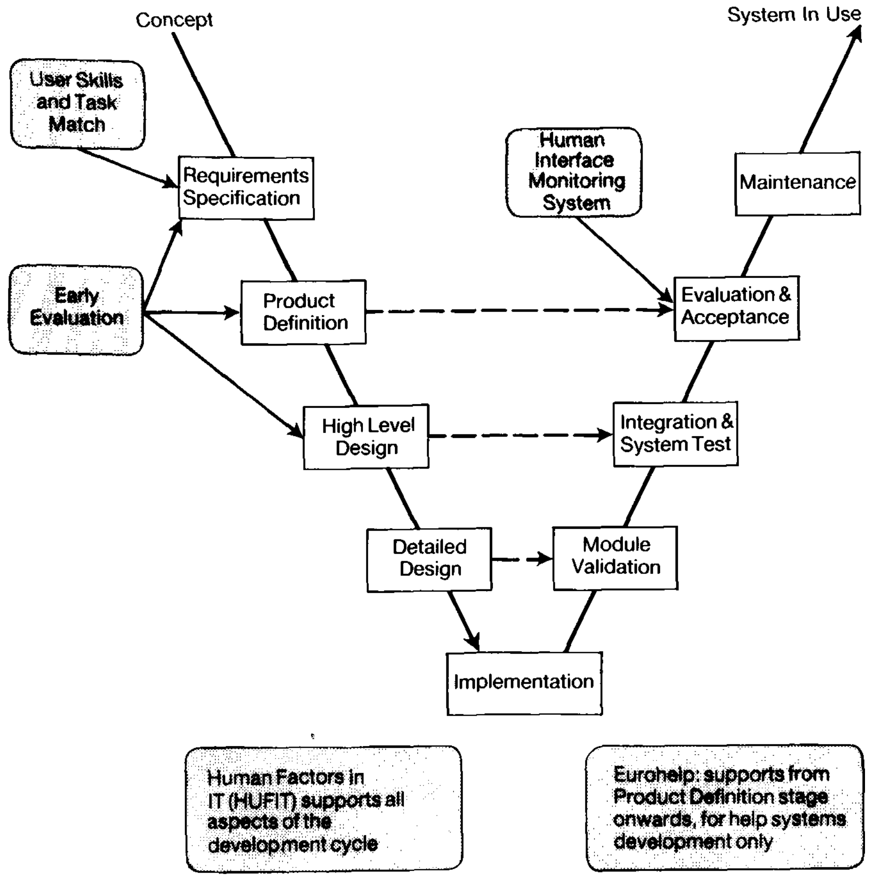
The problem of poor requirements specification was perceived as being particularly acute for large IT companies. This applied especially to those involved in producing generic or off-the-shelf software packages, whose products have to satisfy a wide and variable range of user requirements. In the absence of precise and unambiguous requirements specification, it was felt that suppliers tend to design and market products which have either too many, or inappropriate functions. In consequence, production costs are often higher than need be, and customers are sold products which do not actually meet their needs. In addition, from a user's point of view, a poorly specified product may have the incorrect level of functionality, and may prove difficult to use. The team concluded that there was a need for a process or methodology which provides a rigorously defined set of product requirements for a well-defined market segment.

Another starting point for the HCI Strategy was concerned with the development of MMI technology which included the development of window systems, help systems and user interface management systems. ICL had a number of Alvey and Esprit projects in this area; some of these are described in other papers in this issue of the journal.

The final starting point was concerned with changing ICL's existing and new products to make them more usable. This was seen as an important part of the Strategy but it was perceived that success in this area was dependent on gaining a better understanding of the user's needs which in turn led to greater emphasis on the requirements capture process.

As a basis for research, the development cycle was presented as a 'V-diagram', based on the standard software engineering cycle, and adjusted to fit in with ICL's Phase Review process. Three separate Alvey projects were then proposed to research human factors issues in areas which were identified as having inadequate or ill-defined processes for specifying or evaluating product requirements. The three areas targeted and the projects associated with them were:

Target Area	*Projects*
Requirement capture	User Skills and Task Match
High level design	Early Evaluation
Product evaluation	Human Interface Monitoring System



Concept
System In Use
User Skills and Task Match
Requirements Specification
Human Interface Monitoring System
Maintenance
Early Evaluation
Product Definition
Evaluation & Acceptance
High Level Design
Integration & System Test
Detailed Design
Module Validation
Implementation
Human Factors in IT (HUFIT) supports all aspects of the development cycle
Eurohelp: supports from Product Definition stage onwards, for help systems development only

Fig. 2 Positioning of Alvey and ESPRIT projects against the Product Development Cycle

Figure 2 positions these 3 projects against the product development cycle. it also shows that these projects were supported by two Esprit projects: Human Factors in IT (usually called HUFIT) and the Eurohelp project.

3 The Marketing to Design methodology

Given that this was the plan in 1984, 1990 finds ICL with the Marketing to Design methodology which is a structured framework for incorporating human factors into product specification and design.

Within this context, human factors may be defined as the organisational and human aspects of an IT system. To examine a product's design from a human factors point of view therefore, it is necessary to consider the effects, good and bad, that the product will have on the users' working environment, and the kind of support that will be needed to enable them to learn how to use it effectively.

Assimilating human factors knowledge into the requirements and design process is not a new idea (see for example Eason [1982]). However, the Marketing to Design methodology provides a structure and process which helps to ensure that only **relevant** human factors knowledge is selected and recorded. This process helps design teams to focus on the key human factors issues, and is intended to prevent expensive 'data trawling' exercises.

Marketing to Design has been developed to fit in with ICL's other standard processes: the Marketing method, Product Development Cycle and Investment Management process. The methodology is delivered through a series of five workshops, led by facilitators experienced in applying it to product development. The positioning of each these workshops in the development cycle is shown in Fig. 3.

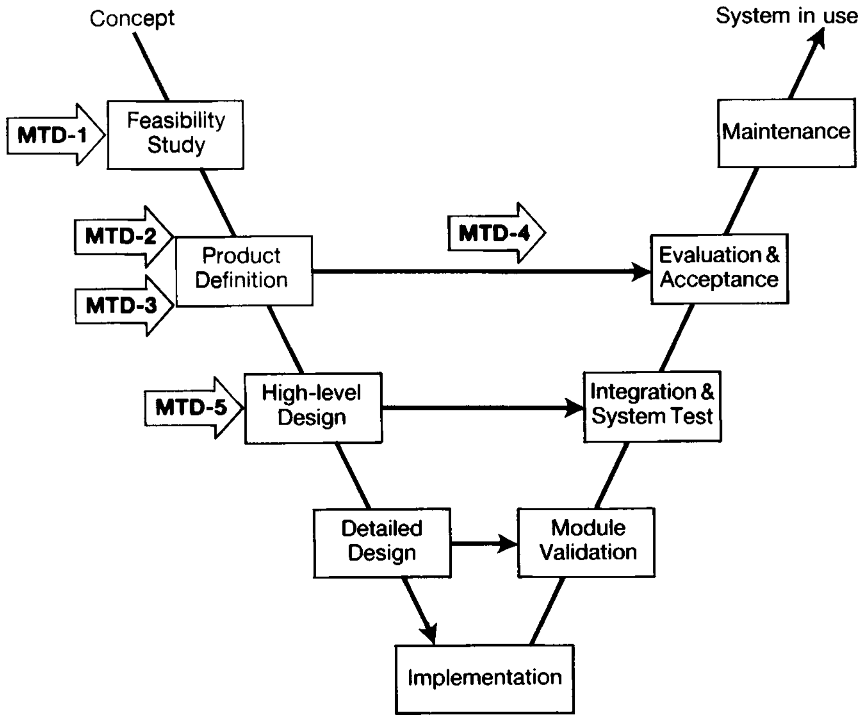


Fig. 3 How the MTD workshops fit into the product development cycle

The first of the five stages is entitled *MTD-1: Describing a Product Opportunity*. In this workshop, multi-disciplinary syndicates are encouraged to explore and describe the critical characteristics of users and their environment both **before** and **after** the product has been installed. Syndicates analyse the proposed users of the product, and any other people who will be affected by it, to collect information on four major issues:

- Work-groups: How users work together in groups
- Users: What they are like as individuals
- Objects: The things they use and produce
- Tasks: What tasks are involved in these jobs

The information collected is documented formally as a *Human Factors Description*, which serves as basic input for the workshops which follow.

The second part of the methodology is entitled *MTD-2: Identifying a High Value Solution*. This workshop provides syndicates with a user-oriented method to produce a significant part of the high level design for their product. MTD-2 addresses a number of issues fundamental to product design. In particular, it focuses on identifying the user roles that the product supports, and on defining accurate task, process and object models. The allocation of tasks between man and the machine is also considered, as are the social environments in which user roles will be performed. It is also considered important at this stage to establish the usability and acceptability goals which the product has to meet before it can be released. This information is documented in the early parts of the Product Requirements Document.

The third stage has been entitled *MTD-3: Delivering a Business Solution*. In this workshop, product syndicates identify the distribution channels for their product, and establish the services, training and documentation required to deliver it via one of these channels. The information collected is formally documented as the remaining parts of the Product Requirements Document. MTD-3 tackles a number of fundamental issues relating to the non-functional aspects of the product. In particular, it examines the learning environment in which users are expected to learn about the product, and defines a teaching model to identify the key facts users have to learn to enable them to get started using it, to become proficient in its use, and to be able to exploit all the facilities it offers. The workshop also helps syndicates to define plans for services, training and documentation that will ensure that the teaching model is delivered in a cost effective way, and that all classes of users are provided with the information they require.

The fourth part of the methodology, entited *MTD-4: Testing Usability and Acceptability* has two main objectives: to establish a development plan which allows time and effort for usability and acceptability testing, and to produce a definition of the first usability or acceptability test. The test is described in the form of a report that outlines the most appropriate method for testing, taking into account factors such as the state of the product, and the available budget, staff and resources.

The fifth and final part of Marketing to Design, *MTD-5: Designing the Man Machine Interface* uses task modelling and planning languages as techniques and tools to produce complete and unambiguous high level models of the functionality provided by the product. The workshop enables the product syndicate to link the task model to the MMI objects by specifying a dialogue model. Low level MMI issues are also discussed, to enable the syndicate to pinpoint major areas of concern, and to discuss alternative solutions.

Each of the MTD workshops is attended by a maximum of six people, made up from a mix of marketeers, representatives from the product design team, services consultants, technical authors, training consultants and project managers who are working together on a development project. A further requirement is for 'typical' users of the product to visit the workshop, to test the validity of the assumptions syndicates have made about users and their jobs.

4 How Marketing to Design was developed

The objective of this section is to describe the major contributions of each of the Alvey projects to the development of Marketing to Design.

4.1 The User Skills and Task Match project (ALV/PRJ/MMI/143)

Within the context of Marketing to Design, the User Skills and Task Match project was mainly concerned with user oriented issues. This work resulted in several important insights which have governed the methodology.

Understanding the relationships between the users and their IT system
One of the earliest findings was the importance of work system theory which describes the relationship between an IT system and the organisation using that system. Figure 4, shows that the work system is made up of two parts, the social system and the technical system:

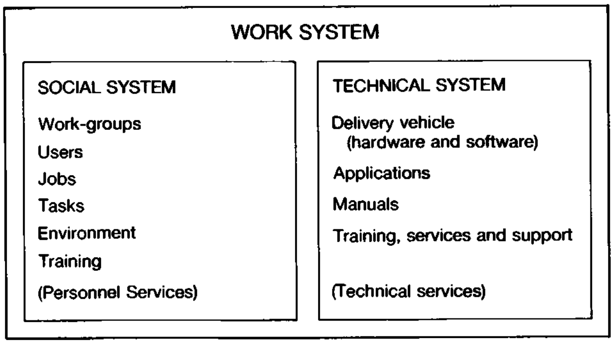


Fig. 4 The Work System

The social system is everything to do with the way users think and behave, how they work together in groups, and the material things that support them. It includes users themselves (their skills, motivation, knowledge and personal concerns); the jobs these users carry out; and the individual tasks embodied within these jobs. It also includes the environment within which the users work, the procedures they follow (manuals and methods of working), and the training and support they need to do their jobs.

The technical system is the IT product response to the needs of the social system. It is, in other words, the whole of the computer system – the hardware and software; the application (including data storage, logic and the man-machine interface); the manuals; and the training, services and support needed for users to be able to understand and exploit the technical system.

The work system is the combination of the social and technical systems; in other words; the overall view of the working environment.

This theoretical foundation is particularly important to an IT supplier because it helps state the relationship with the user and the customer.

Understanding how to quantify how IT systems impact users
Understanding the relationship between users and their IT system shows that if you change the IT system then this will cause some change in the social system and this implies that users will have to do something different.

This is a fine theoretical statement but it is of very little value to a product development team; they need to be able to quantify the change that an IT system is going to bring to users and to assess the user's reaction.

One of the major outputs from the User Skills and Task Match Project was a set of techniques for measuring and assessing these changes as part of the requirements capture process. These techniques look at these changes in terms of the changes to:-

- the workgroups involved with the product:
- the users involved with the product
- the objects handled by the workgroup and the users
- the tasks performed by the workgroup and the users

These changes are measured in terms of changes to specific issues which research has shown to be significant.

For example if you wish to measure the impact of IT on an individual when considering his or her own point of view you need to consider issues such as: Attitude, Motivation, Aspirations/Ambition, Expertise and Skill.

If your wish to measure the impact of IT on an individual when looked at from the point of view of the organisation which is employing them you need to consider issues such as: Mission & Objectives, Importance, Replaceability, and Power.

One of the outputs from the User Skills and Task Match project was that it identified where information of this type collected early in the requirements capture process should be fed into design decisions taken later in the development cycle.

IT systems support peoples' work
Another outcome of the User Skills and Task Match project was set of models which allow a development team to design products which will support users in their work. These models are based on three concepts.

The first concept is that of a user role: which is defined as the part that a person plays in an organisation. The operationalisation of this concept ensures that ICL can develop products (eg. combinations of hardware, software, services, training and documentation) which enable a specific class of user to carry out a specific set of tasks with known learning costs.

The second concept is that of a task model: which provides a top down definition of all the tasks that are to be associated with a particular user role.

The third concept is that of allocation of function which allows a design team to explicitly decide whether a task in the task model is to be allocated (eg to be performed by) to the computer system, to the user or to be shared between them.

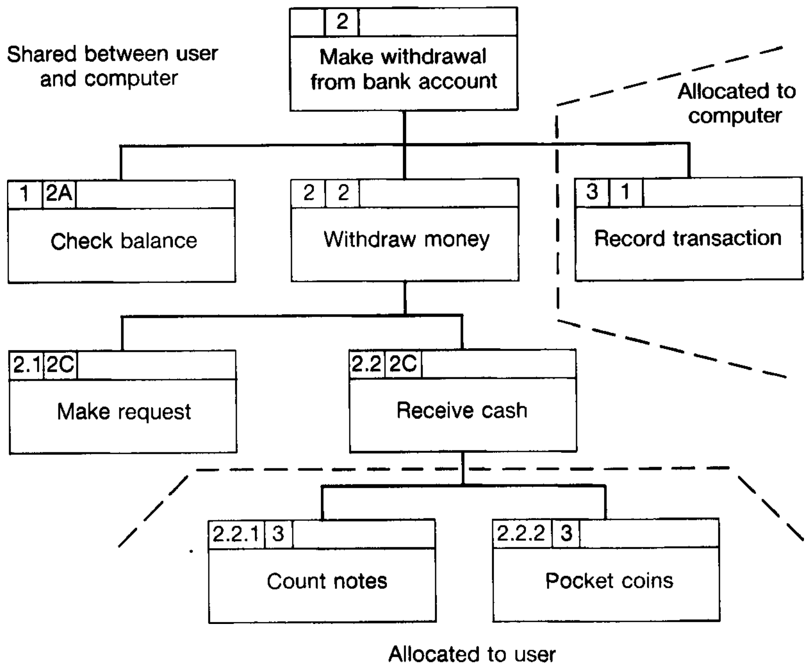


Fig. 5 A Task Model showing allocation of function

To illustrate the use of these concepts, Fig. 5 contains an example of the task model for the user role of bank teller machine user and shows the allocation of functions between the user and the computer.

Understanding how people learn
The last major contribution made by the User Skills and Task Match project was concerned with the model for teaching and learning; it is illustrated in Fig. 6.

The key messages associated with this diagram is that with any product people go through 4 phases of learning which in ICL have colloquially been called:-

- Awareness: concerned with users understanding that the product exists and gaining an understanding of how it will help them in their work
- Getting you started: concerned with helping the user through their first encounter with the product and becoming familiar with its capabilities
- Keeping you going: concerned with users being able to handle error and exceptions encountered while using the product
- Taking you further: concerned with becoming an expert in the use of the product

Further to this there is a certain level of knowledge and skill which can be taught by the product supplier and a level of knowledge and skills that users have to develop for themselves. The first of these is predominantly concerned with product knowledge (eg it is about the technical system) and the second is predominantly about the use of the product (eg. about the social system).

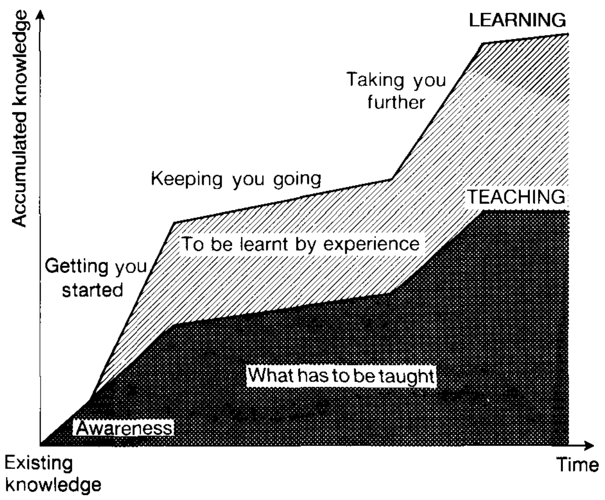


Fig. 6 The Learning and Teaching Model

4.2 The Early Evaluation project (ALV/PRJ/MMI/122)

The second Alvey project to contribute heavily to Marketing to Design was the Early Evaluation project.

The main thrust of the project's research was to identify and specify techniques for evaluating user interfaces to computer-based applications systems at early stages in the design process. The objective was to specify techniques to enable design teams to make a choice between designs for man-machine interfaces *prior* to their implementation in operational prototypes. This would significantly reduce investment in the implementation of designs that later proved inadequate.

From the point of view of the Marketing to Design methodology, the major results from this project were:-

The Positioning Human Factors in the Product Development Cycle
One of the deliverables of the Early Evaluation project was to develop a human factors view of the product development cycle. This understanding results in the positioning of the various parts of the Marketing to Design methodology as described in section 3 and illustrated in Figure 3.

Standards for Task, Process and Object Models
One of the issues which needs to be addressed when discussing a user comprehension of an IT system is their understanding of the system. The project showed that from a designer's point of view this understanding is dependent on three different type of models:

Process models: which show how tasks are passed between the different user roles, and how they are sequenced as time progresses. They are also useful to illustrate the flow of control, objects and resources.

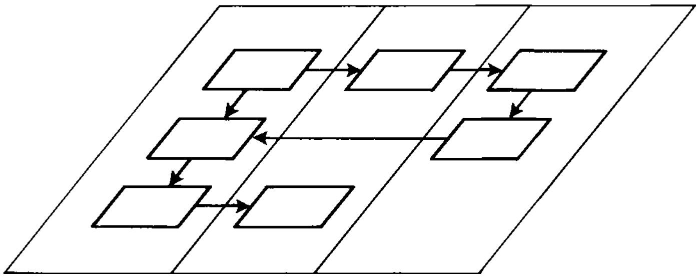


Fig. 7

Task models: which show the hierarchical decomposition of tasks, and are useful to establish the level of detail to which the design of a product should be taken:

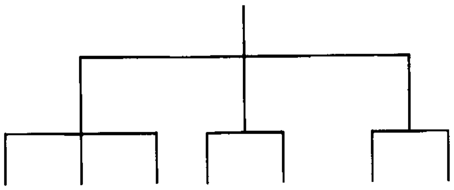


Fig. 8

Object models: which span both the process and task models, and are used to model objects which are shared between user roles, or between tasks. They may be used to illustrate either the object classes, or the whole-part relationships between objects in which the user is interested:

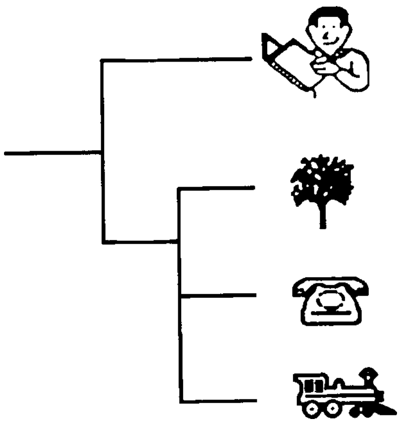


Fig. 9

The Early Evaluation project completed these models by defining a process for their development and standards for their representation.

Usability

The ISO definition of usability is: the degree to which specified users can achieve specified goals in a particular environment with effectiveness, efficiency and satisfaction.

Effectiveness: a measure of accuracy and completeness of the goals achieved

Efficiency: a measure of the resources (eg. time, money and human effort) used to achieve the specified goals

Satisfaction: a measure of the physical comfort and subjective acceptability of the product to its users and other people affected by its use.

The Early Evaluation project was mainly concerned with the development of techniques for measuring usability and in fact identified three sets of techniques:-

Knowledge elicitation techniques: originally proposed as a method of collecting information for the design of knowledge based systems but on this project were used as an evaluation technique (eg. collecting information after the design had been done to check it).

Specification evaluation techniques: aimed at ensuring that the specifications identified a system which would meet user needs. Needless to say these techniques were mainly concerned with verifying the validity of process, task and object models.

Prototype and Product Evaluation techniques: concerned with the evaluation of functional prototypes.

MMI Design Models
Finally the fact that evaluation techniques involved testing a man machine interface led to the position where the Early Evaluation project produced models of the Man machine interface. These were later used as part of the MTD-5 Designing the Man Machine Interface workshop.

4.3 The Human Interface Monitoring System project (ALV/MMI/PRJ/091)

The third major project to contribute towards Marketing to Design was the Human Interface Monitoring System project which delivered a prototype Human Interface Monitoring System comprising four components:

- A methodology that explains how to go about assessing the need for a usability test, and how to produce a test design.
- A capture/replay sub-system that provides the hardware for capturing a user session, and allows the session to be replayed for subsequent analysis.
- A portable workbench that may be used to control the capture/replay subsystem, and to perform simple analysis of test results.
- A laboratory workbench which provides IT support for the duration of a usability test (including management of the test process; capture/replay of the users' sessions using the capture/replay sub-system; and a full sound and video recording system and event log for the human factors consultant). The workbench also provides analysis of the test results to enable deductions to be made about the usability of the system under test.

The results of this project benefited Marketing to Design by providing another part of the methodology and a workbench to support usability trials. The insights embodied in this work are:-

Usability trials are important but difficult to do well
Usability trials are important because they help to confirm that the development team has developed a products which meets its usability goals. There are however two main problems with usability trials.

- they provide feedback which may be unfavourable and hence may need to be addressed.
- they are difficult to do.

Planning for usability trial feedback
The problem of handling feedback is essentially a project planning problem. If a project is planning to hold a usability trial then it needs to decide whether it is going to handle the resulting feedback as part of this product release or the next. Further to this, if it intends to handle the feedback in this release then a good deal of care is required in the planning of the product development and the trial.

The Human Interface Monitoring System project with some help from staff working on the Esprit "Human Factors in IT" project developed a framework and process for addressing these issues.

Carrying out a usability trial
A small usability trial requires you to test 5 or 6 subjects carrying out a prescribed set of tasks using either a prototype or fully working version of a product. Typically it takes about 2/3 weeks to prepare, a week to carry out and 2/3 weeks to analyse the results and produce a meaningful report.

One of the outputs from the Human Interface Monitoring System project was a process for doing all this, a set of tools and techniques for reducing the time and effort involved.

This work has now been incorporated into Marketing to Design workshop 4: Testing Usability and Acceptability.

5 Evaluation of Marketing To Design

From the outset, the User Skills and Task Match project had a programme of work to evaluate Marketing to Design. The initial evaluation was conducted on three levels:

- To assess the courseware, and evaluate the impact of the methodology on design custom and practice. This evaluation was conducted by independent researchers, who interviewed the first ten product syndicates to attend the MTD-1 workshop.
- To perform a retrospective analysis of an existing product (by applying the MTD methodology to the product requirements document for that product). This evaluation was performed on a non-ICL product requirements document, outside the Alvey project.

- To perform an overall evaluation of the effect of the methodology by measuring the use made of products in the field.

Overall, the results of this initial evaluation supported the project team's belief that the methodology was usable and valuable, and the workshop environment helped to improve the communication and understanding between the designers, marketeers, authors and consultants involved. The study also showed that the workshop was necessary because when the take-up of products by users was measured a variety of usability faults were found, many of which would have been resolved by the use of the complete methodology.

The major finding from the evaluation was that the ultimate success of the Marketing to Design programme would depend on active support from ICL management. The project concluded that more effort was required to communicate the benefits of the workshops to ICL's project managers and senior management.

6 On-going use and development of Marketing To Design

The Marketing to Design methodology is already in widespread use within ICL (65 projects have used the parts of the method in the period 1986 to 1989), and is recommended as part of the normal process for developing new IT products. A number of major external customers have also viewed it with some interest.

The methodology is still under development, and much work remains to be done, particularly in the later stages covering usability and acceptability testing and the design of the man machine interface. Further to this, work is now in hand to develop of tools to support it. ICL has also put forward proposals to the IED programme (the successor to Alvey) to progress the development of tools. These proposals – for the Co-operative Requirements Capture project and the Organisational Requirements for IT Systems (ORIS) project – continue the collaborations established so successfully in the Alvey programme.

7 Conclusions

The Marketing to Design methodology represents a significant practical exploitation of the research contributed by the Alvey projects discussed in this paper. Without this research, the programme may never have been conceived, and would certainly not have got off the ground. However, it also underlines the power of industrial muscle: without ICL's unflagging determination to develop and deliver the Marketing to Design workshops, the research from these projects may never have been pooled to produce a single coherent methodology. Moreover, by using the people involved in the research actually to deliver the workshops, the industrialisation period was shortened to less than one year for each of the stages in the process, as opposed to the norm of five to seven years. In short, Marketing to Design has proved that the drive and clarity of purpose of an industrial partner can vastly accelerate the industrialisation of a process developed from research into a specific area of Information Technology.

Acknowledgements

We would like to take this opportunity to thank the many people who have contributed towards the development of the Marketing to Design programme.

In the first instance, we would like to acknowledge the efforts and continued support from all those involved in the User Skills and Task Match project and in particular Linda Macaulay (now at UMIST), Dr. Chris Fowler and Mark Kirby (Huddersfield Polytechnic), and Dr. Ken Eason and Dr. Susan Harker (Loughborough University).

We owe also a great debt of thanks to our co-contractors on the two other major projects whose research has been incorporated into Marketing to Design. To this end, we would like to thank the three teams working on the Early Evaluation project: the team from Plessey Research & Technology, headed by Simon Hakiel; the team from University College, University of London, headed by Dr. John Long; and the team from Queen Mary College, University of London, headed by Dr. Peter Johnson. We would also like to acknowledge the valuable contributions made by the two teams working on the Human Interface Monitoring System project: the HUSAT team headed by Professor Brian Shackel and the team from UMIST, headed by Professor Derek Morris.

In addition, we would like to thank the groups working on other projects which have made valuable contributions to the programme: to Mick Smith of ICL and Professor Roger Hartley from Leeds University for their contributions from the Eurohelp project; and to the HUFIT project team, and in particular its managers Margy Galer from HUSAT and Fred Russell (ICL).

Finally, we would like to acknowledge the help and encouragement we have received from staff at the Alvey Directorate: Dr. Mike Underwood and Dr. Mike Tainsh who have supported this work throughout; Professor Tom Singleton for his advice as project monitor on both the User Skills and Task Match and Human Interface Monitoring System projects; and Dr. Virgilio Pasquali for first estalishing the HCI strategy.

Last but not least, we would like to thank the many ICL product syndicates who have taken part in the MTD programme, and who have contributed their ideas and suggestions for improvement.

References

The documents referenced in this section include the final reports to the Alvey Directorate for each of the major projects which have contributed to Marketing to Design. Each document contains a reference section, which itemises the documents and papers which are of relevance to that particular topic.

1 User Skills and Task Match Methodology, Final Report April 1988 (ALV/MMI/PRJ/143). International Computers Limited and Huddersfield Polytechnic. Available from the Information Management and HCI Technical Strategies unit (ICL), or the HCI Research Unit, Huddersfield Polytechnic.
2 Development of Methods for Early Evaluation of Interface Designs, Final Report, December 1989 (ALV/MMI/PRJ/122). Plessey Research & Technology Roke Manor Ltd, International Computers Limited, Queen Mary College (University of London), University College (University of London). Available from the Information Management and HCI Technical Strategies unit, ICL, Reading, Berks.
3 Human Interface Monitoring System, Final Report, October 1989 (ALV/MMI/PRJ/091). International Computers Limited, University of Manchester Institute of Science & Technology, HUSAT Research Centre. Available from the Information Management and HCI Technical Strategies unit, ICL.
4 IPSE 2.5 Final Report. International Computers Limited, Plessey Research & Technology Roke Manor Ltd, STC Technology Ltd.
5 HUFIT Final Report. International Computers Limited, HUSAT Research Centre, Loughborough University. Available from HUSAT Research Centre, Loughborough University.
6 Eurohelp Final Report. International Computers Limited, Leeds University.
7 HUTT, A.T.F. et al. Marketing to Design (brochure). Available from ICL Internal Staff Training.
8 HUTT, A.T.F. et al. Understanding Users (overview). Available from ICL Internal Staff Training.
9 HUTT, A.T.F. et al. Describing a Product Opportunity (manual). Internal Training manual, available from ICL Internal Staff Training.
10 HUTT, A.T.F. et al. Identifying a High Value Solution (manual). Internal training manual, available from ICL Internal Staff Training.
11 HUTT, A.T.F. et al. Delivering a Business Solution (draft manual). Internal training manual, available from the Information Management HCI Technical Strategies unit, ICL.

The Manchester University Multimedia Information Systems Project

Foreword

The demand to store and access information consisting of an integrated mix of such data types as image, text, graphics, voice and computable data is giving rise to very large and complex databases. The combination of database design and image processing skills available at the University of Manchester, together with the commercial and technical guidance provided by ICL, enables the development of the necessary innovative techniques to manage such multimedia databases.

This research project in multimedia database systems is a collaboration between ICL's OFFICEPOWER Applications Product Centre and the Electrical Engineering and Computer Science Departments of Manchester University. The three year programme of work which will run to mid 1992 involves 14 University research staff, some funded by SERC (Science and Engineering Research Council), and the remainder by the University. ICL is providing joint project management, technical consultancy and equipment.

The benefits from this collaborative project to ICL are significant. It is the Company's aim to be a leading supplier of multimedia systems and this research programme will accelerate the introduction of ICL multimedia products to market. This is against a background where international competition will increase fiercely in the relatively near future.

The major challenge to ICL, which has a history over the last thirty years of successful collaborations with Manchester University, will be to establish the necessary exploitation mechanisms in order to derive the maximum benefit from the research programme.

J.A. Nelson
Manager, Collaborations
OFFICEPOWER Applications Product Centre, ICL, Bracknell, Berks, UK.

Advances in the Processing and Management of Multimedia Information

*M.H. O'Docherty †P.J. Crowther *C.N. Daskalakis †C.A. Goble *M.A. Ireton †S. Kay and *C.S. Xydeas
Victoria University of Manchester

Abstract

Current database systems are efficient at handling simple data such as numerical or textual attributes. They are not well suited to other media except as filing systems using manually generated labels. Conversely, a Multimedia Information System is intended to handle all types of electronically modelled data in a uniform fashion. Examples of multimedia data are text, images and voice. Many information processing applications are multimedia in nature but the technologies needed to implement multimedia systems for such applications have only recently begun to appear.

This paper first examines those features that an ideal Multimedia Information System needs to possess. These include an object-oriented environment to handle multimedia data, the retrieval of all data by content using semantic analysis techniques which can be adapted to each new application, a suitable model of object oriented data such as ODA, a carefully designed query strategy with confidence levels allowing inexact matching, and finally comprehensive browsing and editing facilities.

Prototypes which go some way towards satisfying these goals are described. The limitations of these are specified and this is followed by a description of the prototype system being developed at Manchester University which has the aim of satisfying the requirements of the ideal system.

1 Introduction

Computer-based Multimedia Information Systems (MMISs) have been a popular research topic in recent years for two reasons. First, because the technologies needed to implement MMIS's are only now becoming widely available. The main technological requirements are optical disks for mass

*Electrical Engineering Dept. †Computer Science Dept.

storage, high bandwidth networks for fast access and sophisticated workstations for meaningful presentation.

Secondly, many applications are multimedia in nature or would be greatly enhanced by multimedia techniques. Some of these applications are:

Education Presentations typically contain words, pictures, video and voice; books contain words and pictures, but there is no reason why an 'electronic book' should not also contain narrations and animations

Offices Information may arrive in an office in many different forms, each being processed and filed in a different manner. For example, documents, memos, telephone messages, faxes. Since the output of offices is moving towards 'electronic publishing', why not collect and process the information in the same way?

Medical Records A patient's medical record may consist of case histories, X-rays, notes or sketches from consultations, test results, etc. In addition, several new imaging techniques generate digital or video data directly – computer tomography, magnetic resonance and ultrasound.

Libraries and Museums These contain vast amounts of data from literary works to archaeological artifacts accessed by manually generated indexes.

Computer Aided Design (CAD) For example blueprnts, simulation, or interactive design with 3D rotation and scaling of objects.

Geographic Databases Maps, satellite images, demographics, even tourist information.

The above applications generate five main types of data, all of which can be represented directly or indirectly by an MMIS.

1 **Facts** Author, date, unique identifier – the types conveniently handled by existing databases.
2 **Statistics** Data that requires graphical, (eg. pie-chart or bar-chart), or tabular, (eg. spread-sheet), representation.
3 **Text** As may be contained in a book or magazine article.
4 **Sound** Notably the human voice – narrations, comments, lectures – but also music, sounds from nature.
5 **Images**
   - Raster: Derived from photographs, painting programs, video-discs.
   - Vector: Line drawings derived from engineering drawings, CAD systems, vector graphics.
   - Moving: A sequence of the above with extra clues as to three-dimensional relationships, history, narrative. Examples are animation or (digitised) video.

However, traditional database management systems (DBMSs – network, hierarchical, relational) are designed for facts only [Date, 1986]. This makes them efficient at handling large amounts of data in the record-keeping sense while maintaining *data integrity* and controlling *redundancy.* Maintaining integrity involves checking that new facts added to the database do not conflict with data already there. Data is redundant if it is repeated or may be derived from other data. Redundancy should be controlled not only to save storage but also because if redundant data is changed, it is not always obvious what other data is affected. In practice, redundancy is carefully re-introduced to improve query efficiency. In a well organised database, much of the management is provided by sticking to a well-designed model of the data, or *schema,* that describes inter-dependencies. This schema has to be redesigned for every application.

Unfortunately, real world data is far more complex than simple facts. For instance, a traditional database may be adequate for storing payroll details of employees in a large company, but what about their employment histories, their references, a picture of each one? This may be achieved using existing database techniques, but only if each object – letter, photograph, taped testimonal – is given a reference number and filed elsewhere. In some cases, the database user may be fortunate enough to have access to these objects on-line and queries may relate to meaningful labels, although these will have been entered manually. The major problem is that *traditional databases were not designed for multimedia data.*

A multimedia database will have to manage data differing widely in structure and size – from a text comment a few bytes long to a digitised photograph that may be a half-megabyte array. This is where *object-oriented* techniques become important [Rentsch, 1982]. The object-oriented paradigm is a development of structured programming of the 1960's onwards (Fortran, Cobol, Pascal, Algol et al.) and *data abstraction* of the 1970's (Ada, Euclid, PL/1 et al.). Its use extends beyond programming to the modelling of complex data [ISO, 1988] and the design of large systems, where the high level abstraction leads to rapid prototyping and re-use of code. It is now being applied directly to the storage and retrieval of complex heterogeneous data [Kim and Lochovsky, 1989]. For efficiency of design and homogeneous usage, the same object-oriented techniques should be applied to every aspect of an MMIS, from the user interface to the DBMS.

If a multimedia system is not to degenerate to a mere filing system, albeit an efficient and homogeneous one, it must cater for *content retrieval.* This is the process of retrieving objects according to their *semantics* or true meaning. Since *information* is data along with the semantics of that data, a true Multimedia *Information* System must store both. Most databases leave semantic interpretation of data to the user, the designer of the data model and the operator who enters facts by hand. This is time consuming and unreasonable for large amounts of multimedia data. The system must be taught how to interpret data itself using image analysis, natural language processing, speech recognition and the like. Although none of these disciplines is yet a science, by constraining the application domain and treating the analysis of each medium as a separate problem, automatic content retrieval is possible. This is the principal challenge for multimedia research.

Semantic analysis techniques must be extensible and it must be possible to tailor them to each new application. It is tempting to write ad hoc programs to analyse data from a particular domain, but then all the code must be discarded if the domain changes or inferencing techniques improve. This problem has been most clearly addressed in the field of Artificial Intelligence (AI) [Rich, 1983] and more specifically Knowledge Representation [Ringland and Duce, 1988].

The management of multimedia information is a multi-disciplinary task, not always tackled by a multi-disciplined team. Notably, implementors of MMIS's can learn much from experts in image/voice analysis and knowledge representation as well as from designers of more conventional databases.

A project is underway at Manchester University [Daskalakis et al., 1988] run by The Multimedia Group. The broad aim of the group is to advance the management and retrieval of multimedia data, drawing on expertise in image analysis, databases and AI. A prototype is under development that concentrates on the content retrieval of images in a multimedia framework that allows for the storage of free text, facts and documents (where documents are considered to be a structured hierarchy of other media).

Section 2 of this article is an outline of those facilities that should be provided by an ideal MMIS.

Section 3 is a historical discussion of relevant research and commercial prototypes.

Section 4 describes The Multimedia Group's system in more detail.

Finally, Section 5 contains concluding remarks.

2 What an MMIS should provide

The ideal MMIS will offer facilities far in advance of existing factual databases. The purpose of this section is to give a flavour of those facilities along with comments as to feasibility and technological requirements.

The first requirement is to be able to store all media that can be represented digitally, principally those listed in Section 1. Users should be able to present new instances of any medium for the system to store and retrieve any existing instance for use elsewhere. Users will require hard copies of instances for themselves and other systems and software will require that they are available in a standard interchange format.

It will often be necessary to store composite media, i.e. structured collections of other media as some form of document. The most notable document format is the *Office Document Architecture* [ISO, 1988] developed originally through the European Computer Manufacturers Association. It organises all multimedia data into hierarchical structures of logical and layout objects. Logical objects are chapters, sections and so on, while physical objects refer to pages, captions et al. A mapping is maintained between the two as illustrated in the text document of Fig. 1. Several systems have been based on the idea of a general document structure, (see section 3). The documents are displayed page by page on the workstation screen and the user may look forward and backwards, click on 'buttons' to hear voice annotations

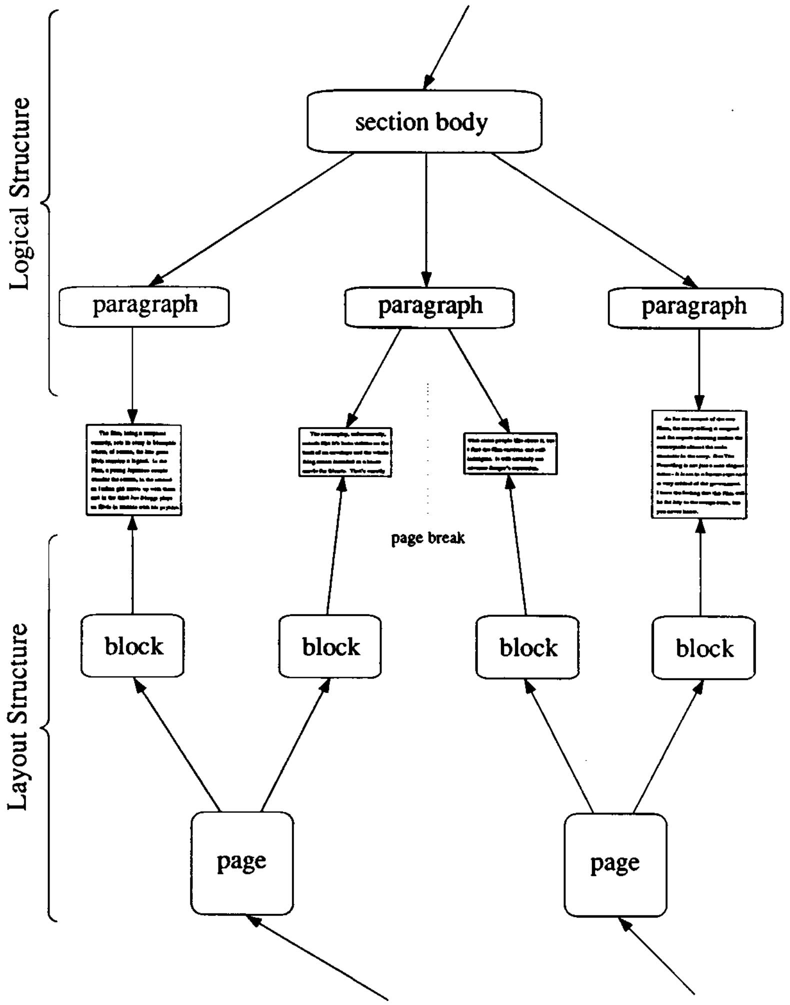


Fig. 1 The correspondence between ODA's logical and layout structures

and see a page build up as transparent overlays are placed on top of a base page.

Once information is stored it must be retrieved using an appropriate query[1] language. In multimedia, retrieval falls into two categories: *by presentation* and *by content*. Retrieval by presentation involves data type and structure. Examples might be "All images that will fit on my screen" or "All documents containing voice comments by Jane Smith". It is frequently applied to composite media where type and structural information is readily available. Retrieval by content is the retrieval of objects according to their semantic content. At its simplest, content retrieval uses manually entered *labels* – descriptions entered by an operator at the keyboard. This is too time consuming to be feasible for large systems hence the motivation for automatic analysis. Manual labels may also suffer from the subjective interpretation of the operator.

If users know what information is stored in relation to each object, e.g. what labels the systems uses, then querying may refer directly to that information. However, querying techniques may be extended to include inexact methods of *sketching*, and *similarity*. A sketch is a 'shorthand' symbolic representation of an object, usually in terms of some other medium. An image sketch may be a line drawing of a house with multi-paned windows to indicate that the user is interested in real images of houses with Georgian windows; they will usually have to be drawn with standard symbols or by system-generated example, otherwise they will become as difficult to analyse as real images. A voice sketch might be the string "profit" from which the system would find voice comments containing the pattern of sounds that make up the word when it is spoken. Similarity retrieval involves providing the system with an instance and the system finds similar instances from the database of the same medium. This problem has been approached using statistical *similarity measures* for images [Toriwaki et al., 1980] and *signature files* for text documents [Faloutsos and Christodoulakis, 1984] that give an idea of the grouping of words in each document. In general, similarity matching involves analysing the example in the same way as the existing instances in the system and comparing features according to their importance.

The result of any query is usually a set of those instances that satisfy the query – the *query set*. In the case of direct queries all members matched exactly so they can't be ordered except alphabetically, chronologically or some other measure specific to the medium. Inexact querying, however, leads to the notion of *confidence levels*. A confidence level describes how well instances fit the query so that they can be ranked and the best matches presented first to the user. To be fully useful confidence levels should be a single normalised value that is independent of the medium, (for instance a real number in the range [0..1]).

[1]For convenience, the term 'query' in this article refers (loosely) to anything that specifies a subset of the items stored in the MMIS. Examples are natural language commands, SQL, logical expressions or graphical queries.

Query sets may be large so the user should be allowed to *browse* over them. This might be achieved by displaying instances laid out on an imaginary desktop over which the user may 'move' the display. To fit a reasonable number on the screen and to avoid overwhelming the user with too much information, instances should be displayed as icons or miniatures. Icons are pictorial representations of the content of instances while miniatures are the instances themselves greatly reduced in size. The user selects those instances that appear interesting and the system blows them up to full size. Various browsing techniques are examined in [Christodoulakis and Graham, 1988; Irven et al., 1988].

An extension to the idea of browsing is for the user to explicitly connect iconic or miniature instances with labelled arcs as in Intermedia (see Section 3). This allows descriptions and presentations to be prepared for educational or mnemonic purposes. The set of arcs (sometimes referred to as a *web*) is managed by the system as another medium instance.

Finally from the user's point of view, editors will have to be provided so that users can create and alter instances. These should all have the same look and feel and possibly the same high-level editor. For instance a web editor is medium independent but if an icon in the web is selected for editing, then a medium-specific editor pops up. The user's interaction with the system is likely to be complex, so in the same way that webs can stop a user becoming lost in a query set, window environments with multiple overlapping windows are preferable for the user to coordinate the whole interaction.

MMISs have huge storage requirements. For example a 512×512, 24 bit image requires 3/4 Mb of store. Mass storage is now becoming available in the form of 'juke boxes' of read/write optical disks, but some mass storage devices are still write-once. This implies that objects should be retained on magnetic disk during editing and only *archived* when finalised. Users may also wish to keep their own versions of objects with slight changes from the archived original. It will often be more efficient to store the changes to the archived object than to store the new object in its entirety. This raises another issue in multimedia, that of *version control.* Storage requirements can be reduced using data compression techniques that also make the transmission of objects over networks faster. The most effective compression techniques are *lossy* in that they lose as much information from the original as possible without making the result unacceptable to the user.

MMISs will be expensive to build and maintain so they must cater for multi-user access. This is best achieved using a network of displays connected to one or more centralised storage servers. High bandwidth local-area and wide-area networks are needed and these are already available. If the displays are high resolution workstations then the user can see more of the information when he retrieves it and do more with it. Workstations vary in resolution,calling for resolution transformations to be applied to make some objects visible [Ireton and Xydeas, 1990]. If parts of an object cannot be displayed at all, it may be possible to replace them with iconic versions and display the rest. For example, an image on a page of text could be replaced with its bounding box and the label 'image' or its title.

3 Existing prototypes

This section describes past and present systems that contribute to the goal of the ideal MMIS. They can be divided into three categories.

MMISs These are systems designed specifically for the management of multimedia data, sometimes with content retrieval facilities.

Image Databases These are interesting because images are the largest atomic objects that an MMIS will have to handle and the most difficult to analyse for content retrieval.

Image understanding Environments These are systems intended to aid the designer of image analysis techniques.

Image understanding environments incorporate a database of images, AI and image analysis tools and the facility to adjust software parameters and store results. For a history of such environments and some examples, see [Lawton and McConnell, 1988; Americanix AI, 1989; Walter et al., 1987]. The next two sections describe MMIS and image database prototypes in more detail.

3.1 Multimedia information systems

Minos (Toronto/Waterloo Universities [Christodoulakis et al., 1986]) was designed for the creation, editing and archiving of object-oriented documents similar to ODA. These are displayed, a page at a time, on a workstation screen, successive pages being displayed by clicking a mouse button. Pages may also be flicked automatically to allow simple animation, or they may be overlaid as transparencies. All multimedia data types are provided; notably, voice may be linked to a section (comments) or to the turning of a page, (narrations). Minos has been implemented using Sun-3 workstations, Unix and Ethernet. A document is actually a directory hierarchy with one file for each content portion. These are archived on a file server as concatenated, compressed files, along with a 'structure descriptor' containing the details necessary for reconstruction. Another system, Muse (Crete Institute [Gibbs et al., 1987]) was similar to but less complete than Minos.

Esprit project Multos is similar in scope to Minos, with the addition of content retrieval for images [Conti and Rabitti, 1987; Rabitti and Stanchev, 1987] and the 'Conceptual Structure Definition' (CSD) [Bertino et al., 1988c]. Image content retrieval is achieved by the extraction of Attributed Relational Graphs (ARG's) from images and comparison with the model held in a Prolog knowledge base. An ARG is a graph with multiple symbolic attributes on nodes and arcs. This knowledge base has been tested on ARG's generated from graphical primitives as direct input rather than real images, side-stepping the problem of image segmentation in favour of exercising artificial intelligence techniques. Queries are probabilistic, eg. "Find documents containing a picture of a car with certainty >75%". The CSD is similar in style to the logical and layout structures of ODA. It carries semantic data to allow office documents to be classified [Lutz, 1989]. Multos has a distributed architecture, (see Fig. 2), to allow for the huge volume of data and number of users that any future MMIS will need to handle. Members of the project team have also investigated query optimisation [Bertino et al., 1988a; Bertino et al., 1988b].

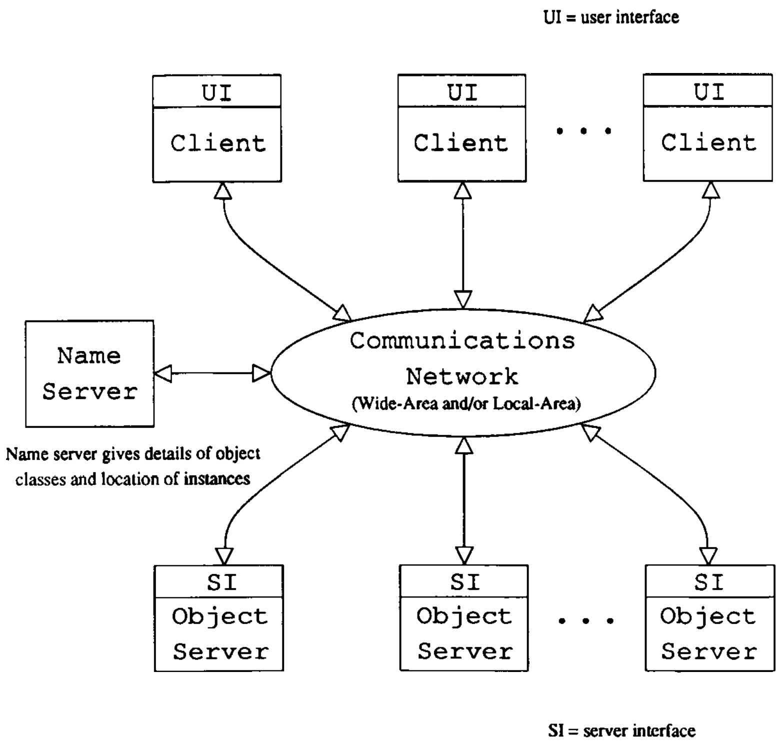


Fig. 2 A distributed architecture

An MMIS has been developed at MCC (Woelk and Kim, 1987) for object-oriented multimedia documents similar to ODA. It caters for the electronic capture, storage and presentation of bit-mapped images and voice using the Orion object-oriented database. The system uses application-specific knowledge based on graphs. Version control is also provided allowing for the ordered manipulation of different formats of or extensions to a particular document.

A prototype 'Telesophy' system has been built at Bellcore [Caplinger, 1987]. The aim of a Telesophy system is "to provide transparent access to all of a community's on-line information". The system, called 'tsl', handles multimedia Information Units (IU's) that may reference or contain other IU's. IU's are manipulated and stored in an object-oriented fashion, using a variety of networked workstations.

The aim of Intermedia (Brown University [Meyrowitz, 1986; Smith and Zdonik, 1987]) is to enhance the learning process in higher education by building on the ideas of hypertext [Conklin, 1987]. 'Documents', (really just atomic high-level objects), are linked into a 'web'. The user sees a plan of the web with labelled arcs and iconic documents and may navigate through it using a mouse, zooming in on interesting documents. Documents may contain text, images or lists of historical events. Hypertext and hypermedia owe their origins to the tendency of humans to perform tasks such as reading or discussion in a parallel fashion – hopping between pages and ideas as necessary. Unfortunately, for a large database of complex objects, it is very easy for a user to become disoriented and data management also becomes very difficult. Therefore, it is suggested [Irven et al., 1988] that webs should only be used for browsing a small number of documents that have been retrieved using other methods. Despite this, Intermedia is still a very useful system for development of educational software using object-oriented tools. It is implemented using Inheritance-C on Sun workstations with webs maintained as Ingres relations.

Apple's 'Interactive Multimedia' [Ambron and Hooper, 1988] is a hybrid system of video-discs, CD-ROM's and other peripherals controlled by a Macintosh running Hypercard, (a hypertext development comprising stacks of windows containing text, images and process control commands). It is intended as an educational tool with the teacher gathering together pieces of information on a particular subject – such as real images from video-disc, voice segments from speech chips – and connecting them so that the pupil can wander through the result. Compact Disc Interactive and Digital Video Interactive are similar commercial standards for management of multimedia data based on micros and video-discs. Another hybrid is Plexus XDP [Tuckwell, 1987] with intended applications in desktop publishing.

3.2 Image databases

REDI (Purdue University [Chang and Fu, 1980]) was designed for Landsat images to allow simple similarity retrieval. Roads, rivers, cities and meadows are identified and stored as relations – eg. roads and rivers are represented by line segments, one per tuple. Queries take the form "Find me the name of the river I just pointed to" or "Find me pictures of cities with a similar road pattern to this one" and are entered in a tabular query language. However, much information is already contained in the image attributes – names of roads and cities, world coordinates, etc. – which makes interpretation easier.

A database of chest X-rays at Nagoya University provides the user with 'sketches' of X-rays it contains. The sketch is a four-level outline of ribs, ribcage and blood vessels produced by image analysis. In addition, radiologists can indicate abnormal patches and provide text comments. As well as mixing textual and graphical annotation, the sketch could be used as a key for similarity retrieval.

A remote sensing image database at Osaka University, Japan [Nagata, 1984], uses five-digit chain codes[2] to encode line segments of rivers, roads, etc. These are stored in a relational database for use in similarity retrieval. Chain codes are also proposed for similarity retrieval in an image database at Michigan University [Grosky and Lu, 1986]. They are concatenated and compressed to yield a string for pattern matching.

Finally, Picture Archiving and Communication Systems (PACS) are beginning to appear. The purpose of PACS is to store large numbers of digital images while minimising access time and allowing standard communication [Rogers et al., 1986; Parrish et al., 1986]. The images are labelled with facts and maybe text comments, for analysis by and communication between experts. An example of their use is in the radiology department of a hospital, where images are produced on film, video and computer – X-ray, ultrasound scan and computer tomography/NMR respectively. At present, X-rays are filed in envelopes and ultrasound scans on videotape, with the obvious potential for loss or damage. Each image type also requires a different display device – light box, TV or monitor – whereas PACS will allow all of them to be integrated electronically.

4 A new MMIS based on content retrieval

All the systems described in Section 3 contribute to the goal of an ideal MMIS but each has serious limitations. MMISs that rely on a single data model constrain the user to thinking in terms of that model. For instance hierarchical document models do not adapt well to the notion of a web of information while webs do not cater for page-by-page presentation; neither documents nor webs are ideal for a pure image database where the main requirement is fast and meaningful access to a large number of images from a single application. Content retrieval is commonly provided as an afterthought rather than being treated as a fundamental requirement. It is impractical to consider that the extraction and organisation of semantic data can be achieved independently of storage and retrieval. Image understanding environments are powerful tools for studying the effects of different analyses of images, but they are intended for use by experts and cannot easily be extended to other media. The system described in the rest of this section is one which was designed to provide automatic content retrieval of a large volume of simple and complex multimedia data. It is not constrained by a single data model nor by application and can be extended to exploit new analysis techniques as they emerge.

[2]Chain codes are concise descriptions of a connected pixel path – each digit representing the movement from one pixel to one of its 8-neighbours.

The aim of the Multimedia Group at Manchester University is to advance the techniques used for the management of multimedia information by drawing together the disciplines of image analysis, database design and artificial intelligence. The architecture of the group's prototype MMIS, (described fully in [Crowther et al., 1989]), is built around two major assumptions.

- That the most powerful facility that the system can provide is automatic content retrieval. This calls for interpreters for each medium since there is little common ground between the analyses of different media. Note that under this scheme a complex object such as a document will use an interpreter for each medium it contains and an interpreter for the document structure.
- That automatic analysis is time consuming, for example some image analysis algorithms can take hours. To expect the system to analyse every object in response to a query and then discard the results is out of the question. As a result all medium interpretation is performed pre-query.

The system architecture is shown in Fig 3. From the user's point of view raw data is stored in the *information base* and can then be retrieved via a graphical query interface.

As data arrives from the outside world it is digitised and put into the raw database. At some convenient time when the system is not busy a *semantic representation* is created for each object by the relevant *medium-specific interpreter*. The semantic representation describes the results of interpretation and is stored in the *semantic representation base*. A mapping is maintained between each semantic representation and its parent instance so that the latter can be retrieved during the query process.

It is sometimes necessary or at least useful to compress medium instances to save storage. Some applications require that this compression is loss-less, (e.g. text), but some are less stringent and permit lossy compression, (e.g. images of the human face). Interpretation of the compressed version is sometimes possible, (e.g. pyramid node linking for images [Pietikainen and Rosenfeld, 1981]), hence the switch between database-interpreter and de-compression-interpreter paths.

As discussed in Section 1, interpretation of medium instances requires knowledge and this knowledge should not be embedded in the code for medium specific interpreters. To this end, the information base contains a third component – the *knowledge base*. The knowledge base contains a model of the data it expects to encounter so that it can guide the interpreter, querying and matching processes. The knowledge can be considered to have two components: *general* and *application-specific*. The long term aim is to minimise the amount of application-specific knowledge so that the system can quickly be tailored to other applications. Unfortunately the process of interpretation varies greatly from one medium to another implying that

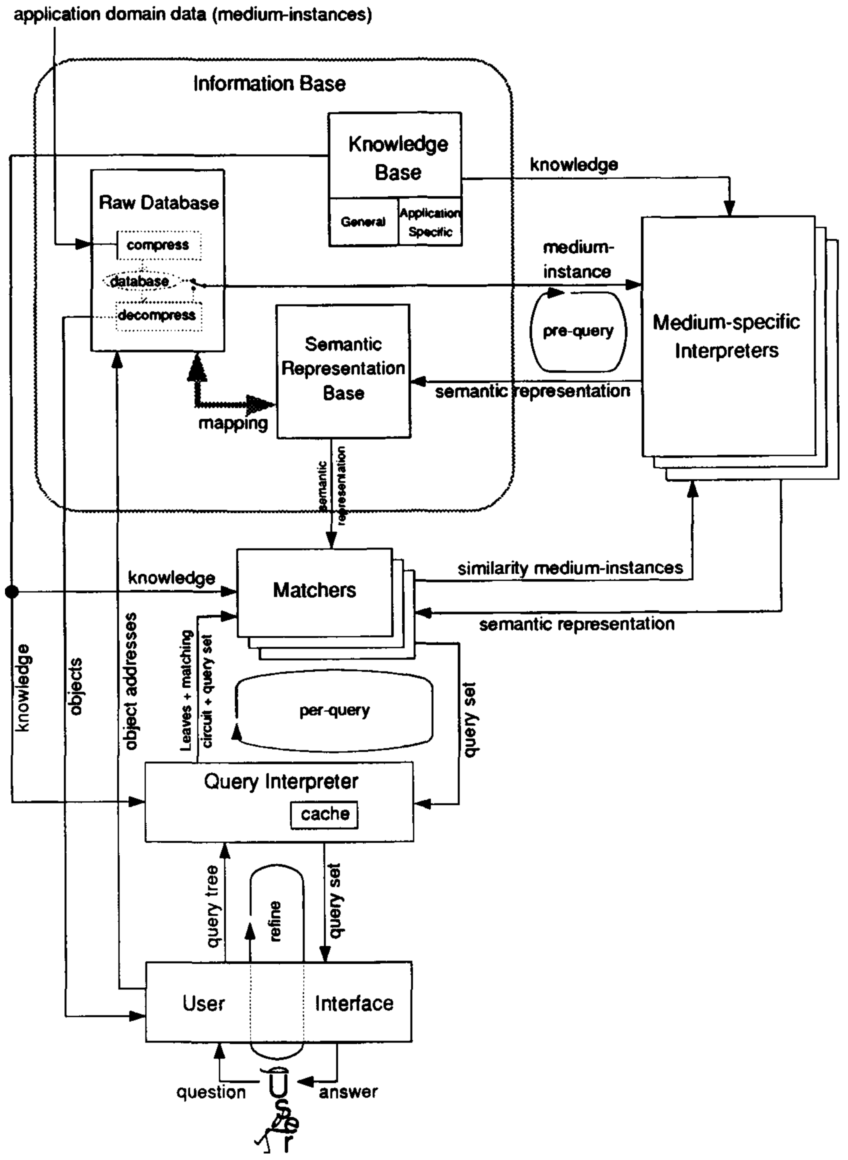


Fig. 3 The Multimedia Group system architecture

application-specific knowledge will always be present, indeed it is necessary even for humans, but it may still be minimised within each medium. Knowledge representation research aims to devise formalisms for describing any knowledge in ways understandable by computer [Ringland and Duce, 1988], so that tailoring future systems to new applications may only involve entering the new knowledge in a standard format.

As an example of a semantic representation consider the *semantic net* of Fig. 4. A semantic net is a graph of objects and unidirectional relationships between those objects used to describe a complex entity, situation or concept. This can be combined with rules of the form "Object A is to the right of object B if object B is to the left of object A" so that further information can be deduced. Semantic nets can be used to answer queries such as "What images have teacups in them?" or "Where is the cup in relation to the jug?".

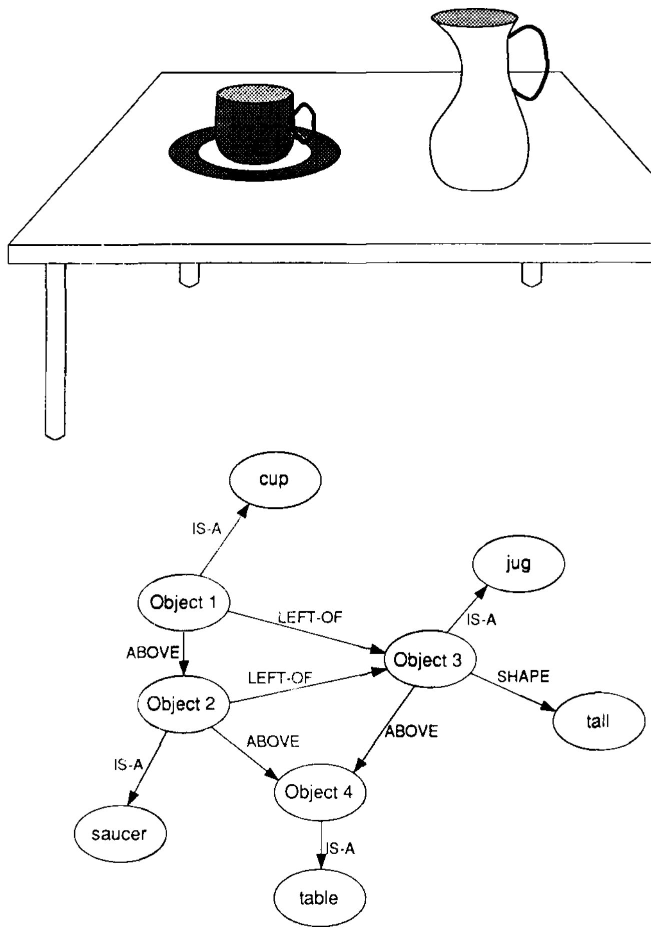


Fig. 4 A simple scene described using a semantic net

In the system, queries are expressed by the user as *query trees* – hierarchical objects with logical operators (AND, OR) as nodes and various specifiers as leaves. The specifiers indicate which medium instances will match a particular leaf and may be facts, medium instances (for similarity retrieval) or other query trees. Therefore each leaf specifies a set of medium instances that are logically combined to produce a description of the *query set* that satisfies the query.

The query set is sent to the query interpreter that calls on medium-specific *matchers* to find the qualifying semantic representations for each medium. If a matcher is handed a raw medium instance for similarity retrieval it must first extract its semantic representation by calling on the appropriate interpreter. The resulting semantic representation is compared with others in the semantic representation base to find those that are similar to a specified degree. The matchers generate a query set that is passed back to the user via the query interpreter and interface. The user is told how many instances were retrieved and, if the number is large, he may wish to refine the query before viewing any instances. In order to make this more efficient, the query interpreter retains the latest query set in a cache so that the new query is only applied to this set. The query interpreter can look to the knowledge base for pieces of the application data model.

Finally, although the query set conceptually contains medium instances it actually only contains their addresses within the raw database, for transport efficiency. Once the user indicates that he is satisfied with the size of the query set, the actual medium instances are retrieved from the raw database.

5 Conclusions

Today's computer information systems do not fully exploit the potential offered by high resolution colour workstations, mass storage, fast networks and continual improvements in Artificial Intelligence. Their natural successor is the Multimedia Information System. The ideal MMIS with content retrieval of all media is still a long way off. It will require the cooperation of experts in all areas of computing technology and from specific applications. Although automatic interpretation will probably never be better than human interpretation it has already shown great potential and will prove invaluable to users of large MMIS's. Other areas in which multimedia is finding a place are electronic mail [Reynolds et al., 1985] and distributed real-time conferencing [Sarin and Greif, 1985].

The Multimedia Group at Manchester University is building a prototype of the system described in Section 4. Initially, the automatic interpretation is restricted to images and will later be extended to text, fact and structured documents. The software is written in C++ [Stroustrup 1986] and operates on a network of Sun workstations.

Acknowledgements

The work of The Multimedia Group is funded by the Systems Integration Division of International Computers Limited and by the Science and Engineering Research Council. Our thanks to the Departments of Electrical Engineering and Computer Science at The University of Manchester for accommodating The Group's equipment and staff. Thanks also to Joe Nelson and Ian Hatton of ICL for their support.

References

AMBRON, S. and HOOPER, K., editors (1988). Interactive Multimedia. Microsoft Press, Washington.

AMERICANIX AI (1989). KBVision System – a tool for image understanding. Sun Technology Magazine, 2(1).

BERTINO, E., GIBBS, S. and RABITTI, F. (1988a). Document query processing strategies: Cost evaluation and heuristics. ACM SIGOIS Bulletin, 9: 169–181.

BERTINO, E., RABITTI, F. and GIBBS, S. (1988b). Query processing in a multimedia document system. ACM Transactions on Office Information Systems, 6(1): 1–41.

BERTINO, E., RABITTI, F. and THANOS, C. (1988c). MULTOS: A document server for distributed office systems. Lecture Notes in Computer Science, 303: 606–615.

CAPLINGER, M., (1987). An information system based on distributed objects. In OOPSLA '87: Conference on Object-Oriented Programming, Systems, Languages and Applications, pages 126–137. ACM.

CHANG, N. and FU, K. (1980). Query by pictorial example. IEEE Transactions on Software Engineering, SE-6(6): 519–524.

CHRISTODOULAKIS, S. and GRAHAM, S. (1988). Browsing within time-driven multimedia documents. ACM SIGOIS Bulletin, 9: 219–227.

CHRISTODOULAKIS, S., THEODORIDOU, M., HO, F., PAPA, M. and PATHRIA, A. (1986). Multimedia document presentation, information extraction, and document formation in MINOS: A model and a system. ACM Transactions on Office Information Systems, 4(4): 345–383.

CONKLIN, J. (1987). Hypertext: An introduction and survey. IEEE Computer, pages 17–41.

CONTI, P. and RABITTI, F. (1987). Retrieval of multi-media images in MULTOS. In Fourth Esprit Conference, pages 1389–1412, Amsterdam. North Holland.

CROWTHER, P.J., DASKALAKIS, C.N., GOBLE, C.A., KAY, S., IRETON, M.A., O'DOCHERTY, M.H. and XYDEAS, C.S. (1989). The way forward. Internal Multimedia Group document.

DASKALAKIS, C., GOBLE, C., KAY, S., NELSON, J. and XYDEAS, C. (1988). Project definition – advances in processing and management of multimedia data. Internal Multimedia Group document.

DATE, C. (1986). An Introduction to Database Systems, volume 1. Addison-Wesley, fourth edition.

FALOUTSOS, C. and CHRISTODOULAKIS, S. (1984). Signature files: An access method for documents and its analytic performance evaluation. ACM Transactions on Office Information Systems, 2(4): 267–288.

GIBBS, S., TSICHRITZIS, D., FITAS, A., KONSTANTAS, D. and YEORGAROUDAKIS, Y. (1987). Muse: A multimedia filing system. IEEE Software, 4(2): 4–15.

GROSKY, W. and LU, Y. (1986). Iconic indexing using generalized pattern matching techniques. Computer Vision, Graphics and Image Processing, 35: 383–403.

IRETON, M. and XYDEAS, C. (1990). A progressive encoding technique for binary images. In Proceedings of the IEE Electronics Division Colloqium on Low Bit Rate Image Coding, London.

IRVEN, J., NILSON, M., JUDD, T., PATTERSON, J. and SHIBATA, Y. (1988). Multi-media information services: A laboratory study. IEEE Communications Magazine, 26(6): 27–44.

ISO (1988). IS 8613: Text and Office Systems – Office Document Architecture (ODA) and Interchange Formats, ISO.

KIM, W. and LOCHOVSKY, F.H., editors (1989). Object-Oriented Concepts, Databases and Applications. ACM Press, New York.

LAWTON, D. and MCCONNELL, C. (1988). Image understanding environments. Proceedings of the IEEE, 76(8): 1036–1050.

LUTZ, E. (1989). Knowledge based classification of office documents. In WOODMAN '89: AFCET Workshop on Object-Oriented Document Manipulation, pages 353–362, Rennes, France.

MEYROWITZ, N. (1986). Intermedia: The architecture and construction of an object-oriented hypermedia system and applications framework. In OOPSLA '86: Conference on Object-Oriented Programming, Systems, Languages and Applications, pages 186–201. ACM.

NAGATA, M. (1984). Retrieval for remote sensing image database with multimedia user interface. In IGARSS '84: Remote Sensing – From Research Towards Operational Use, pages 77–82.

PARRISH, D.M., CREASEY, J.L., THOMPSON, B., ROGERS, D.C., JOHNSTON, E.R., PERRY, J.R. and STAAB, E.V. (1986). Functional requirements for interfacing PACS to RIS. Proceedings SPIE – International Society of Optical Engineers, 626: 597–602.

PIETIKAINEN, M. and ROSENFELD, A. (1981). Image segmentation by texture using pyramid node linking. IEEE Transactions on Systems, Man and Cybernetics, 11(12).

RABITTI, F. and STANCHEV, P. (1987). An approach to image retrieval from large image databases. In Proceedings of the Tenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 284–295.

RENTSCH, T. (1982). Object oriented programming. ACM SIGPLAN Notices, 17(9): 51–57.

REYNOLDS, J. et al. (1985). The DARPA experimental multimedia mail system. IEEE Computer, 18(10): 82–91.

RICH, E. (1983). Artificial Intelligence. McGraw-Hill.

RINGLAND, G.A. and DUCE, D.A., editors (1988). Approaches to Knowledge Representation – An Introduction. Research Studies Press.

ROGERS, D., WALLACE, R., THOMPSON, B. and PARRISH, D. (1986). Information flow analysis as a tool for PACS development. Proceedings SPIE – International Society of Optical Engineers, 626: 690–697.

SARIN, S. and GREIF, I. (1985). Computer-based real time conferencing system. IEEE Computer, 18(10): 33–49.

SMITH, K. and ZDONIK, S. (1987). Intermedia: A case study of the differences between relational and object-oriented database systems. In OOPSLA '87: Conference on Object-Oriented Programming, Systems, Languages and Applications, pages 452–465. ACM.

STROUSTRUP, B. (1986). The C++ Programming Language. Addison-Wesley, Menlo Park, California.

TORIWAKI, J., HASEGAWA, J., FUKUMURA, T. and TAGAKI, Y. (1980). Pictorial information retrieval of chest X-ray image database using pattern recognition techniques. In Proceedings of MEDINFO '80, pages 1154–1158.

TUCKWELL, R. (1987). Mixed data. Computer Systems, pages 41–42.

WALTER, I., LOCKEMANN, P. and NAGEL, H. (1987). Database support for knowledge-based image evaluation. In Proceedings of the Thirteenth International Conference on Very Large Databases: 13th VLDB, pages 3–11.

WOELK, D. and KIM, W. (1987). Multimedia information management in an object-oriented database system. In Proceedings of the Thirteenth International Conference on Very Large Databases: 13th VLDB, pages 319–329.

An Overview of Multiworks

M.E. Morris and I. Cole[1]
STC Technology Ltd., London Road, Harlow, Essex, CM17 9NA

Abstract

The paper provides a simple overview of the intended functionality of Multiworks (MULTI-media Integrated WORKStation) a Technology Integration Project (No. 2105) supported by ESPRIT II.

The goal of Multiworks is to develop a workstation for the office-based professional that manipulates voice, sound, 2D and 3D graphics, video, and text. The aim is to provide facilities for multi-media comparable to those currently available for the traditional media of text and graphics.

The project is taking a layered approach to design and development, with workpackages devoted to the design of a basic hardware platform, a multi-media programming environment, an applications environment and the end user environment. User Requirements in the form of scenarios are being used to drive the configuration and integration of the various layers.

1 Introduction

Existing IT systems have tended to address specialist activities or basic office activities rather than support for more general professional activities in organisations. Professional work has been seen as being too complex and varied to be easily supported by IT, even though professionals provide considerable added-value to their organisation and their time can be expensive. Any support provided to professionals should aim to increase not just their efficiency but also their effectiveness, which is derived from the achievement of key organisational objectives.

Professionals spend much of their time handling issues. This can vary from a quick individual consultation, to a lengthy investigation involving working as part of a large multidisciplinary team. In either case, a professional often has to work with information that is vague, fragmentary and perhaps even unreliable. A professional is also expected to draw upon extensive knowledge

[1]Ian Cole is manager of the Professional Community Support Group at STC Technology Ltd. in Harlow. He is also project manager of the Multiworks project on behalf of ICL.

and experience to produce creative solutions to the problems facing an organisation and to develop plans for the exploitation of business opportunities.

Professionals rarely work in isolation, much of their work is collaborative in nature. The collaboration may have a formal basis, as in the setting up of a work group to handle a particular issue, or it may be informal, as in ad hoc peer-to-peer consultations. Moreover, professional collaborations increasingly involve personnel working on split sites. Therefore, any IT support provided must be able to unite the members of the professional community and provide support for collaboration.

A key aim in exploiting IT is to provide professionals with a system which is supportive to their activities, rather than coercive or restrictive, with minimal system-imposed constraints on their actions.

Modern technologies, particularly those associated with multi-media, have opened the way for novel kinds of IT support more suited to such professional activity. In particular, there is a growing realisation amongst global IT companies (eg. IBM, Apple, HP, Digital), including those with a European base (eg. Bull and Olivetti), that multi-media is becoming a major force in the market place and a candidate for significant product investment. Thus we must also respond to this challenge if we are to position ourselves in this growing market place.

2 The Multiworks solution

Multiworks (MULTI-media Integrated WORKStation) is an ESPRIT project (No.2105) involving collaboration between several European computer manufacturers and software houses. The collaborators include STC Technology Ltd on behalf of ICL, Bull, Olivetti Systems & Networks, AEG Electrocom, Phillips Kommunications Industrie, Philips Components, Acorn computers, Chorus Systèmes, Harlequin, SGS-Thomson, and Triumph-Adler.

The goal of Multiworks is to develop a range of workstations for the office that manipulate video, voice, sound, graphics and text with facilities comparable to those available today for traditional media: text and graphics [MW 90]. Major concerns are with modularity, such that a range of configurations can be produced, adherence to international standards and the use of European technology where possible. Multiworks workstations are also seen in the context of distributed systems and will have the capability to interwork across a heterogeneous range of other workstations.

The project is taking both user-centred and a layered approach to design and development, with workpackages devoted to overall system design as well as to the design of a basic hardware platform, a multi-media programming environment, an applications environment and the professional end-user environment.

2.1 The design process

The design of the Multiworks workstation is being driven by scenarios. Scenarios are sets of "substantiated" requirements based on focussed case studies undertaken in end-user organisations. As such they are marketing and user requirements led exercises. This approach was chosen because of the dubious value of past approaches where the technology was built first and its application sought afterwards. The methodology essentially identifies crucial issues that have to be resolved by the domain professionals in the context of community working.

Scenarios are currently being developed in health, office systems, inventory control, music and education. The scenarios, especially when rapidly prototyped, are a very effective way of moving from requirements to design, and are essential in under-pinning a user-centred approach to system development. The scenarios and prototypes are presented to the target user communities for confirmation that they represent useful and viable solutions from the user's point of view.

The scenarios drive the configuration and integration of the various system layers, and act as the basis for the development of system demonstrators which integrate and add value to the Multiworks technology in relation to particular markets. For example, STL are using a health and a generic office scenario to drive the configuration and integration of early demonstrator systems in Multiworks, as well as the basis for exploring more advanced applications of the Multiworks technology of value to future markets.

2.2 The basic platform

Two hardware configurations will be produced for the basic platform. The primary version will use the powerful 80486 chip as its main CPU, but there will also be a lower-cost option, utilising the ARM3 RISC chip. The use of UNIX* V.4 as the target operating system will ensure hardware independence and portability for higher levels of software. Extra facilities for management of the various peripherals, real-time handling of multi-media data and the basis for an efficient distributed operating system will be provided by the CHORUS distributed operating system nucleus [Rozier et al 88]. Multiworks will also use X-windows and Motif as the standard user interface service.

2.2.1 Multi-media peripherals One of the major technological advances in Multiworks is the integration of multi-media as part of the workstation environment. A variety of media will be supported including all aspects of sound (e.g. speech and music), video, document and script recognition, two and three dimensional graphics, animation, etc. A variety of peripherals will be compatible with the workstation, including scanners, printers, video

*UNIX is a trade mark of AT and T in the USA and other countries.

cameras, etc. A modular approach has been taken to allow for different configurations of workstation to be delivered, with the standard EISA bus providing a common backbone linking the CPUs to the peripherals.

Some of the media, for example speech recognition, require processor boards of considerable power to cope with the large amounts of data produced. The provision of intelligent controllers dedicated to handling specific peripherals will increase the degree of parallelism within the system, facilitate peripheral management and remove workload from the CPU, thereby avoiding a common dataprocessing bottleneck.

2.2.2 Communications Multiworks will also provide advanced communications facilities including FDDI-II and ISDN. This will permit voice and data communications to be integrated right across a distributed system and allow professionals to share and discuss computerised information via the telephone. In particular, the high bandwidth provided by optical fibre local area networks conforming to the FDDI-II standard will facilitate the transmission and integration of multi-media information.

2.3 The programming environment

Programmers must be considered as a second, different group of users for the Multiworks workstation who make rather different demands on the system from those made by professional end-users. Programmers require a system which allows them to produce high quality software in the minimum amount of time.

Two object oriented programming languages, C++ & CLOS [DeMichiel 87] will be provided in addition to the more conventional C. Object and function libraries will be provided to promote the quick development of applications through maximising the reuse of existing code. The provision of a graphical interface builder will allow the rapid implementation of usable and consistent interfaces with the minimum of effort.

2.3.1 Distribution There is an important and growing need in computing to link work stations, peripherals, and specialised servers into distributed networks which maximise the use of expensive resources and create synergy between end-users. However, little application software is currently capable of taking full advantage of the benefits associated with such distribution.

Distribution imposes requirements on both application programmers and end-users which can be minimised by an architecture designed to introduce 'transparencies' which hide conventional concerns such as resource location, operating environment, and inter-process communication.

STC Technology Ltd will be providing an ANSA conformant platform on Multiworks [APM]. The architecture it provides gives a number of long term benefits. It is capable of supporting both Local and Wide Area

Networks, and it allows heterogeneous systems, i.e. systems provided by multiple vendors, to be linked in a productive manner by hiding many of the complexities normally associated with such a scheme.

The programmer benefits from a richer environment, where the distributed architecture will provide novel programming opportunities. Applications will be written in a modular fashion to take advantage of distribution, with the functionality of a given application being shared between different modules and access to such functionality shared between end-users.

2.3.2 Multi-media system resources The application programmer will also have access to a wide range of multi-media resources. Application programs can be written to take full advantage of the multi-media peripherals, both as input and as output devices. It will be possible to create and access libraries of multi-media objects and functions as basic building blocks for the rapid development of new applications. The new software will be able to cope with a variety of data types, including "unreliable" data, such as speech and the output from scanners.

2.4 The application environment

Several exemplar applications will be produced, including an electronic desk, an intelligent forms editor and Resource Connection Manager, based on a hypermedia toolkit called Multicard.

The applications programmer will also be provided with tools to help him develop knowledge-based applications. He will have access to LUIGI, a knowledge engineering environment which will be integrated with Multicard.

2.4.1 Multicard Multicard is a hypermedia system with application development tools which will make it possible to link individual resources together to form complex webs of information, through which the user can either navigate at will or follow pre-defined paths supplied by the developers. The tools will be sufficiently high level to allow end-users also to create personalised applications, perhaps for information management.

Underlying Multicard will be Multitalk, a language similar to Apple's Hypertalk, which will allow the manipulation of multi-media resources. Multitalk will permit the rapid creation of hypermedia-based applications using multi-media building blocks and their associated editors to create more complex resources. For example, documents will be able to have embedded cartoon and video sequences activated at the touch of a button; reviewers will also be able to add voice annotations to documents, etc.

2.4.2 LUIGI Luigi is a knowledge representation system with an elaborated graphic knowledge manipulation environment and especially tailored to support the construction of large knowledge-based systems. It is implemented using the Lispworks^{TM} development environment supplied by Harlequin [Harlequin 90].

2.5 The professional end-user's environment

The professional will be provided with one or more graphical user environments providing access to all the resources and applications they require, including facilities provided by UNIX e.g. Email. A variety of presentation metaphors can be provided, including office desktops, hospital ward overviews, etc. whatever is appropriate for the user and/or tasks involved.

The user environment will be based on the direct manipulation of multimedia resources, and will have a consistent 'look-and-feel' due to the provision of a set of basic building blocks for the programmer. The user will have the ability to transform the data presented on screen into a variety of representations. For example spreadsheets will be capable of conversion into graphs, bar charts, pie charts, etc. It will also be possible to have multiple representations of the same data on screen simultaneously, with the system maintaining their consistency. Users will also be able to work co-operatively with colleagues.

However, this raises two questions: how will applications be integrated?; and, what kind of co-operation between users will be supported?

2.5.1 Multi-media resources A resource can be any data the professional finds useful in his or her work e.g. documents, spreadsheets, graphs, etc. For Multiworks resources are not just made up of the text and bitmaps found in most current systems. Multiworks aims to provide a multi-media environment where text and graphics can easily be mixed with speech, music, video, etc.

The system will allow resources to be plugged together using the hypermedia toolkit, i.e. professionals will be able to link resources and specify the data that is passed between them. For example a spreadsheet could be linked to a database, with changes in the database reflected by corresponding changes in the spreadsheet (and vice versa).

Professionals will be able to share data simply and easily with their peers e.g. webs of resources relating to a specific project can be shared between the members of a work group. They will have transparent access via the underlying distributed platform to information no matter where it is stored in the system. This is in contrast to current practice where users can only access information if they know its format and location.

Established professional communities and organisations will be able to set up databases of information relating to their activities. These databases will be accessible by any member. In this way changes to corporate standards, policies, document layouts, etc. can be propagated quickly throughout the workforce. New members of a community will be able to browse the information at will, helping to reduce training overheads. Specific exemplars and generic templates can be provided e.g. templates for monthly reports, example project plans, etc. which can be reused, thereby reducing the time spent on needless repetition of tasks and consequently increasing efficiency.

2.5.2 Computer-supported co-operative work Professionals are rarely located together, however they need to share all kinds of resources, from data and documents to expensive hardware.

Multiworks will provide the capability for professionals to co-operate in real time by sharing applications, information on their screen, and establishing a voice dialogue. Such a capability will be provided as an intrinsic part of the distributed platform rather than as a single application. This again prevents users from having to know where such an application might be and allows them to concentrate on the collaboration rather than finding the application.

3 Conclusion

Multiworks will provide the potential for a range of configurable, multimedia workstations. Not only will these workstations intercept major international standards, but they will also support the development of applications suitable for increasing the effectiveness of professional communities in organisations. Such a capability will provide a significant differentiator in a market with a hugh potential for exploitation, both by ourselves and our European partners.

4 Acknowledgements

The authors would like to acknowledge the contributions made to the project by their colleagues throughout the Multiworks Consortium.

5 Glossary

ANSA	Advanced Networked Systems Architecture
ARM	Acorn RISC Machine
CHORUS	A distributed operating system nucleus from Chorus Systèmes
CLOS	Common Lisp Object System
CPU	Central Processing Unit
EISA	Extended Industry Standard Architecture
ESPRIT	European Strategic Programme of R&D in Information Technology
FDDI	Fibre Distributed Data Interface
ISDN	Integrated Services Digital Network
IT	Information Technology
PCS	Professional Community Support

References

[APM] ANSA Reference Manual, v 00.01, APM Ltd., Poseidon House, Castle Park, Cambridge.

[DeMichiel 87] DEMICHIEL, L. and GABRIEL, R., "The Common Lisp Object System: An Overview", Proceedings of the 1987 European Conference on object oriented programming, Paris, France, June 1987.

[Harlequin 90] The Lispworks Manual, v 1.1, Feb 1990, Harlequin Ltd., Barrington Hall, Cambridge.

[Rozier et al 88] ROZIER, M., ABBROSSIMOV, V., ARMAND, F., BOULE, I., GIEN, M., GUILLEMONT, M., HERRMANN, F., KAISER, C., LANLOIS, S., LEONARD, P. and NEUHAUSER, W., "Chorus Distributed Operating System", Computing Systems 1(4), 1988.

[MW 90] "Multiworks Technical Annex", May 1990, Multiworks Consortium, ref EP2105.

RICHE – Réseau d'Information et de Communication Hospitalier Européen (Healthcare Information and Communication Network for Europe).†

Tony Drahota
International Computers Ltd, Slough, Berkshire, UK

Abstract

The RICHE project was conceived and organised as a response to the requirements of the health care sector for more effective use of information technology. A large consortium of European I.T. companies and health care organisations set out, in 1989, to define and take to pilot implementations an architecture for health care systems which will apply equally in all European countries and which will reflect the need to integrate support functions for healthcare professional, administrative, technical and managerial activities. This ambitious three year project, now in its second year, has successfully completed the first phase of its activities – the analysis and modelling of the overall requirement and the initial definition of the global architecture. This paper gives an overview of the project findings to date and it previews some of the results expected from current and future work.

1 Background

RICHE is a project partially funded by the Commission of the European Community (CEC) under the Office Systems Branch of the European Strategic Programme for Research in Information Technology (ESPRIT). The RICHE consortium is an international one, consisting of health services and hospital representative organisations, research institutes, software houses and computer manufacturers from France, Ireland, Italy, The Netherlands and the United Kingdom. The consortium's expertise in health care and health informatics covers all major branches of health care systems. Since the product of RICHE must be applicable to the whole of Europe, the project has established working relations for collaborations in Belgium, Denmark, Greece, Federal Republic of Germany, Luxembourg, Portugal

†Funded by ESPRIT Project 2221.

and Spain, as well as in all EFTA countries. RICHE also has contact with the World Health Organisation and other international bodies, national health authorities and standards institutions.

The members of the RICHE consortium, which will invest approximately 1,000 man-months of effort during the period 1989–92, are:
CONSEIL DE FILIERE STAF, le Mans, France, (prime partner)
BULL S.A., France
IIRIAM (Institut International de Robotique de l'Intelligence Artificialle de Marseille), France
SIG SERVICES B.V. (Informatics in Health and Welfare), The Netherlands
BAZIS (Central Development and Support Group, Hospital Information Systems), The Netherlands
IRISH MEDICAL SYSTEMS, Ireland
UNIVERSITA CATTOLICA DEL SACRO CUORE, Italy
GESI (Gesione Systemi per L'Informatica S.r.l.), Italy
LOMBARDIA INFORMATICA S.p.A., Italy
ICL (EUROPE), UK

2 Rationale for RICHE

Health care systems, and the hospitals within them, are information intensive and vast amounts of data are transferred within them. Due mainly to demographic factors, demand for health care will increase in all societies of Europe in the coming years and, consequently, so will the information volumes and traffic. Health Authorities will have to place even more emphasis on cost containment; exploitation of advanced information processing technologies can contribute significant savings.

The progressive unification of Europe is creating a market large enough to permit the very costly development of European solutions for the health care market. However, as many past studies and experience show, the most significant benefits will be realised by enabling integrated systems to be built and for this standards are required. It is one of the main objectives of RICHE to define a system architecture which is applicable in European countries regardless of their political-administrative and socio-economic environments and national legal systems.

The RICHE reference architecture will define the building blocks (or types of system components) from which RICHE systems can be built. It will also define the rules constraining the inter-relations between the components, i.e. how they may be inter-connected and used in conjunction and co-operation with one another. These rules will be based in part on a global data model, which the project is developing, and on a variety of open systems standards (e.g. communications).

The RICHE reference architecture will allow the I.T. industry to design and develop new applications for health care which will be able to inter-link with each other, and it will also show how existing applications could be interfaced to RICHE systems, this being necessary to protect existing investments and reduce the development burden.

Historically, the application of I.T. in health care has been concentrated in administrative areas and little direct support has been offered to the professional activities of doctors and nurses.

RICHE pays particular attention to the needs of these professionals, indeed applications specifically aimed at them will be designed and prototyped during the project.

Significant benefits are anticipated for patients, health care professionals and managers of health administrative systems as a result of RICHE. These will be achieved through better on-line and up-to-date information about patients, possibilities of direct consultation with colleagues and experts, use of decision support and knowledge-based systems, availability of comparable and reliable epidemiological data for health surveillance, cost savings by more efficient use of facilities, improvements in quality and timeliness of information for policy making and day-to-day management, and so on.

3 Scope and objectives of RICHE

A health care system can be considered to be composed of six groups of functions and the interfaces between them:

- Patient management – dealing with patient identification, admission, transfer and discharge
- Medical care
- Nursing (short and long-term)
- Medical support (e.g. laboratories, operating theatres)
- Ancilliary services (e.g. catering, cleaning)
- Management and administration

Figure 1 illustrates this view. In its present phase, RICHE has restricted itself to detailed examination of the Patient Management, Nursing and Medical care process and to the interfaces and data which are required by Management, Medical Support and Ancilliary Services. On the basis of modelling and analysis of the requirements within the above RICHE domain, a RICHE architecture definition is being produced and prototype applications, conformant with it, will be designed and implemented for the functional areas of Nursing and Medical Care.

The context within which the RICHE project is developing the architecture is best illustrated by the key criteria which RICHE must meet:

Openness: the ability to incorporate new technologies and, possibly unforeseen, application services.

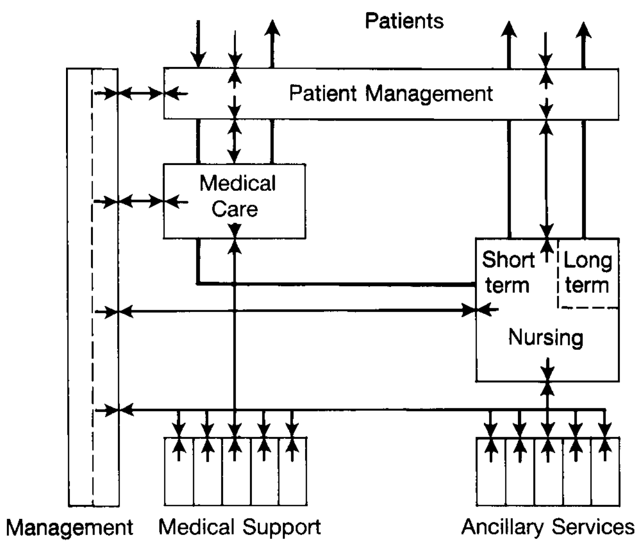


Fig. 1 Healthcare system functions.

Modularity: which ensures that changed requirements result, as far as possible, in localised changes, and allows modules to be re-used in several application services to reduce costs.

Distribution: of both information and processing over a heterogeneous machine environment.

Standards: the maximum exploitation of agreed, emerging and de facto European and international standards.

Exploitation: wherever possible, of existing commercially available products and system components.

Homogeneous system appearance: to be achieved through the use of a common Human-Computer interface service and a common style guide.

Application portability: through the provision of a standard Application Programming Interface (API) to the generic and basic services.

Additionally, the RICHE results must be equally applicable in different countries of Europe, and hence RICHE must fit into different organisational, cultural and economic environments. A brief overview of the approach adopted to achieve the RICHE objectives is given in the next section.

It is evident that the RICHE initiative is tackling a set of very complex problems. Finding the project within the ESPRIT pogramme, one could assume that the consortium intends to research and develop new techniques and technologies to address the Health Care requirements. However, this is not so. The RICHE consortium believes that its mission can be achieved through the adoption of existing technologies and approaches, albeit sometimes very advanced ones, and therefore RICHE should be viewed as, essentially, an integration project focusing on understanding of needs and bringing together of appropriate I.T. capabilities to provide a solution.

4 Modelling

To develop its solution, the RICHE project had to gain a comprehensive understanding of the requirements for health care systems. Examining these requirements, different systems professionals have different viewpoints and often focus on different concerns they consider crucial.

In RICHE, due to the project timescales, it was important that the definition of "technical" architecture was conducted largely in parallel with the capture and modelling of the requirements.

Therefore, a framework for modelling was required, which permitted the parallelism of activities, as well as supporting the different viewpoints of, for example, the (medical) doctors, nurses, system designers, hospital managers and so on.

The framework of abstractions, which is the subject of reports from the ISO Group working on Open Distributed Processing (ISO/IEC JTC1/SC2/WG7), suggested itself. Within this framework, information systems are considered from five perspectives or viewpoints:

Enterprise
Information
Computation
Engineering
Technology

Within each viewpoint, the system can be considered from various aspects (e.g. security, dependability, privacy, etc). Indeed these aspects will usually appear in all the viewpoints, although possibly in a different order of priority or expressed differently. Also with each viewpoint, the system can be positioned at a different level of abstraction: ranging from a totally general view of systems to a highly constrained view of a particular system. Thus a three dimensional framework is created (see Figure 2) where:

- The role of the information system within an organisation is described in the **Enterprise** projection.
- The information requirements and structure are described in the **Information** projection.
- The operations to be performed on the Information are modelled in the **Computation** projection, but independently of any computer systems and networks.
- The description of how to mechanise the application definition given by the computation model is contained in the **Engineering** projection.

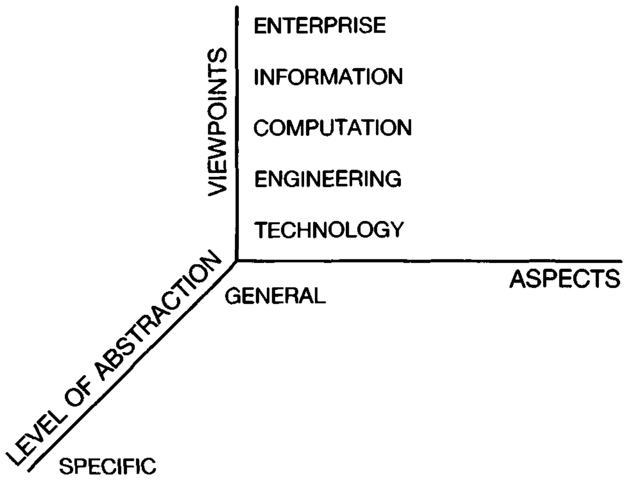


Fig. 2 Modelling framework.

- The technical components and standards from which a system can be constructed are defined in the **Technology** projection.

Although the projections are not independent of each other, it is quite possible to work on several in parallel, at different levels of abstraction.

RICHE has proceeded along two parallel and closely co-ordinated lines of activity, one addressing the enterprise, information and computation models, the other concentrating on the engineering projection. Given a well defined boundary for the functionality which the RICHE system is to provide, the former was conducted in highly specific terms. The engineering projection results are more abstract and as such apply to a wider range of system solutions than RICHE. This was exploited and an agreement was reached with another ESPRIT project (BANK 92) to combine efforts to specify and develop components required by both projects. The two projects are now jointly specifying and developing a common platform, designated RIBA (for RICHE and BANK 92), which will support the development and use of application programs and services within the domain of either project.

4.1 Enterprise viewpoint

Key concerns related to Community Health Care and Hospital systems within this were identified from past studies and from experience of the consortium members. A study report on 14 European countries was produced by Prof. Albert van der Werff.

Amongst its findings, the differences in organisational and financial structure were highlighted and are reflected in the approach taken to modelling within the information and computation projections.

4.2 Information and computation viewpoints

The RICHE consortium contains, and collaborates with, many users, who are directly participating in the definition of the models. It is essential that these people understand and can use the modelling language. Therefore, RICHE has adopted the classical Entity-Relation and Data Flow models for the information and computation viewpoints. The chosen methodology, which is supported by a case tool (MASTER) provided by one of the RICHE partners (GESI), has proven to be highly effective in the RICHE environment, and extensive models have been developed very quickly using it. These models provide a common view of the health care systems in different countries. Of course, there are differences between countries; indeed, differences can be found between districts of a single country, in terms of the exact process and data which they use or need.

Various techniques were used by RICHE to cater for such variations with a single model. One of the techniques is to model the local variations in processes, rules and procedures as attribute values within the data structure. This also implicitly introduces the need for knowledge-based support within the RICHE system.

It is through this knowledge-based approach that a RICHE system will be adaptable to different end users' day-to-day practices and will cater for different hospitals' procedures and rules.

Four different levels of knowledge have been identified

- Overall RICHE level, e.g. common medical criteria/practice
- Country level, e.g. country laws, regulations
- Organisation level, e.g. district/hospital regulations and circumstances
- End-user level, e.g. user profile – personal practice.

A set of concepts were introduced for knowledge representations and processing including:

- taxonomies of abstract classes, where:
  - a class features structured objects with multi-valued attributes
  - pre-defined relationships are set between classes of objects
  - pre-defined constraints and properties are set on the class itself or its attributes values.
- actual objects are instances of terminal classes
- classes can denote composite objects
- pre-defined properties or relations may be attached to a composite object and may be propagated to its components
- a hierarchy denotes inheritance from class to sub-class and sub-class is a specialisation of the parent class
- rules or scripts of actions can be attached to a class or its attributes.

The knowledge-based approach is extensively exploited within an important part of the RICHE system – ACT MANAGEMENT. An ACT is any activity which modifies a patient status or the information concerning the patient status. Much of the activity within hospitals can be expressed in terms of acts, and, in fact, most acts are standard and can be accurately classified. For example, the removal of an appendix can be considered as an act belonging to the class of acts "Operation". Acts follow a lifecycle, they can be requested, planned/scheduled, performed, reported, etc. Acts can also be composed of other acts (sub-acts), and there may be "interaction" and "obligation" relationships between acts.

Act management is the key to enabling integration of health care systems. As patients progress through the system, appropriate acts are determined by their doctors or nurses, reported upon and, from their results, further acts are triggered. Other activities within the system are usually consequences of acts (e.g. laboratory tests, preparation of operating theatres, etc.). Information about acts (e.g. usage of consumable objects during an act, attendance of staff during performance of an act, etc.) feeds into resource management functions. Relationships between acts (e.g. obligation, interaction) can be exploited for scheduling purposes and to provide intelligent assistance to hospital staff.

4.3 Engineering viewpoint

In the first year of RICHE, work within the engineering viewpoint concentrated on the RIBA platform.

RIBA is a platform (i.e. a collection of software) enabling the application program designer and writer to view a network of, possibly heterogeneous, machines as a single abstraction with an interface providing a rich choice of the services and functions that are required for the application. RIBA offers the benefits of distributed processing and openness but without the normally attendant complexities.

Some of the benefits of RIBA are:

- the system designer is not tied to a single supplier of H/W or operating regime
- systems can be expanded and enhanced more readily
- application designers and programmers are freed from concerns with local machine environment and the mechanics of interaction between application components, and hence development productivity increases
- overall system performance can be enhanced by exploiting true parallelism in processing, through for example, concurrent use of replicated servers.
- dependability of systems can be improved by replication of key components.

Conceptually, the RIBA platform is seen as a number of layers above the networked machines (Figure 3). The lower layer is the Support Environment for Open Distributed Processing (SE-ODP). In the initial implementation, this is largely based on the work of ANSA[1] as evidenced in the ESPRIT 2 project I.S.A. Concepts from the COMANDOS project will be incorporated at a later stage, to provide specialist support for highly distributed applications.

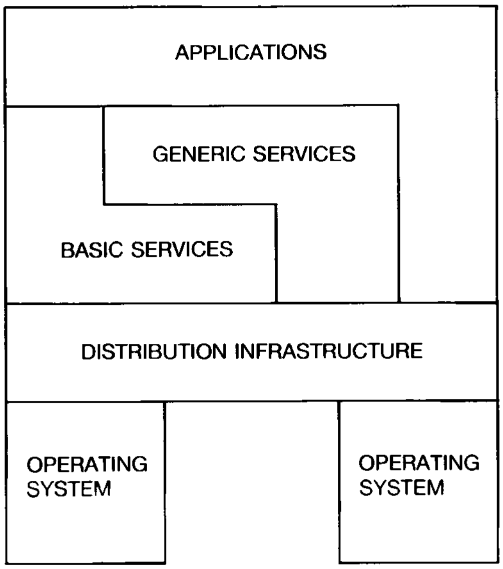


Fig. 3 RIBA overall structure.

The upper layers of RIBA form the application support environment and consist of servers commonly required by applications. The servers are divided into those that are general purpose for a wide range of applications (i.e. basic servers) and those which are restricted to use in health care (generic servers).

It is important to realise that the layering is conceptual only. Objects in any layer can access objects in all other layers directly, this also applies (but with a consequent loss of object portability) to interfaces in the underlying operating systems.

RIBA contains mechanisms for concealing the effects of distribution from the components and users of the system. In particular, RIBA provides Access Transparency and Location Transparency in its current release (RIBA V.O.). Access Transparency conceals the communication services from objects such that interactions between objects are identical semantically and syntactically, whether the objects are local or remote.

Location Transparency conceals the location of objects in a distributed system by ensuring that bindings between them are independent of the routes that connect them. Functions for naming, addressing and routing are provided in RIBA for this purpose.

Future versions of RIBA will introduce Migration Transparency (i.e. ability to move objects transparently to other objects), Replication Transparency (i.e. ability to run multiple copies of an object) and Concurrency Transparency (i.e. parallel usage of resources).

A very important consideration for the portability of applications is that RIBA should conceal the differences present in the underlying machinery and operating systems. Currently, RIBA runs on UNIX V.3*, MSDOS*, SunOS*. Support on UNIX V.4*, OS/2* and ULTRIX is expected.

RIBA supports the client-server model of interaction between objects. The bindings between clients and servers are established through the services of a *Trader*. The Trader acts as a directory of interfaces; servers export their interface to the Trader, clients import the desired interfaces from it. The Trader matches descriptions against requests from clients.

RIBA also provides the necessary tools for construction of distributed systems. These include the Interface Definition Language (IDL)[1] which allows the definition of primitive and constructed data types and their use to specify the signature of each operation of an interface. Also provided is a stub compiler which converts the IDL specification into a set of stub routines. These stub routines handle the marshalling and unmarshalling of arguments and results; they also invoke the appropriate interpreter instructions to convey the arguments and results between separate capsules containing the communicating objects. Distributed Processing Language (DPL)[1] and a pre-processor are provided. DPL is used for interface trading (Import, Export, Discard, Withdraw, Initiate, etc.), the pre-processor provides the linkage between the DPL statements and the stubs generated by the stub compiler.

5 Planned activities

So far, RICHE has developed a comprehensive integrated model of the core functions of a health care system. The project has also released the first version of the RIBA platform.

*MOTIF is a trademark of Open Software Foundation, Inc.
MSDOS is a trademark of Microsoft, Inc.
OS/2 is a trademark of International Business Machines Corporation.
SunOS is a trademark of Sun Microsystems, Inc.
UNIX is a trademark of AT&T in the USA and other countries.
XWindow is a trademark of MIT.

Work is progressing in several streams

- design of act management
- prototyping of applications (nursing and medical processes)
- development of RIBA basic servers
    - Human Computer Interface (HCI) servers based on X-11 and MOTIF*
    - Authentication Server based on the mutual authentication principle
    - File and Print Servers
- specification of an integrated multi-media information server, to support text, image and relational data.

Further releases of RIBA are planned, these will progressively introduce new basic servers as well as enhancements to the distribution infrastructure (Replication Transparency support, Migration Transparency support). Support for C+ + will also be added.

6 Summary and conclusions

RICHE is a large multi-national consortium dispersed throughout a large part of Western Europe. Despite these factors, the consortium is achieving remarkable progress. In part, this can be attributed to the multi-disciplinary nature of the project team – we seem to have a specialist for anything. The involvement of users in several countries is a major benefit. Another possible reason for good progress is the willingness of the consortium to adopt other people's good work. Thus the project is not side-tracked into extensive research activities – the "not invented here" syndrome has not appeared.

7 Acknowledgements

This paper has drawn on much RICHE internal material. The author would particularly like to acknowledge work of

- Prof. Albert van der Werff (SIG) on Healthcare in Europe
- Dr. Fabrizzio Ferrara (GESI) on modelling
- Dr. Henri Kanoui (IIRIAM) on Knowledge Management
- Hubert Mensch (STAF) on Act Management
- Len Crockford (ICL) on Information Management and RIBA
- David Whysall (STL) on RIBA

Reference

1 ANSA, "The ANSA Reference Manual release 01.00", March 1989. Architecture Projects Management Ltd, Cambridge, England.

ESF – A European Programme for Evolutionary Introduction of Software Factories

Richard Thomas
ICL – Project Director
Christer Fernstroem
Cap Gemini Innovation – Project Technical Director
Olaf Hesse
University of Dortmund – Market Integration Manager for Project

Introduction

The Eureka Software Factory (ESF) project was established under the auspices of the Eureka programme by a consortium of fourteen companies, universities and research institutes.[1] They are:

AEG/GEI, Germany
British Telecom, United Kingdom
Cap Gemini Innovation, France
EB Technology, Norway
ICL/STC, United Kingdom
Imperial College, United Kingdom
INRIA, France
Matra, France
Nixdorf Computer AG, Germany
SEMA GROUP, France
SEMA GROUP, United Kingdom
Softlab, Germany
Telesoft, Sweden
Universität Dortmund, Germany

The members include industrial groups and research centres and have a wide range of interests in software, in its various forms of application. Concern about software supply and changes in world population trends and economies led the members of the consortium to launch a project to industrialise software production. The vehicle for this industrialisation is the software

[1]The partners acknowledge the fact that they would not have undertaken such a long-term intensive effort without the support of the following governments within the Eureka programme: Federal Republic of Germany, France, Norway, Spain, Sweden and the United Kingdom and the interest of the Commission of the European Communities.

factory. However, the use of the word "factory" in this context has little to do with the classical notion of a factory with its emphasis on centrally controlled automatic production lines. Instead, we must look at the modern-style factory where centralisation and decentralisation are carefully balanced and where continuous efforts are directed towards the integration of independently evolving asynchronous production facilities. The prime goal of the consortium is to establish software factories in practice in industry. Intermediate targets essential to achieve the prime goal are: visibly to demonstrate that software production is improved by the software factory approach and to establish a base of knowledge and facilities to support the development, construction and application of software factories in key segments of software-dependent industry.

To support these aims, the 1987 Eureka Ministers' Conference accepted the ESF International Proposal, to establish a project over a ten-year period, to be funded at 400 million dollars. Due to the scale of the operation, the commitment of the consortium is legally consolidated in a joint venture agreement.

The decision to built the consortium within the Eureka programme was based on the need to collaborate in the selection, development and exploitation of leading-edge technology, combined with the need to exploit existing technologies. The latter factor recognises the present-day investment position in existing technology and the need to evolve in financially feasible steps, e.g. 1–2 year funded investment programmes. The consortium saw the need for co-operation on a critical mass scale which is able to go further than the pre-competitive research stage. In this sense, the attractions of the Eureka programme are twofold. It will create new products and services in European industry aimed at the world market, and it will increase productivity and thereby strengthen the competitiveness of Europe's industries and national economies on a world-wide basis. In the Eureka programme, participants have full responsibility for defining and implementing their scientific and technological co-operation projects. They are their own judges of the best course to new markets for Europe.

Overall Perspective of ESF

The major aim of this paper is to present an overall perspective of the Eureka Software Factory, as seen and understood within the consortium in the summer of 1990. We have already given an impression of the scale of investment, our goals and stated which companies are involved. We also stress that the consortium will deliver products i.e. not terminate activities at the research prototype stage. So, what is the need for operating on this scale of investment and organization?

A significant part of our investment is given to understanding the strategic trends on the level of world-wide commercial problems and analysis. In our project definition and project start-up phases, we examined the major trends in technology and major industrial requirements. This resulted in a solutions strategy which is being used to co-ordinate the work of the consortium. In this overview we will describe our solution strategy using a framework to contain all relevant technology and focussing on the core technology specific to software factories.

The problems related to software development are well-known and documented. The major concerns when addressing these problems can be summarised as follows:

- High investment in individual software solutions needs to be protected
- Variety of standards are emerging
- Industrial standards set by large companies with world-wide domination
- No existing market for standardized software components
- No common understanding of the software production process
- Fragmentation of industry and technology
- No single vendor solution to the software factory

It is clear that the scope of a factory must embrace the organization and its workforce, whatever their sphere of control or individual activity. Equally, it must embrace the information which controls, guides or supports their work, and the tools which they use to perform their work. In the software factory, we recognise the existence of computerised and non-computerised parts. We must provide the means of making sensible choices for provision, selection and use of computerised parts. We must also provide the support for understanding the organization and its needs.

It follows that the ESF framework must embrace, at one level, the range of methods and tools specific to life-cycle support in a variety of industries. The framework must also support, at a platform level, the range of extensions to OS in the areas of User Interface, Object Storage etc. Within this framework, ESF will focus on a set of enabling core technologies (see Figure 1).

These core technologies will support four fundamental principles: support for the process, plug-in facility for components, a 'user in focus', and flexible construction.

The core technologies are configured in the framework to conform to a reference architecture which contains a minimal kernel focusing on the 'plug in' of components and common aspects of process support. The minimal kernel will enable the integration of essential factory facilities with industrial methods and tools as well as the industry platforms.

We recognise that within the framework of a factory it should be possible for many existing structures, products, technologies and standards based on present day investment to co-exist. The investment position is determined by a combination of known development products, configured and stabilised to serve a development organization and its established custom and practice. If

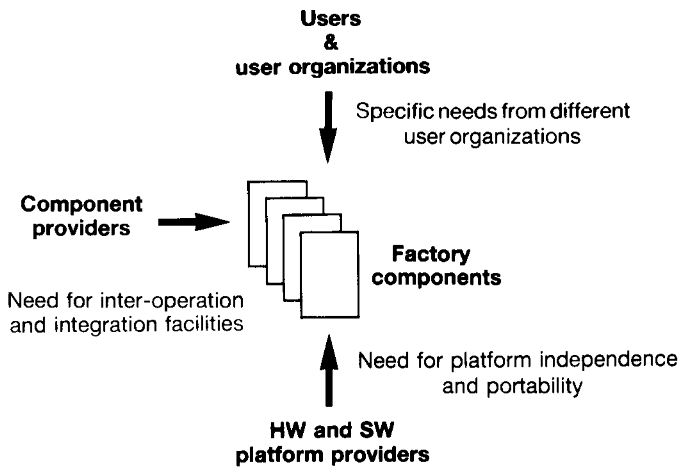

Users
&
user organizations
Specific needs from different user organizations
Component providers
Need for inter-operation and integration facilities
Factory components
Need for platform independence and portability
HW and SW platform providers

Fig. 1 Requirements on the core technologies come from different sources

improvements are to be made without destabilising the existing organization, this may only be achieved by establishing a committed 1–2 year investment programme. Change should therefore be seen to be beneficial and there should be constraint against changing what is already seen to be valuable. Consequently, it is the major elements of existing development support which must evolve.

The emergence of new ideas and technology from various research initiatives raises the question of their inclusion in the wider scope factory environment of people and of the computer-based services commonly referred to as 'tools'. European investment in methodology has made substantial progress in understanding the type of computer-based functionality essential to support the work of say a designer, an analyst etc. However, the perceived need for co-operative work requires a widening of the scope to include all major 'operations' and to provide the means of co-operation. This reinforces the idea of a framework and an essential core technology for the software factory. The scheme must be able to include new technology as it emerges and as it ceases to be an 'isolated operation' and becomes an 'operation' in the integrated environment.

So of all the new technology, what is the essential core of the software factory? The results of the initial stages of the project clearly focus on the enabling technologies which support improved co-operative work, and the re-use of existing high quality 'components'. These principles are naturally accepted in most industries using modern factory thinking.

Co-operation can be viewed on two levels:

- The co-operative working between people in an organized group.

- the intercommunication between the computer-based services that support the work of individual people in the organized group.

The Core Technology of ESF

One important aspect of co-operation is co-operation between people. The target scope of ESF is to support the needs of all people who play a role in producing software, e.g. designers, builders, maintainers, quality managers, product managers etc.

The management decision-taking level of an individual factory organization will be concerned that:

- the overall goals for the organization are clear.
- the essential processes to support the goals are defined and understood.
- the key roles of the individual workers are understood.
- the skills base exists to support these roles.
- the basic requirements of computer-based support are provisioned.

One area of ESF core technology is therefore the modelling of support consistent with the above concerns.

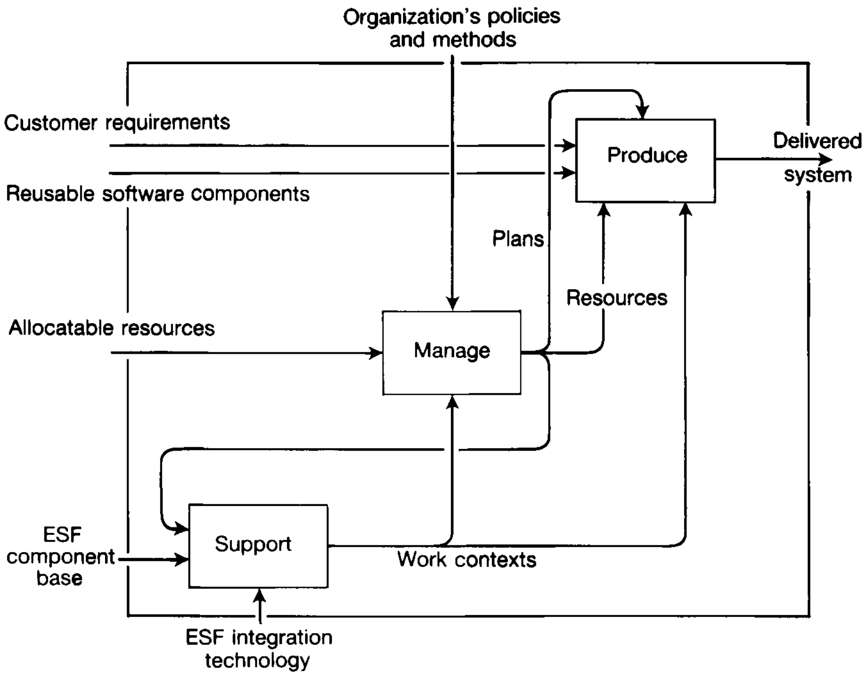


Fig. 2 Software factory process framework, showing the principal sub-activities of software production

Figure 2 gives an example of the software manufacturing process framework supported by an ESF factory. It shows how the main process is decomposed into three principal sub-processes: *produce*, *manage* and *support*. The "produce" sub-process, which results in the delivered system, uses customer requirements and reusable software components as its inputs. It is controlled by plans generated by the second sub-process, "manage", and supported by resources, allocated by the manage sub-process and work contexts, generated by the "support" sub-process. The responsibility of the support sub-process is to provide all factory sub-processes with computerized support in the form of user work contexts (see below). The generation of work contexts is derived from project plans and available factory components. It is supported by the ESF integration technology.

An organization that is modelled can be more easily adjusted to new circumstances arising from external issues, e.g. market change, economic change and internal change, e.g. staff migration, costs.

The computer-based support at a functional level is provided as a total set of services available to all workers. The bridge into the organization is achieved by placing the user in a *work context* derived from the process model and by providing the services needed for that work context e.g. access to functionality, access to common data etc.

The definitive ESF work in this field is derived from the research trends in:

- the design of IT artefacts such that they are sympathetic to the organization and its human population.
- the open systems architecture view of connecting services as required to support the user.

The deployment of resources in the overall application of the software factory is consequently primarily under the direction of the factory management team and key technical staff. Thus all major processes and roles are defined and supported.

A further level of support will be provided to *assure predictability*; i.e. defined processes from the process model will be supported by mechanisms which guide the user to be consistent with that defined process. Support for this feature, *active process*, will appear with newly emerging process modelling languages and expert systems.

The scheme does not preclude the existence of a work context which provides freedom of access to additional services – here the degree of control or freedom is open to the individual organization. The opportunity can exist for highly skilled users to have freedom of choice in their solution strategy, but still within a defined framework of work content and time.

The idea of *process* can be applied at various levels e.g. the overall industrial process for developing avionic software, the process which captures the procedure for fault clearance, the process which captures the procedure for signing off a release.

Although the idea of common overall processes has become a strategic issue, the result will probably be a generic process in each industry allowing freedom of choice of specifics in individual companies or departments. This will be influenced by procurement practice. The value of defining complete industrial processes lies in the clarification of the essential activities and parts necessary to support that industry, creating a demand framework for motivating the availability of the 'parts'. Many aspects of differing industrial processes will be common, but there will be specific attributes of the individual industry products which call for special attention in that area. This leads to the ESF view that a re-usable parts set can be configured with specific new parts to achieve more cost-effective support of development organizations. Therefore it is worth aiming at a European base of services and knowledge to customise individual factories from the base. This aspect of componentry is the essential core of ESF and combines with process support, focused user interface support and communication-centred architecture to provide the essential integration technology of ESF.

The ESF model for factory construction, which is shown in Figure 3, is based on the instantiation of generic models:

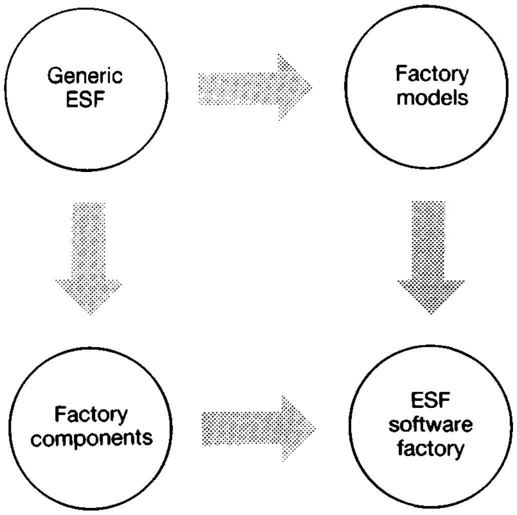


Fig. 3 The ESF model for software factory construction

- The generic ESF level defines a reference architecture for software factories, conformance criteria for ESF-compliant factory components, a meta-model for factory modelling and notations (languages) for describing components and factory processes.

- Implementation of components or re-engineering of existing CASE tools in conformance with the generic ESF definitions continuously extends the base of available factory components.
- The specific needs of a software factory customer organization are described in terms of factory models, where both the processes to be supported and the characteristics of the support environment are specified in accordance with the generic ESF.
- The actual generation of an ESF software factory instance is driven by the corresponding factory models. The generation process is supported by capabilities for factory process analysis, component selection, component adaptation and parametrization and for component integration.

The approach of flexible integration from a common base of components, in which specific, related sets can already exist, provides the following possibilities:

- that major platform suppliers and their key standards can be configured together to achieve an adequate degree of co-operation.
- that major industrial users can pool their expertise in common processes and combine their support, focusing on the process essential to their specific industry (note that this aspect also affects education and training).
- that cross configuration between industrial users and platform suppliers becomes more controllable.

The ESF strategy is to begin an evolutionary process of partial support across each of the core technology threads making balanced progress at successive levels of integrability consistent with a common ESF reference architecture. Development of integration technology and the transfer of this technology will be in a number of evolutionary steps through the 1990s. The movement can embrace several industrial variants converging only where necessary, consistent with open system interface strategies.

Factory Architecture

ESF defines the architecture for factory support environments in terms of a reference architecture which specifies requirements that must be fulfilled by every particular factory instance. The three main concerns which have driven the definition of the architecture are directly derived from the requirements in Figure 1 above. Multiple platforms and market fragmentation are fundamental aspects, as well as the need for adaptation to the needs of customer organizations. The latter has three immediate consequences:

- Factories must be modular and must allow components to be easily added, removed or replaced, with minimum perturbation of ongoing work.
- Components are "black boxes" as seen by users and their organizations and must therefore be policy-independent. Policy should be defined by process models, which means that components must not be monolithic.

- Questions of availability of support functions should be separately handled from questions of how support functions are used. This means that user interaction should be handled by distinct components.

Current trends in systems architecture seem to favour service-oriented architectures, since these allow a high degree of re-use in that the same service may be used by many clients, possibly simultaneously and in different contexts. The ESF architecture is based on the service concept and allows clients to be dynamically bound to services.

The ESF architecture is furthermore based on well-known principles of modern software engineering practices, such as encapsulation, modularization and the powerful structuring mechanisms of object-oriented systems. A component description language specifies different aspects of component functionality: as abstract entities (e.g. in terms of abstract data types), as module interfaces (similar to package specifications in Ada) and in terms of representations (allowing differences in representations and mechanisms of the underlying platforms to be factored out). A low degree of direct interdependency allows modules to be treated as "pluggable" components.

Figure 4 shows a structural view of the architecture. In this view a factory consists of components connected to a software bus. The Figure shows two types of component: user interaction components and service components. The software bus is an abstract communication channel which hides distribution aspects and which allows the exchange of data without loss of structural and conceptual information, thus supporting component interoperation and integration. It involves two principal services:

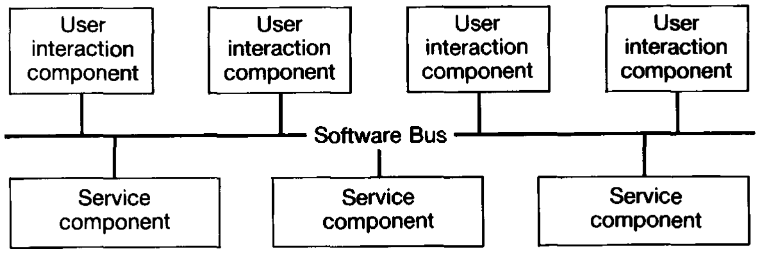

User interaction component
User interaction component
User interaction component
User interaction component
Software Bus
Service component
Service component
Service component

Fig. 4 The ESF architecture in the structural view: a set of service components and a set of user interaction components connected to a software bus

- A plug-in mechanism, for the purpose of binding client requests (imported definitions and operations) to services (exported definitions and operations). The plug-in mechanism is used statically at component installation time, when sessions are established or dynamically as requests are executed.
- A communication mechanism for the purpose of exchanging control and data.

Tools and, at a larger level of granularity, work contexts, may be dynamically established and configured through bindings between user interaction components and (sets of) service components.

Management of the Programme

The consortium saw the need for a distributed operation to enable the centres of expertise in Europe to be connected within the framework, for whatever period of time is necessary. Co-ordination and direction required central facilities. Consequently, the consortium has adopted a distributed project organization with a central headquarters, to provide the management framework for the project. Major requirements of the organization are to manage information flow and the overall work-programme. A simple view of the organization is shown in Figure 5.

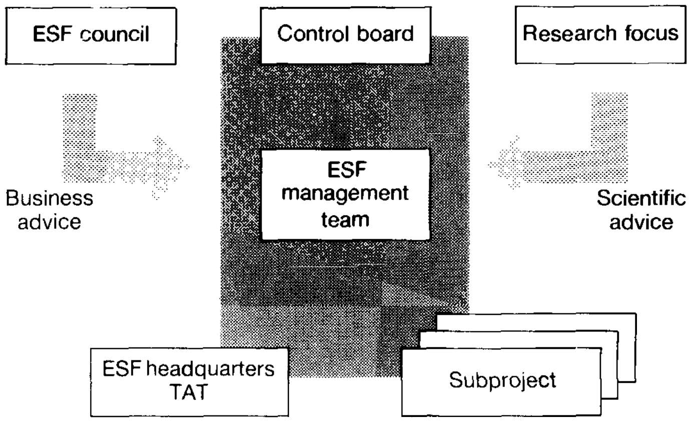


Fig. 5 The ESF project organization

Work is organized into sub-projects in which sub-consortium members combine to address specific work areas. Central management and technical co-ordination is from the Technical and Administrative Team (TAT) which is housed in Berlin. The central group is resourced at around 10% of the overall project resource of 280 staff in 1990. The distributed operation is established across 24 sites and connected through a systems network to a central configuration of advanced work-stations. There are seventeen sub-projects. The TAT manages a detailed work breakdown onto which specific company interests are mapped. Committed work is contracted to sub-projects.

Decisions concerning policy and strategy are made by the Control Board which meets formally four times a year. The chairman of the Control Board is additionally the official spokesman or "Speaker of the Control Board". This appointment is by election each year. Responsibility for the day-to-day operational management of the consortium rests with the Project and Technical Directors and a Market Integration Manager. The operational management team is extended to comprise three managers in the central headquarters and a manager from each company who is responsible for that company's day-to-day ESF activities. This group is dedicated to the project and meets formally 7–8 times per year. There are two advisory boards, the ESF Council and the ESF Research Focus. The ESF Council comprises company directors and general managers from the member companies and guest members from leading software-dependent corporations. This group meets each year to react to an overall statement of ESF project strategy, policy, plan, achievements and perceived trends in the industry. The ESF Research Focus comprises software engineering authorities of world-wide recognition. These experts meet with ESF leaders in seminars and workshops to examine research trends and solution strategies in the field of advanced software development environments and factories, and undertake specific consulting assignments.

To summarise this description of the programme, we describe our consortium strategy as a joining of European forces in the area of software production:

In one area we join forces to create the technology base, which involves the following:

- Jointly defining and promoting standard definition
- Jointly supporting technology development
- Jointly preparing market awareness of new technologies

In the second area, we shall join forces to approach the market:

- Jointly announce a coherent product range
- Jointly demonstrate a coherent product range

ICL's strategy for software engineering in 1987, established 3 main threads: an engineering improvement programme, the ALVEY/IPSE2.5 project and the need for a strong presence in a critical mass European project to achieve cost-effective integration of the components of advanced environments. The goals of the strategy are to improve the control of quality of delivered software systems and to improve productivity of the processes.

ICL will contribute to four areas in ESF: technology development and exploitation in process support, technology development and exploitation in component integration, quality and productivity measurement and large-scale trials of constructed factories in established development organizations. Participating companies in the group are Systems Software Product Group in ICL Computer Products division and the Information System Division of STC Technology Ltd. Further details from Roger Oakden, STC Technology Ltd, Copthall House, Newcastle-under-Lyme, Staffs. The member companies of ESF contribute to ESF in a wide variety of areas drawing on considerable industrial and research experience. Further details of the total ESF operation can be obtained from the ESF Headquarters, Hohenzollerndamm 152, D-1000 Berlin 33.

References

A complete list of references on ESF can be obtained from the project secretariat (ESF Headquarters, Hohenzollerndamm 152, D-1000 Berlin 33, phone +49-30.820.90.30.) General principles of the project are presented in some detail in the following papers and documents:

1 ESF Project: The ESF Technical Reference Guide, ESF project technical report, 1989.
2 ESF Project: The ESF Populated Mini, Presentation Manual, 1990.
3 FERNSTROEM, C., OHLSSON, L.: ESF – an Approach to Industrial Software Production. In: "Software Engineering Environments – Research and Practice", ed. K.H. Bennett, Ellis Horwood Books in Information Technology, 1989.
4 HUBERT, L., PERDREAU, G.: Software Factory: Using Process Modeling for Integration Purposes, First International Conference on Systems Integration, Morristown, April 1990.

A Spreadsheet with Visible Logic

Vincent West & Edward Babb
Strategic Systems, ICL Bracknell, Berks

Abstract

This paper describes a prototype logical spreadsheet developed as part of VisiLog, a EUREKA project involving ICL and BULL. The goal of VisiLog (Visible Logic) is to make the full power of logic available to the end-user through easy-to-use graphics.

The paper covers the capabilities of the spreadsheet, illustrating by examples its benefits over a conventional spreadsheet, which include generalised answers, goal seeking, handling of alternatives and inequalities, and (logic) programmability. It also describes the internal architecture, in particular the logical engine used, PLL (Pure Logic Language), which was the result of an ALVEY project involving ICL, Imperial College and Edinburgh University.

1 Introduction

The spreadsheet was originated by accountants as a way of presenting a set of figures in an orderly fashion. It consists of a rectangular array of *cells*. The past decade has seen increasing use of "electronic" spreadsheets on personal computers for budgeting and other financial applications. The electronic spreadsheet has the advantage that a cell can contain a mathematical *formula*, describing how the value of the cell depends on other cells and allowing the computer to recalculate it as they change. The user can make experimental changes (WHAT IFs) and see what the results would be.

Current state-of-the-art spreadsheets, such as WINGZ^{TM} from Informix Software Inc. (WINGZ, 1988), are very sophisticated systems with a multiplicity of presentation capabilities, but the underlying calculator is still essentially a *mathematical* one.

This paper describes a prototype *logical* spreadsheet developed as part of VisiLog, a EUREKA project involving ICL and BULL. The goal of VisiLog (Visible Logic) is to make the full power of logic available to the end-user through easy-to-use graphics. The prototype runs on a SUN workstation using X Windows.

Section 2 covers the capabilities of the spreadsheet, illustrating by examples its benefits over a conventional spreadsheet, which include generalised answers, goal seeking, handling of alternatives and inequalities, and (logic) programmability. Section 3 describes the internal architecture and section 4 the logical engine used, PLL (Pure Logic Language), which was the result of an ALVEY project involving ICL, Imperial College and Edinburgh University.

2 Capabilities

The VisiLog spreadsheet has the following benefits over a conventional spreadsheet:

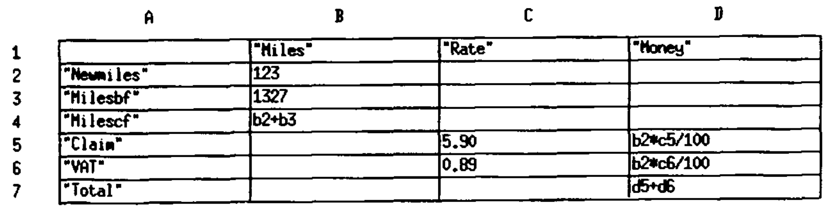
- *Generalised* answers – answers can be formulas as well as numbers
- Goal seeking – HOW TO questions as well as WHAT IF questions are handled
- *Alternative* formulas and answers
- Programmability – user-defined functions and databases
- Inequalities (like "greater than") as well as equalities

The following sections, with examples in part based on the ICL expense claim procedure for car mileage, will illustrate the basic operation of the spreadsheet and show these benefits. They do not exhaust the spreadsheet facilities – it can for example handle lists as well as numbers and strings.

2.1 Generalised answers

Fig. 1 shows the spreadsheet loaded with formulas, some of which are simply numbers or strings. You click on the CALC button to get the answers shown in Fig. 2. This of course represents a conventional use of a spreadsheet.

If you now delete the number 123 in cell B2 (Fig. 3), making B2 variable, the answers will be as in Fig. 4. Four of the answers are formulas (the ones for D5 and D6 could have been simplified further).



	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	b2+b3		
5	"Claim"		5.90	b2*c5/100
6	"VAT"		0.89	b2*c6/100
7	"Total"			d5+d6

Fig. 1 Conventional Formulas

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	1450		
5	"Claim"		5.9	7.257
6	"VAT"		.89	1.0947
7	"Total"			8.3517

Fig. 2 Conventional Answers

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"			
3	"Milesbf"	1327		
4	"Milescf"	b2+b3		
5	"Claim"		5.90	b2*c5/100
6	"VAT"		0.89	b2*c6/100
7	"Total"			d5+d6

Fig. 3 Conventional Formulas (Revised)

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"			
3	"Milesbf"	1327		
4	"Milescf"	(b2 + 1327)		
5	"Claim"		5.9	(b2 * 5.9 / 100)
6	"VAT"		.89	(b2 * .89 / 100)
7	"Total"			(d5 + d6)

Fig. 4 Generalised Answers

Generalised answers are useful when only *incomplete* information is available. They also provide *diagnostics* – in the above example you might have forgotten to give B2 a value.

2.2 Goal seeking

Suppose that you would like to "fix" the expenses to be exactly £9 instead of £8·35. To do this you add "&9" to cell D7 (Fig. 5). The spreadsheet complains "Contradictory formulas" as the answer cannot be both £8·35 and £9 at the same time. If you now delete the rate in C5 (Fig. 6), the answers (Fig. 7) show that it would need to be 6·42707 rather than 5·90. It would be nice if the

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	b2+b3		
5	"Claim"		5.90	b2*c5/100
6	"VAT"		0.89	b2*c6/100
7	"Total"			d5+d6 & 9

Fig. 5 Goal Seeking – Formulas

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	b2+b3		
5	"Claim"			b2*c5/100
6	"VAT"		0.89	b2*c6/100
7	"Total"			d5+d6 & 9

Fig. 6 Goal Seeking – Formulas (Revised)

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	1450		
5	"Claim"		6.42707	7.9053
6	"VAT"		.89	1.0947
7	"Total"			9

Fig. 7 Goal Seeking – Answers

spreadsheet could work back to B2 if you deleted it instead but this prototype cannot do that yet.

2.3 Alternatives

Suppose further that you are not sure whether you drove 123 or 124 miles. If you reset the formula in cell B2 to 123/124 (which means 123 OR 124) as in Fig. 8, there are now two sets of answers with different rates (Fig. 9 and 10). The first is the set of answers for 123 we have already seen and the second is the set for 124 (where the rate in C5 would need to be 6·36086). You use the

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123|124		
3	"Milesbf"	1327		
4	"Milescf"	b2+b3		
5	"Claim"			b2*c5/100
6	"VAT"		0.89	b2*c6/100
7	"Total"			d5+d6 & 9

Fig. 8 Alternatives – Formulas

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	123		
3	"Milesbf"	1327		
4	"Milescf"	1450		
5	"Claim"		6.42707	7.9053
6	"VAT"		.89	1.0947
7	"Total"			9

Fig. 9 Alternatives – First Set of Answers

	A	B	C	D
1		"Miles"	"Rate"	"Money"
2	"Newmiles"	124		
3	"Milesbf"	1327		
4	"Milescf"	1451		
5	"Claim"		6.36806	7.8964
6	"VAT"		.89	1.1036
7	"Total"			9

Fig. 10 Alternatives – Second Set of Answers

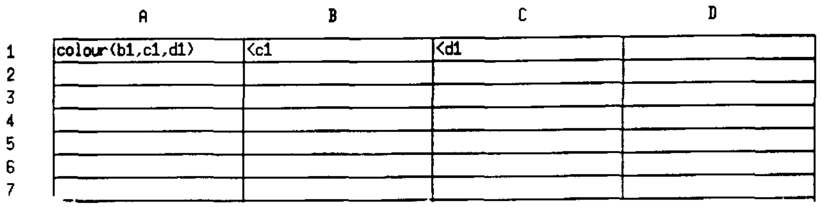
CALC button to step through the sets of answers or the QUIT button if you have seen enough.

2.4 Programmability – user-defined functions and databases

It is particularly important that the user can define his own functions and so databases (in PLL) and then use the FETCH button to fetch them for use in formulas. So for example you can write functions to convert between metric and imperial units. An example of a database is the X Windows colour database which consists of entries with four attributes – the colour name and the corresponding intensities in the range 0–255 for the red, green and blue guns (equivalent to a function whose arguments are the intensities and whose value is the colour name). For an illustration of its use see the following sub-section.

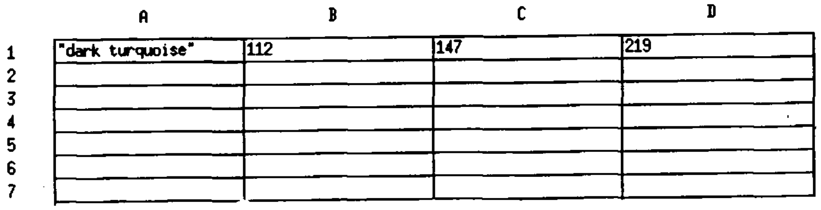
2.5 Inequalities

Suppose you wanted to use the colour database described in the previous sub-section to find the (blueish) colours where the guns are in increasing intensity. The formulas you would use are shown in Fig. 11. There are four such colours, the first being "dark turquoise" (Fig. 12).



	A	B	C	D
1	colour(b1,c1,d1)	<c1	<d1	
2				
3				
4				
5				
6				
7				

Fig. 11 Databases – Formulas



	A	B	C	D
1	"dark turquoise"	112	147	219
2				
3				
4				
5				
6				
7				

Fig. 12 Databases – Answers

3 Architecture

The architecture is shown in Fig. 13. There are three main components:

- the *User Interface* handles all communication with the user via X Windows and provides the environment for loading and saving spreadsheet formulas
- the *Symbolic Interpreter* handles all communication with the PLL engine, translating spreadsheet formulas into PLL queries and PLL answers into spreadsheet answers. It also provides the environment for fetching the user's PLL rules for functions and databases

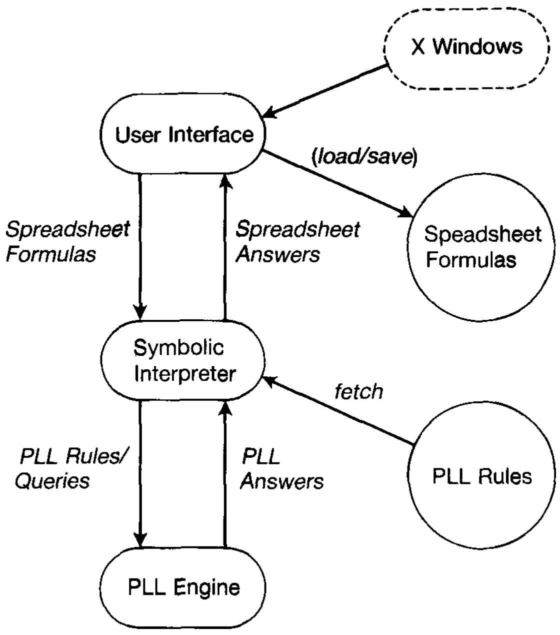


Fig. 13 Architecture

- the *PLL Engine* receives logical rules and queries and *rewrites* the queries to provide answers

4 The Pure Logic Language

For an introduction to PLL see Babb (1989a), and for a fuller description describing features not used by the spreadsheet see Babb (1989b).

PLL has two broad classes of rewrites:

- Atomic formula rewrites which apply to *atomic formulas* involving data structures such as numbers, strings and lists.
- Compound formula rewrites which apply to a *set of formulas.*

4.1 Notation

The *rewrite* symbol –R–> is used to show that something is rewritten using a rewrite definition. The greek letter ϕ is used to denote a PLL expression such as *(x=2* 3)*. The letter *c* is used to represent bound variables or constants such as the number *12* or the string "*abc*" or the list *[1,2]*.

4.2 *Atomic formula rewrites*

These rewrites apply to *atomic formulas* and try to reduce them to the solution *atomic formula* of the form x_1 *op* c_1 & ... & x_m *op* c_m where *op* represents a comparison operator.

Comparisons
The rewrites for the six comparison operators $= \neq > \geq < \leq$ can be illustrated by those for equality:

$c1 = c2$	-*R*->	TRUE if $c1$ equal to $c2$
	-*R*->	FALSE if $c1$ not equal to $c2$
$x = x$	-*R*->	TRUE

Arithmetic and Strings
The arithmetic rewrites are:

$c1 + c2$	-*R*->	the sum of $c1$ and $c2$
$c1 - c2$	-*R*->	the difference of $c1$ and $c2$
$c1 * c2$	-*R*->	the product of $c1$ and $c2$
$c1 / c2$	-*R*->	the quotient of $c1$ and $c2$

The string rewrites are:

$c1 + c2$	-*R*->	the concatenation of $c1$ and $c2$

These rewrites do not include the use of associative or distributive rules, but constants on opposite sides of a comparison operator are combined.

Lists
The list rewrites are:

[x1,x2, ...] = head :: tail -*R*->	x1 = head & [x2, ...] = tail
[x1,x2, ...] = [y1,y2, ...] -*R*->	x1 = y1 & ...

4.3 *Compound formula rewrites*

Compound formula rewrites operate on sets of atomic formulas.

Conjunction
If an atomic term in a conjunction is reduced to a binding for a variable, it is then available to any other atomic term involving that variable.

The rewrites for expressions involving TRUE or FALSE are:

ϕ & *TRUE*	-*R*->	ϕ
ϕ & *FALSE*	-*R*->	*FALSE*

Disjunction
The rewrites for expressions involving TRUE or FALSE are:

$$\phi \mid TRUE \quad \text{-}R\text{->} \quad TRUE$$
$$\phi \mid FALSE \quad \text{-}R\text{->} \quad \phi$$

Predicates
The expression:

<predicate name> <arguments>

is rewritten using the built-in definition for the predicate or the user definition:

define <predicate name> <parameters> <guard expression> tobe ϕx

with the arguments being substituted for the parameters. The guard expression prevents the rewrite if the guarded variables are not bound.

5 Conclusion

The spreadsheet user can be given capabilities beyond the scope of conventional spreadsheets by an architecture which uses a logical engine instead of the conventional calculator.

References

E. BABB (a): *Pure Logic Language.* ICL Technical Journal, May 1989.
E. BABB (b): *An Incremental Pure Logic Language with Constraints and Classical Negation.* To be published in: *Expanding the Horizons: Proceedings of the 1st UK Logic Programming Conference*, 1989, ed Dodd, T, Owen, R and Todd, S Intellect Ltd, 1990.
WINGZ User's Guide. Informix Software, Inc., 1988.

Intelligent Help – The Results of the EUROHELP Project

Mick Smith
ICL, Strategic Systems, Manchester, UK
Colin Tattersall
Computer Based Learning Unit, University of Leeds, UK

Abstract

Intelligent help is distinguished from conventional help in three important ways. First, help is generated from a representation of the application. Second, and because it is generative, it can be dynamically adapted to the knowledge of the user, the task in hand, and the state of the application. This is in contrast to conventional help where the emphasis has been on access mechanisms to prestored texts. Finally, as a result of empirical and scientific studies of the nature and requirements for help, it is designed to meet the real needs of the user.

The approach to the provision of intelligent help as developed by the EUROHELP[1] project requires four major components: An Intelligent Help System Shell (IHSS) which forms the generic run-time system, an Application Modelling formalism for representing an application to the IHSS, a Help System Development System (HSDS) to support the construction of Application Models, and finally a Help System Development Methodology (HSDM) for guiding the whole process.

This paper is the first of a series of papers on the provision of intelligent help and its impact on application development. It describes the lessons learned from the EUROHELP project with respect to the nature of intelligent help, the potential benefits to be gained from such help and practical techniques for its provision. Later papers will address the relationship between application development and the provision of help.

1 Introduction

Making the transition from a limited understanding of a computer system to a level of expertise permitting efficient accomplishment of tasks is not

[1]EUROHELP was an ESPRIT project (number 280), 50% funded by the European Commission, which ran from November 1984 to July 1990, and involved some 100 man years of effort. The partners in the project were CRI A/S, now Axion A/S (Denmark), Courseware Europe BV (Netherlands), ICL Knowledge Engineering Buiness Centre, now ICL Strategic Systems (UK), University of Leeds CBLU (UK) and the University of Amsterdam (Netherlands). The Dansk Datamatik Centre (Denmark) were a subcontractor for the first half of the project.

straightforward. Providing a help facility with an application is one way of bridging the gap between utility and usability. However, the quality and sophistication of help systems for application programs has not mirrored advances in software development. With the bulk of the effort in the production of a piece of software devoted to supporting the functionality of the system, help facilities tend to be viewed as outside the software engineering process – few textbooks on the subject feel it warrants a mention. Traditionally, help systems have been produced by software designers towards the end of the software engineering life cycle, and the style of such help provision has led to the criticism that help systems are often only useful to those with a detailed knowledge of the application. The EUROHELP project aimed to address the well-documented (eg [Hou84]) deficiencies of conventional help systems and show how it is possible to provide better help automatically.

1.1 Conventional help system technology

The Unix[2] manual is typical of help systems for software packages. The facility is an online reference source giving details of the various commands and applications available under the Unix operating system.

Manual entries are accessed from the Unix shell using the **man** command, which takes as argument the name of a command or facility in the Unix system. This accessing represents the first problem: a user unfamiliar with the idiosyncrasies of this particular application may have difficulty alerting the system to his or her need for help.

For the user with some knowledge of Unix, there exists a handful of tools which help in locating particular commands. An example is **apropos**, which uses keyword-lookup on a database of command-task associations to cross reference manual pages. However, being aware of the command is not enough to ensure reasonable help. The user is also required to know the appropriate vocabulary in which to describe the task. Discovering how to invoke the c compiler can be achieved by typing **apropos "c compiler"**. Typing **apropos c compiler** includes this information with over seven hundred other references.

One of the criticisms levelled at Unix is that the availability of help varies within particular subfacilities. An example is the Unix line editor, *vi*, which has no associated help facility. Although help may be accessed by suspending the current process and examining the manual, or by issuing a Unix shell command from within the editor, both require a high degree of familiarity with the system. Furthermore, the help produced concerns invocation of the editor from Unix rather than how to use the subsystem. While the electronic mail system makes help available, it uses a different keyword (?) and produces a terse description of the mail commands together with their

[2]Unix is a trademark of A.T. & T. in the USA and other countries.

arguments. Thus, the user is forced to remember not only whether help is available with the subsystem, but also which command is associated with requesting help.

In an attempt to alleviate some of the problems associated with signalling for help, some systems provide a menu of options containing a *help* entry. When selected, this option produces a screen of help text. Such menus may be on permanent display, or may appear (pop-up) when the user presses a special help key. The inclusion of this type of facility provides a beacon to alert the new user to the availability of help, and relieves the burden of shooting in the dark for the help keyword. An example is the spreadsheet Multiplan [Lim82] which has a portion of the screen displaying the commands from which the user may select.

Though such an approach assists the user in signalling for help, it does not solve the problem of specifying what the help should address. These difficulties were examined by O'Malley [OMa85] who noted that, in conventional systems, users must know both what their problem is and how to express it in the terminology of the help facility [Mor83].

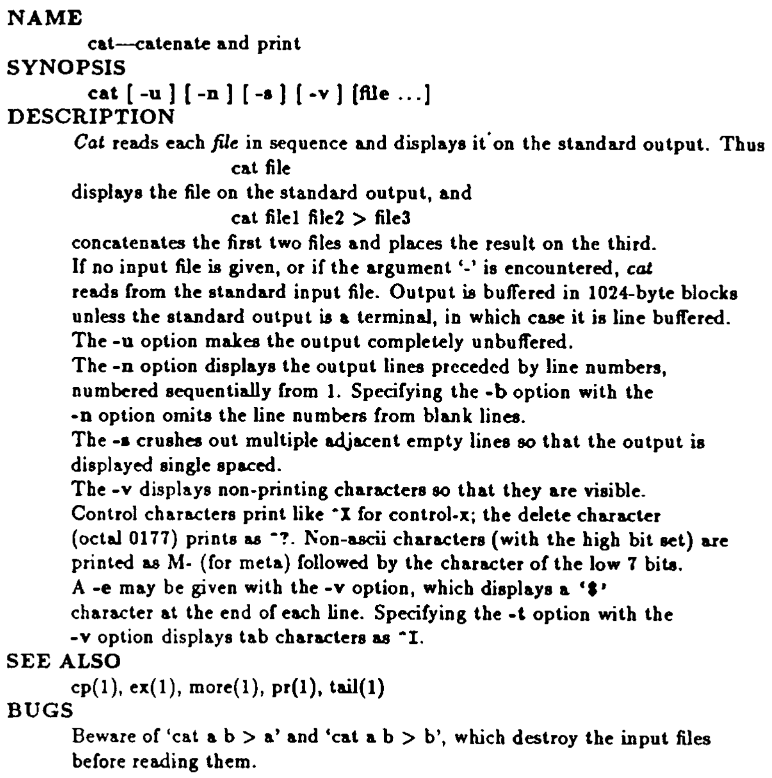
More sophisticated systems have begun to emerge which assist the user in formulating a query using advanced user interface techniques. The **Andrew User Interface Toolkit** [RMH88] for example, includes a front-end to the Unix manual using hypertext-like techniques [Sha88]. The system provides many help files in addition to the usual Unix manual texts, and requesting help (by typing the command or selecting it from a menu) results in a "guided tour of the Andrew System" which explains the available facilities. The user may select so-called *active* text, which is sensitive to mouse clicks and which produces a pre-stored help text for the selected item. Browsing menus of related topics is also allowed, permitting questions at both a general level (*Mail, Printing, Documents, Using Other Computers*), and at the level of individual commands (**cat, mkdir**, **pwd**).

1.2 Help from the conventional approach

While the process of requesting help has seen major developments over the recent past, the output from the system as a result of this request remains the same. One of the aims of EUROHELP was to show that the traditional approach to help production contains fundamental flaws concerning the ability of a help text to address a user's actual problem. The Unix manual will serve to illustrate this point.

Manual entries vary in size and detail from several lines of general outline (eg **man cb**) to several pages of detailed descriptions (eg **man csh**). An example entry, produced in response to the command **man cat**, is reproduced in Fig. 1.

Each manual entry has several parts, beginning with the name of the command and a brief outline of its purpose, followed by a synopsis



```
NAME
        cat—catenate and print
SYNOPSIS
        cat [ -u ] [ -n ] [ -s ] [ -v ] [file ...]
DESCRIPTION
       Cat reads each file in sequence and displays it on the standard output. Thus
                      cat file
       displays the file on the standard output, and
                      cat file1 file2 > file3
       concatenates the first two files and places the result on the third.
       If no input file is given, or if the argument '-' is encountered, cat
       reads from the standard input file. Output is buffered in 1024-byte blocks
       unless the standard output is a terminal, in which case it is line buffered.
       The -u option makes the output completely unbuffered.
       The -n option displays the output lines preceded by line numbers,
       numbered sequentially from 1. Specifying the -b option with the
       -n option omits the line numbers from blank lines.
       The -s crushes out multiple adjacent empty lines so that the output is
       displayed single spaced.
       The -v displays non-printing characters so that they are visible.
       Control characters print like ^X for control-x; the delete character
       (octal 0177) prints as ^?. Non-ascii characters (with the high bit set) are
       printed as M- (for meta) followed by the character of the low 7 bits.
       A -e may be given with the -v option, which displays a '$'
       character at the end of each line. Specifying the -t option with the
       -v option displays tab characters as ^I.
SEE ALSO
       cp(1), ex(1), more(1), pr(1), tail(1)
BUGS
       Beware of 'cat a b > a' and 'cat a b > b', which destroy the input files
       before reading them.
```

Fig. 1 An example Unix manual entry

illustrating its syntax. Most commands in Unix permit several variants with slightly differing effects, these being invoked by supplying flags as arguments to the command. The synopsis indicates the available options together with ordering constraints. A description of these options follows, forming the majority of the entry, then a list of other related Unix commands with manual entries. Finally, any bugs associated with the implementation are noted. Although the partitioning of individual manual entries varies, the **cat** example can be considered representative.

The help is expressed in highly technical language – bytes, octal, non-ascii characters and buffering are terms commonly found in the computing literature but scarcely elsewhere. The description is couched in Unix terminology, with the notions of standard input, standard output and redirection all referred to in the examples. This makes knowledge of the way Unix functions a prerequisite to understanding the text, while the synopsis requires familiarity with the use of square brackets to denote optionality and dots to denote continuation. The first criticism to be levelled at the entry is that it is highly technical in nature.

The entry is also pitched at fine level of detail. Buffering output into 1024-byte blocks and the special handling of non-printing characters are both associated with specialised tasks requiring a high degree of Unix knowledge. This represents the second criticism: the entry is designed to provide as much information as possible in a single text for all users.

Some help systems have attempted to separate the fine detail from a general description by providing a hierarchy of related help texts. An example is the VAX VMS help facility [Equ80] which presents a general description followed by a sequence of options allowing the user to request further details. As with a Unix manual entry, the text shows the purpose of the command together with a general description of its syntax. However, a portion of the screen is used to display options for which further information is available.

While the text still contains several technical terms, it pushes virtually all the detailed description to a deeper level of the hierarchy, shielding it from inexperienced users. Though common, this style of help facility poses its own problems, particularly for the experienced user. Locating a detailed piece of informtion pertinent to a specialised task may require a series of descents and ascents through the hierarchy.

A severe shortcoming of both texts is the virtual absence of information relating to the tasks the user is likely to want to achieve with the application. **cat** is described as being used to "concatenate and display", a very general, though still technical, indication of the purpose of the command. In practice, **cat** is used either to display the contents of a file on the screen, to feed the contents of a file into another command or to concatenate a number of files. These tasks will only roughly correspond to those in which the user is (mentally) engaged. Most manual entries describe the effect of issuing the command upon the system, rather than why that effect might be desired.

A more fundamental problem with help produced in the traditional way is that, by its nature, texts cannot be tied to the state of the system. This means that advice cannot indicate specific actions to be taken in particular contexts. If, for example, an attempt is made to examine the contents of file *test* for which the appropriate read permissions are not set, the system will print the error message:

```
test: Permission denied.
```

Typing **man cat** will find no reference to this precondition, nor to the operator which could enable the read (**chmod**).

Furthermore, pre-stored help texts cannot differ with respect to the knowledge of the user. Those familiar with the application receive exactly the same text as those who have never used the system before. The advice required by the two differs widely, as it does for users between the two extremes. While some users will know what the system can do, how this can be achieved and how the system supports this functionality, others will know little or nothing of these matters. Providing a single text to address all problems for all users inevitably only provides some solutions for some users. Such help texts are either too complex for inexperienced users to understand, or not detailed enough for those with specific problems.

All the above criticisms of the Unix manual and related help systems stem from the fundamental limitations associated with static canned text. Independent of any contextual information, it can take no account of a user's current task, level of knowledge of the application or current state of the system, and so provides help in isolation from the context of the request.

1.3 Intelligent help systems

If help systems are to address actual rather than perceived user needs, a radically different approach to their design must be adopted. Though most commercially available systems employ the well-tried but flawed techniques described in the previous section, an active line of research has been pursued in the area of Intelligent Help Systems (IHSs). Drawing from the field of Artificial Intelligence (AI), the work aims to produce help systems better able to assist the user in learning and manipulating applications programs.

Perhaps the best known exemplar of this approach is the Unix Consultant (UC), originally developed at Berkeley in the early eighties [Wil82]. The system design examines many issues in AI, including knowledge representation, natural language processing and planning.

Most effort on UC was initially directed towards allowing the user to question the system in natural language over simple dialogues. In an extended version of the earlier paper, Wilensky et al. [WAC84] describe the various components and processing stages of UC. A user may query the system about the various concepts in the domain and about how tasks may be accomplished. The authors emphasised that the research was not an exercise in user interface design to provide a better front-end to Unix, but that UC should allow "better use [of] the Unix environment in which [it] is embedded". To facilitate this process, six modules were designed and integrated into a question-answering system:

- a language analyser
- a context model
- a planning component
- a goal analyser
- a language generator
- a system for the acquisition of additional domain knowledge into the representation

A user question input to UC is parsed into an internal conceptual representation by a component known as PHRAN (PHRasal ANalyzer). The resulting internal meaning serves as a query mechanism to the underlying knowledge structures, described in detail by Chin [Chi83]. This domain representation holds information about the Unix domain. A planning component, operating on the knowledge base, attempts to find a plan to accomplish the user's stated or inferred goal. When a plan is selected the corresponding precondition frames are verified against the actual system state – UC is allowed to probe the user's environment rather than continually having to emulate the effect of user actions – and failed preconditions are highlighted to the user. The answer is articulated by a component known as PHRED (PHRasal English Diction), which shares the same linguistic knowledge as PHRAN and which converts the conceptual representation into natural language.

The following is an example of UC's output in response to the question of *how to change the read premission of a file.*

```
Use chmod.
For example, to add group read permission to the file
name foo, type 'chmod g+r foo'.
```

Despite some drawbacks, UC represents the most developed work into an IHS, and the most recent description [WCL*88] shows further work in planning and knowledge representation. The work is seminal in its identification of the modules needed to support work on IHSs:

- an input mechanism for a user's queries
- a plan generation component to respond to task-based queries
- a model of the user's knowledge
- a text construction module
- an output generator to phrase replies to questions

These modules rest upon a knowledge representation formalism which must encode relevant information about the domain. The modularisation can support the user's working, but has little to offer with respect to the user learning about the system. These difficulties have typically been overcome by hand-crafting appropriate teaching information into the domain representation.

1.3.1 From passive to active assistance By monitoring the user's input, it was a small step to have a system volunteer help when deemed appropriate. This approach was taken by Shrager and Finin [SF82] in their help facility for the VAX/VMS operating system. The research attempted to ease some of the problems of help provision by removing the requirement that users initiate the process by asking a question. The system recognised "correct yet inefficient command sequences" and advised on how the associated task could be accomplished more efficiently, using a library of "bad plans".

The system stored the command sequences to be recognised, together with an action (typically the presentation of a help message to the user) on the last command in the sequence. The help messages "either provide immediate advice or a pointer to a manual or on-line HELP entry". The work was continued by Finin [Fin83] who claimed that IHSs should

- be active rather than passive
- contain an explicit model of the task, user and system
- engage in an interactive dialogue with the user to identify his or her needs

Finin was particularly concerned with assisting users who either do not know they need help or who do not know how to ask for help. He designed a system called WIZARD as the first step towards one which could address six stages in the provision of help:

- is help needed?
- what help is needed?
- what help can be given?
- of this, which best matches the user's needs?
- how can this be recognised?
- does the user understand this help?

WIZARD examined the first of these problems "recognising certain situations in which the user may need help or advice", using the same mechanism as the earlier work.

Fischer, Lemke and Schwab [FLS85] published a more detailed examination of the nature of active help. The authors noted that "contrary to tutorial systems, help systems cannot be structured in advance but must 'understand' the specific contexts in which the user asks or needs help". A diagram, shown below in Fig. 2, illustrates the role of help systems, both active and passive, in terms of users' knowledge of application programs.

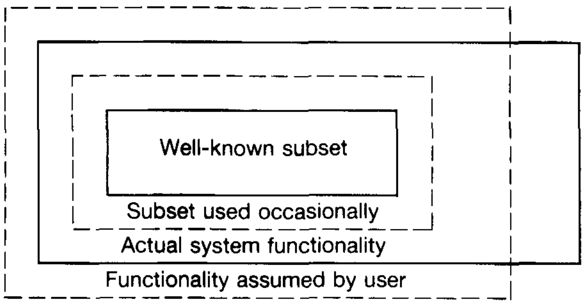


Fig. 2 User's knowledge of application software

For any particular system, each user utilises both a well-known subset of the available functionality and a less-well understood subset. In addition, the authors claim users carry a "mental model" of the system, perhaps only partially realised by the actual application. The authors see passive help systems as enlarging the well-known subset by responding to queries concerning the less-well known portion. Active help systems are placed in the role of informing the user about those commands and entities whose existence is entirely unknown to the user, and about which a query cannot be formed.

Interest in helping users of the Unix system was such that a workshop entitled "Knowledge representation in the Unix help domain" was organised during 1987 at which several IHSs were described [NWW87]. The Yucca-II system [Heg87] incorporates a natural language parser to handle user questions about the Unix command language. Based on the earlier UCC system [DH82], it permits two classes of query: process queries and conceptual queries, using a separate "solver" to deal with each. The former corresponds to *How do I ...?* questions while the latter seeks definitions and descriptions of Unix entities. Yucca-II makes use of so-called clichés (e.g. redirection of output and pipe connection) to rewrite the command sequence into an interconnected command line which is presented to the questioner. The user is permitted to ask exploratory questions (*What happens if...?*) and plan failure questions (*Why didn't ... work?*), the query passing to a "simulator" to be emulated in an attempt to discover the answer.

The AQUA system was designed as "a computer program intended to model the process of problem understanding and advice giving of a typical computer consultant". AQUA takes a natural language description of a problem concerning the use of Unix and provides a natural language solution, "together with an explanation for any previous failed attempts by the user".

The authors illustrate the process using an example referred to as Stubborn File, in which a user states:

> *I tried to remove a file with the* **rm** *command. The file was not removed, and the error message was permission denied. I checked and I own the file.*

The system not only models a failed user goal (removing a file) and failed user plan (using **rm**), but also a user hypothesis (owning a file is a precondition for removing it) and satisfied goal (of verifying the file's ownership). This information is used to generate its response:

> *To remove a file, you need to be able to write into the directory containing it. To remove a file, you do not need to own it.*

This response is contrasted with the capabilities of (the early versions of) UC which, when asked how to remove a file, indicated that **rm filename** would achieve the goal. The system could only talk in terms of general plans for accomplishing general tasks, not in terms of specific situations such as precondition violation.

The Sinix[3] Consultant (SC) Project [Kem87] is an attempt to "assist beginners in getting acquainted with the Sinix system as well as to offer appropriate and useful information to the advanced user". Again incorporating a natural language front-end, the system offers both active and passive help, the two styles being handled by a Plan Recogniser/Adviser and Answer Formulation Module respectively.

Central to the project is the Sinix knowledge base, a hierarchy of frames describing Sinix entities and accessed by the query language KB-Talk. The emphasis is placed on representing commands, which feature as the leaves of a hierarchy of Sinix actions. More general than commands, these actions model the functionality of the system. A less well-developed hierarchy exists to capture the objects manipulated by the commands.

SC contains a parser of German allowing the user to pose a variety of question forms concerning objects and actions. The parsing of a query upon an action is done in relation to the action hierarchy, with the Question Evaluator "determining the most specific concept which is addressed in the user's question". Queries are answered by the Answer Formulation Module, described in further detail in a project progress report [HKN*88]. The Module provides "tutorial explanations depending on the user's expertise concerning the Sinix system".

1.3.2 Conclusions to be drawn from the IHS literature Research into IHSs carried out over the past eight years has identified the major components and representational requirements for provision of intelligent help. The first and most fundamental decomposition is that based around the distinction between active and passive assistance. Though much of the information required by each is shared, active help requires intelligent monitoring of the user's interactions with the application program. Passive help requires some mechanism for interpreting the user's queries, either at a general level or referring to specific objects in the system state. This can be the type of natural language front-end used by many IHSs or some form of form-filling or menu driven interface. Both types of assistance require the help system to keep track of the state of the application, a fact often neglected in IHS research, but one which is crucial in providing appropriate advice. Tracking the system state points to another split in the architecture of IHSs. If the help system is produced in isolation and without reference to the source code of the application (an *add-on* help system), then it must maintain its own version of the state of the application and emulate the effects of user interactions upon it. If, however, the system is able to examine the actual state of the application program (an *integrated* help system), the complexity of the task is dramatically reduced since no such maintenance or emulation is needed.

A planning module (which must also reference the current state of the system) is shared by both styles of help. Passive assistance uses such a component to

[3]Sinix is a derivative of Unix developed by Siemens AG.

provide answers to *how do I ...?* questions, while active help uses it to generate alternative plans to the user's own.

An adequate model of the capabilities and knowledge of the user must be maintained so that advice dispensed is sensitive to each individual user's particular situation, whether given by active or passive component.

Both components must share the capabilities of a response formulation module, able to construct a text to address a user's problem.

However, the literature reveals that below all these modules is the most crucial and complex aspect of Intelligent Help Systems, an adequate representation of the application. This must capture the tasks which can be achieved with the software, how the user may achieve these tasks, and how the system supports this functionality.

The literature is clear on this modularisation of the help giving process. It is also united in calling for the dynamic generation of help texts, but offers few solutions, with most systems adopting some form of pre-storing of answers. Furthermore, the IHSs leave unaddressed issues associated with facilitating the learning of IPSs through help provision.

IHSs should be in a position to help the user over immediate hurdles as and when they arise – difficulties with particular concepts, lack of knowledge concerning a particular method plan or in the interpretation of the working of the application. Most systems described above are in a position to accommodate these requirements, although the third style of assistance, that of explaining the state and workings of a system, has been little addressed. Beyond this, as indicated earlier, the IHS must attempt to advance the user's understanding of the application in terms of what is known and what has successfully been done before, an aspect virtually unaddressed by the systems described in this review.

2 The Approach of EUROHELP Project

2.1 EUROHELP's Objectives

The EUROHELP project had the following main objectives:

- To develop cost-effective approaches for the introduction of users to the use and full exploitation of Computer Systems
- To develop a proven and established methodology for integrating help systems with computer systems
- To develop an environment and/or set of tools to support the established methodology
- To produce a generic Help Facility employing state-of-the-art AI techniques for industrial environments

At the end of the project significant progress had been made toward these objectives with most concepts and techniques being demonstrable in a working prototype Intelligent Help System Shell and tools to instantiate it for a specific application. However there is still some work to be done before the techniques developed can be employed on a regular basis in applications development. The major issues to be addressed now are the cost-effectiveness and practicability of the techniques developed.

2.2 EUROHELP's architecture for the provision of help

In order to accomplish these objectives the project has produced four main deliverables:

The IHSS – The Intelligent Help System Shell which embodies all application independent knowledge of *how to give help.*

The AM – The Application Modelling Formalism which provides a means of describing specific applications to the IHSS.

The HSDS – The Help System Development System which enables and facilitates the construction of specific Application Models.

The HSDM – The Help System Development Methodology which provides the guidelines for the developer to construct an application which incorporates Intelligent Help.

These are shown in Figure 3. First there is a generic **Intelligent Help System Shell** (IHSS) which embodies all the *application-independent 'know-how'* for providing on-line, adaptive help. A generic shell has been achieved by factoring out all representations of the application-specific knowledge into an **Application Model** (AM). Each application requires its own Application Model but *no* modifications to the Shell. The construction of an Application Model is not a trivial task and for this reason a **Help System Development System** (HSDS) has been developed which is a collection of editing and browsing tools specifically designed to support the construction of application models which are complete and consistent in so far as is technically and humanly possible. To guide the whole process a **Help System Development Methodology** (HSDM) has been produced to assist the application developer in the construction of an application model. Over and above these four main components the project has set down the requirements on the interface between the Shell and the Application.

2.3 The nature of intelligent help

EUROHELP set out with the mission of adding some 'intelligence' to the provision of help by

- employing natural language generation techniques in place of using pre-stored text – this is a necessary pre-requisite to any form of intelligent adaptation.

- making use of contextual information about the application state and the current user task, in order to determine
  - when help is required,
  - the specific 'need' of the user,
  - and the appropriate content and form of the generated help text.
- modelling the user's knowledge of the application.

To accomplish this the project had to draw upon various techniques, particularly from the field of Artificial Intelligence, ranging from Plan Recognition and Planning, Knowledge Representation, Intelligent Teaching Systems, Question Answering, to User Modelling. In addition the project undertook a number of empirical studies in order to capture the real nature of help requirements. From analysis of these studies and from examining the literature the project distinguished between two basic forms of help namely Passive and Active help:

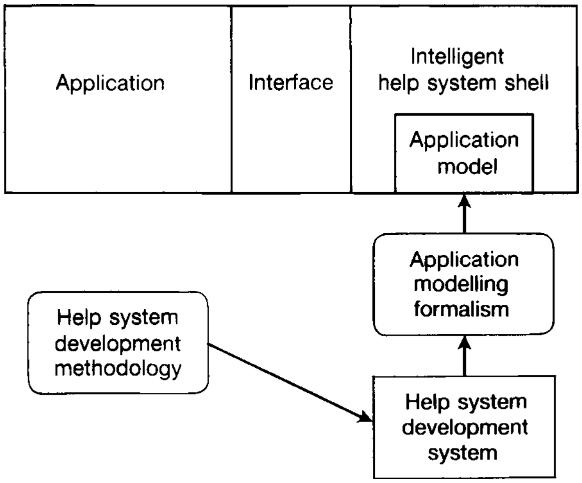

Application
Interface
Intelligent help system shell
Application model
Application modelling formalism
Help system development methodology
Help system development system

Fig. 3 EUROHELP Major Deliverables

Passive Help – This comprises the capability of answering questions, some of which can be asked and answered at the top level[4], and some of which can be asked/answered in a follow-on situation. In addition some questions can be answered in both specific and general terms.

Enablement – This is a top level question in which the user can ask a *"How do I do ...?"* question. This can be answered in general terms, or with reference to specific objects, in which case a Planner is employed to generate a plan. This latter capability is certainly not found in existing help systems and can often enable the completion of a task without the need to pause to consult a manual or an expert.

[4]Here top level means that the user has just switched from interacting with the application to ask a question.

Elaboration – This is both a top level question and a follow-on question and allows the user to ask *"What is ...?"* or *"Tell me about ...?"* questions. This capability obviously provides similar texts to those which would otherwise have to be pre-stored, but can be adapted to user knowledge and previous learning. This is a fundamental question type and supports many forms of learning about the application.

Evaluation – This is a top level question which allows the user to ask a *"What happened?"* question. Again this is not supported in existing help systems simply because the required knowledge is not available. It also supports the user in continuing with his task.

Comparison – This allows the user to ask *"What is the difference between ...?"* type questions between one command and another, and between objects. It is always used in a follow-on to a top-level question. This capability is not found in conventional help systems, and supports the 'natural' learning patterns observed in empirical studies.

Exploration – This is a follow-on question to an Evaluation question and allows the user to ask *"What could I have done?"*. Again this is not supported in existing help systems simply because the required knowledge is not available, and is designed to support task continuation.

Continuation – This is again a follow-on to Evaluation, allowing the user to ask *"How do I undo ...?"*. Again this is not supported in existing help systems simply because the required knowledge is not available.

Active Help – This form of help requires the help system itself to take the initiative and intervene to offer help in appropriate circumstances. This is not typical in existing help systems, although most applications attempt some form of help in error situations. The Active Help component of EUROHELP provides the following forms of help:

Feedback – Here the help system will intervene and offer a short explanation of the effects of what the user has just done. This is typically done by the application itself but not usually in a coherent and consistent fashion, nor in a user/context sensitive or educationally constructive manner.

Remediation – Here the help system recognises that something is wrong, for example a syntax error or the use of an inefficient plan, and intervenes with an explanation of what was wrong, and/or an unknown effect, and where appropriate an undo/alternative plan. Typical help systems do not have this capability since it requires application state knowledge, plan recognition, and planning. This form of help again specifically addresses the need of the user to continue working.

Expansion – Here the help system recognises an opportunity to introduce a new facility to the user. This will only occur if the new facility is relevant to what the user is currently doing, within the *range of understandability* of the user, and also set as a didactic goal. This provides a form of on-the-job training specifically aimed at broadening the knowledge of the user.

2.4 The intelligent help system shell

As indicated earlier to provide such Active and Passive help functionality a help system requires certain additional abilities. These include

Planning – in order to work out what the user *should* be doing to accomplish a particular task and to answer *'How'* questions.

Plan Recognition – to determine what task the user is currently undertaking, and also the method being employed.

Diagnosis – having determined what the user is doing, and observed an inefficiency and/or mistake, it is necessary to determine the underlying *lack of knowledge* and/or *misconception* of the user, i.e. the particular local need of the user.

User Modelling – to undertake Plan Recognition, Diagnosis, and help text generation it is necessary for the help system to *'know' the strength of understanding* of each concept which could feature in an answer, and the performance strength associated with each task.

Application Modelling – underlying all the above capabilities is the need for a representation of the application itself, ranging from the tasks for which it can be used, to the effects of each unit of functionality, to the syntax of its commands.

Application Emulation – not only is a static description of the application required, but it is also necessary to model the dynamic application state.

Natural Language Generation – once the content and structure of help has been determined a Natural Language Generator is required to transform it into readable text.

All of the above topics plus the nature of Active and Passive have been researched in the course of the EUROHELP project, and valuable results are reported elsewhere ([Bre90]).

Figure 4 shows how all the above techniques and capabilities are combined into an architecture for an Intelligent Help System Shell. Owing to the complexity of Figure 4, it does not show a module for Application Emulation, nor an Application State Representation. These should be taken as read.

In order to understand this architecture it is worthwhile examining the normal cycle of events when the user uses an application for which help has been provided. At the top of Figure 4 one can see that the user communicates directly with the application in the normal manner. There is a basic requirement in using the IHSS

> *that all input and output, but particularly input, can be intercepted and monitored by the IHSS.*

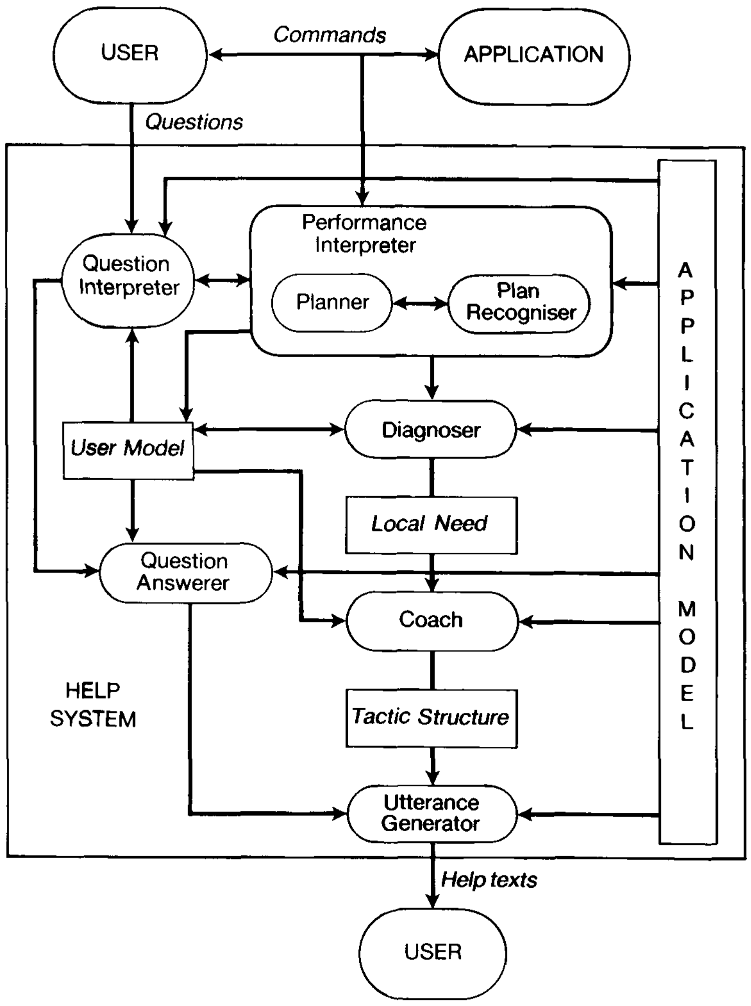


Fig. 4 The IHSS Architecture

Let us imagine that the user types a command and its arguments to the application. The *Performance Interpreter* can intercept this, parse it and determine the intended piece of application functionality to be invoked. (This is all done by employing the descriptions found in the Application Model.) If the command is syntactically correct, a *Plan Recognition* process is initiated whereby the IHSS determines which plans or methods, for the various tasks declared to it, could involve this particular command. This information is combined with information from previous cycles to narrow down the list of possible plans and hence tasks which the user could be undertaking. A similar sort of process is performed for tasks forming sub-tasks of larger tasks, and so on up a task decomposition hierarchy which will be described later. The net result of this is that the IHSS has some idea of the task the user is currently undertaking, together with the objects involved in that task.

Having determined the current task of the user, the *Planner* is invoked to determine whether there is an alternative, more efficient plan which the user could have employed. If so the IHSS has recognised that the user may have a problem or lack of knowledge for which help could be provided. In such a case the *Diagnoser* is called to determine the reason why the user didn't use the more efficient plan, and pinpoint what is called the *Local Need* of the user. For example the user may not know about a particular facility which is required for the more efficient plan. Having determined an *Occasion* for *Expanding* the user's knowledge, the *Coach* is invoked. This selects an appropriate strategy for dealing with the current Local Need, and constructs the basis of a Help Text in the form of *Tactic Structure.* A Tactic Structure is a collection of standard text templates called Tactics, selected by the strategy from a library, and then instantiated with specific concepts from the Application Model. In this example the Local need would indicate a *Lack of Knowledge* about a specific *Command,* say, and the Tactics would all correspond to statements about a command, and the fact that an inefficiency had been detected.

Finally the Tactic Structure is passed on to the Utterance (or Natural Language) Generator which fills out the text templates to form a piece of English text.

The whole process is analogous to medical diagnosis, with the Performance Interpreter first recognising the signs or symptoms of a need for help, the Diagnoser then identifying the cause, the Coach deciding upon a suitable help response, and finally the Utterance Generator actually delivering a piece of help text.

As can be seen from Figure 4 all the components discussed so far make use of information held in the Application Model and the User Model. The principles embodied in the Plan Recogniser, Diagnoser, Coach, and Utterance Generator, are all generic and driven by the contents of the Application Model, adapting to the content of the User Model.

What has been described thus far is an example of the Active side of the IHSS coming into operation. This is only one type of occasion where the Active side would intervene, but one which is new to help systems in that it is a capability for *'expanding'* the user's knowledge as the opportunity arises.

If the user asks a question directly then the Passive side is activated. The user is restricted to asking only certain types of questions which were determined from empirical studies. There are two important types of question which account for the majority of questions asked, namely Enablement (*"How do I do Task-X?"*), and Elaboration (*"What is Object-Y"* or *"Tell me about Command-Z"*). The Question Interpreter supplies the Question Answerer with both the type of question, and the topic of the question, which is a concept from the application model. A Strategy for answering is selected based on the question type and the strength of understanding of the topic concerned as determined from the User Model. An appropriate answer scheme for the question type is then used to gather together material from the Application Model to form the basis of an answer content. This is then critiqued, and/or filtered according to User Model information on any other concepts involved in the collected answer material. Finally the material is organised into a form suitable for the Utterance Generator to generate a piece of English text.

In addition various *'follow-on'* opportunities are prepared so that, for example, the user can ask a further question about a topic raised in the top level answer. In this way the user can directly follow-up on any topic he wishes.

2.5 Application modelling

2.5.1 Introduction As implied by Figures 3 and 4 the IHSS is literally driven by an Application Model of the application for which help is required. The Application Model contains all the application-specific knowledge that the IHSS requires. As such its content has evolved out of a complex and sometimes conflicting set of requirements. As one might expect this requires representations of a great deal of knowledge, ranging from the syntax of commands to the tasks and methods undertaken by the user. All this knowledge forms part of the Application Model.

2.5.2 Operational knowledge A basic spine of an Application Model is a hierarchic decomposition of user tasks, right down to the detail of what has to be typed or clicked to invoke an application facility. This task decomposition hierarchy represents the operational knowledge that the user requires to perform tasks with the application. It is composed of a number of types of concept which form the building blocks of the Application Model. Figure 5 shows this hierarchic decomposition in schematic form.

A *Task* is a job or goal which the user has to accomplish using the application. It can have one or more *TaskPlans* (as depicted by the crows-foot) each of which represent an alternative method of accomplishing the same Task. TaskPlans are defined in terms of other sub-Tasks (as depicted by the looping arrow), with a defined sequence and control structure. Each sub-Task will have its own alternative TaskPlans, each with their own sub-Tasks, and so on, forming a Task decomposition hierarchy.

At some point in the task decomposition the user will have to employ the functionality of the application. To represent this the lowest TaskPlans are decomposed into sub-tasks called *SystemProcedureTasks.* A SystemProcedureTask is the task of invoking a *SystemProcedure*, which is defined to be an atomic unit of application functionality. SystemProcedures represent the

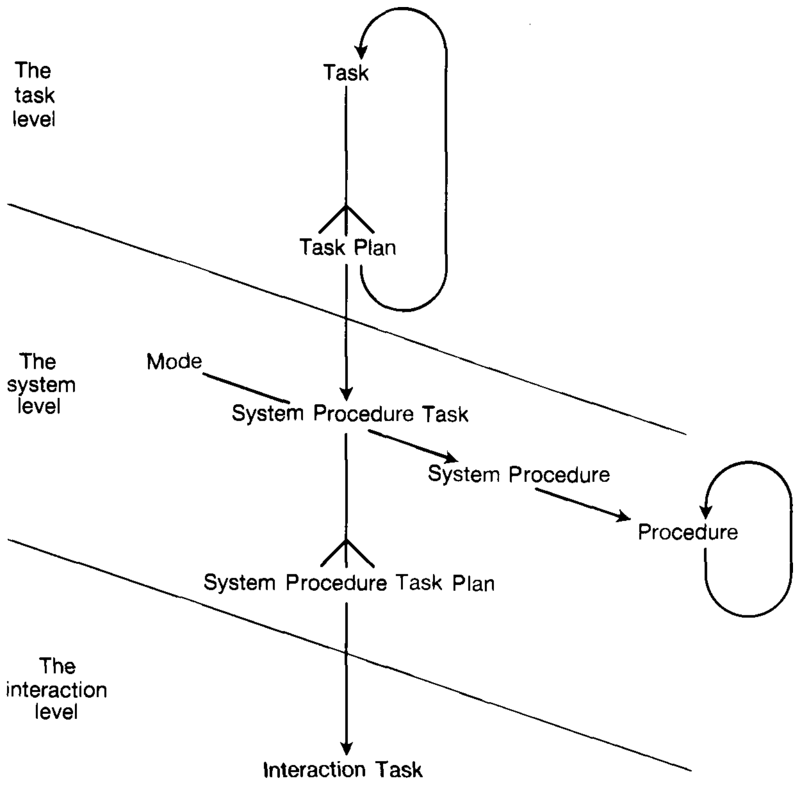


Fig. 5 The Two Decomposition Hierarchies

smallest unit of functionality that the user can directly and independently invoke. SystemProcedureTasks, like Tasks, can have one or more plans called *SystemProcedureTaskPlans* which define alternative interaction sequences for calling upon the desired application facility. Each step of a SystemProcedureTaskPlan is represented by an *InteractionTask* which defines an interaction of the user with the system. An InteractionTask describes the Event-Type (a mouse click, or typing etc.), what is clicked or typed (defined by a grammar), and where the event takes place (e.g. in a window or menu).

In summary the operational task decomposition knowledge is formed out of Tasks with TaskPlans of other (sub-)Tasks. The lowest TaskPlans are formed out of SystemProcedureTasks whose plans are in terms of InteractionTasks.

This decomposition can be used in answering enablement (how) questions to a level of detail which includes what has to be typed or clicked. It also serves for plan recognition where InteractionTask elements of SystemProcedure-TaskPlans are *ticked-off* in the parsing process in order to recognise

SystemProcedureTasks, which are, themselves *ticked-off* in TaskPlans to recognise Tasks, and so on up the hierarchy.[5]

Having *recognised* a Task, the Planner applies rules to decide whether the appropriate method (TaskPlan) has been employed in the current situation. If not then both the recognised TaskPlan and the one selected by Plan Selection Planner are used by the Diagnoser in order to analyse the cause of such sub-optimal performance.

2.5.3 Support knowledge Orthogonal to the task decomposition hierarchy is the organisational structure of the application itself. Typically an application has a number of *Modes*, which manifest themselves to the user in the form of prompts (e.g. % in Unix) to which the user is allowed to respond with a limited set of *commands.* These invoke corresponding *SystemProcedures* or application facilities. Thus in an Application Model we have *Modes* which are defined by a set of *SystemProcedureTasks* (low level tasks) which can be undertaken in those Modes. Each SystemProcedure Task has a single associated SystemProcedure describing the overall effect of a system facility.

Since the overall effect of a SystemProcedure can be quite large, it too can be decomposed into smaller units called Procedures, which can themselves be further decomposed into other Procedures. Eventually this decomposition *bottoms-out* in primitive Procedures whose effect can be concisely defined in terms of Pre- and Post-Conditions. Procedures can be shared between SystemProcedures, but *cannot* be independently invoked by the user.

This SystemProcedure/Procedure decomposition is employed in a number of IHSS modules but in particular:

- In the Coach and Question Answerer which use the successive layers of decomposition to explain the effects of an application facility.
- In the Emulator where the decomposition make possible the description of primitive procedures, and hence makes possible the maintenance of a representation of the Application State

2.5.4 The three levels Figure 5 shows the two orthogonal decomposition structures. As can be seen an Application Model is divided into three levels, the *Task Level*, the *System Level*, and the *Interaction Level.* The Task Level is intended to describe what the user would wish to do with an application, that is, the purpose for which the application is being used. Its contents are the definitions of *Tasks* and the *Objects* which form part of the real world. The System level contains the *Support Knowledge* required for a *deeper* understanding of how an application works. The Interaction Level is used to represent the MMI of the application.

[5]Note that additional analysis is required to keep track of objects being manipulated, and also to recognise possible alternative plans that the user could be following.

The intention behind the levels is to enable (reasonably) independent analysis and construction of different aspects of the Application Model. For example it is to be hoped that if an application had its command-based interface replaced by a mouse-based one, then all that would need to be re-modelled would be the interaction level. In practice changing the MMI of an application in such a way actually requires changes to the application code itself and hence re-modelling at the System Level will also be required. Nevertheless a high degree of independence has been achieved in adopting these levels.

In addition to what has been described so far the Application Model also describes all the objects involved at the different levels, and how they are passed or mapped up and down the two main hierarchies. This is described in the next section.

2.5.5 Objects and their manipulation There are four types of object:

- TaskObjects are those of which the user is expected to have a working knowledge and they relate to the real world purpose for which the application is being employed.
- SystemObjects are those which are manipulated/supported directly by the application itself; the current system state is principally defined in terms of them.
- InteractionObjects are things such as Windows and Menus, which provide the medium through which the user and application communicate.
- ObjectReferenceObjects are things such as *filenames* and *message-numbers*, which in a sense form the language in which users refer to SystemObjects.

All such objects can be organised into a classification or IsA hierarchy, plus two part-whole decomposition hierarchies based on the HasPart and Contains relations.

Having defined the objects at the three levels it is necessary to describe how they are actually used and manipulated by the Tasks, SystemProcedures, etc. in an application model. To do this all concepts shown in Figure 5 (i.e. Tasks, TaskPlans, SystemProcedureTasks, SystemProcedureTaskPlans, InteractionTasks, Modes, SystemProcedures, and Procedures) can have associated *Arguments*, *Variables* and *Results*, in much the same way that functions or procedures have in conventional programming languages. Each Argument, Variable, or Result can be further defined, and in particular has an associated *Value-Restriction* in the form of a specific Object Class. Thus for example a *Task* can be said to take an *Argument* which should be a *Document*. Arguments and Results are *mapped in* and *out* respectively by *ArgumentMappings* and *ResultMappings*. These define how objects are passed up and down the two main hierarchies of the application model.

2.5.6 The application state representation The current state of the application is represented by a set of assertions, typically of the form

(±Relation Object Object)

For example

(+Contains SOI−MailBox−1 SOI−Message−4)

means that **SOI−MailBox−1 Contains SOI−Message−4** is *true*. The same assertion with a '−' means that it is not true. The Pre- and Post-Conditions of procedures are expressed in a very similar manner, except that *'variables'* can be used in place of actual objects, thus facilitating emulation. These variables can be Arguments, Variables or Results of the Procedure, or else simply free-standing ones which exist only in the Emulation process. This process takes the Procedure's Pre-Conditions and *matches* them against the current application state both to check that the pre-condition and to instantiate the free-standing variables. These are then substituted in the post-conditions which are asserted into the current application state. In asserting these post-conditions the Emulator ensures that no contradictions occur, that is both a '+' and a '−' assertion existing at the same time.

2.6 The Help System Development System

The Help System Development System (HSDS) is a collection of tools specifically tailored to the construction of Application Models. They comprise:

- A variety of *Hierarchic Browsers* which are primarily intended for inspection and navigation purposes.
- A *Frame Editor* intended for both inspection and editing of individual concepts viewed as a Frame.
- A *Plan Editor* which is a specialised editor allowing the builder to construct plans of tasks, specifying ordering constraints and the objects being manipulated.
- A *Step Editor* which is a specialised structured editor for defining the precise control and sequencing of tasks within a plan.
- A collection of *validation utilities.*
- Generalised *Storage and Access Mechanisms* based upon a relational model which is used by both the HSDS and IHSS.
- A number of mechanisms for deriving parts of the Application Model automatically.

The tools do not include facilities to support knowledge elicitation, since these are adequately covered by existing toolkits (for example KADS [HWB87], and KANT [SB89] from ESPRIT and Alvey involving STC/ICL respectively).

2.6.1 Storage and Access Mechanisms A basic implementation decision was to use a relational model for the storage of an Application Model and an SQL-like query mechanism for access and update. All relations and concepts are defined in the Application Model Definition report [S*89] which is used to generate a Meta Application Model (*Meta-AM*) description. This *Meta-AM* defines the value restrictions on the columns or fields of relations and is used to validate a Model as it is being constructed. Clearly this will never prevent the construction of meaningless Application Models or parts thereof, but it makes a significant contribution to the overall quality/correctness of the models produced.

2.6.2 Browsers and Editors Hierarchic Browsers provide the basic navigation and inspection facilities of the HSDS tools and allow the AM Builder to explore easily the structure of a particular Application Model. Figure 6 shows a task decomposition hierarchy as might be viewed using the HSDS tools.

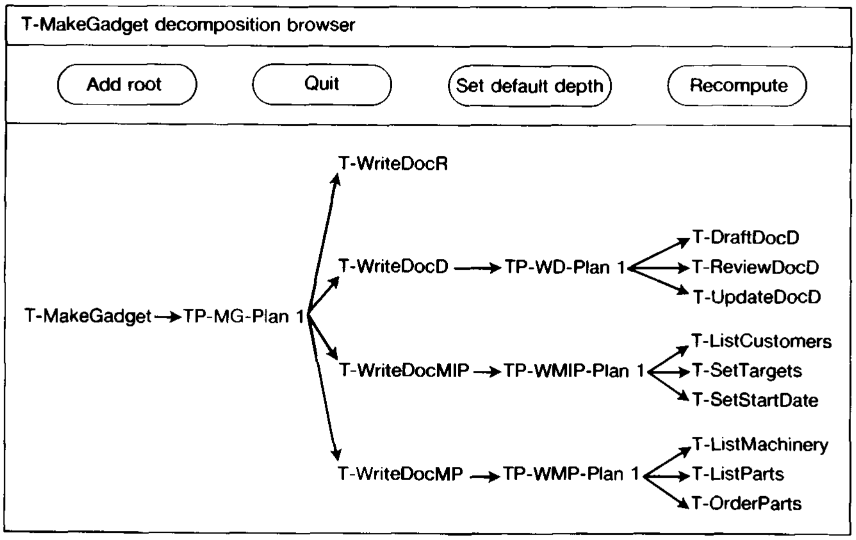


Fig. 6 The Task Decomposition Browsing Tool

Given a specific concept, of a specific type, the Frame Editor examines the Meta-AM to find all the relations in which the particular concept could be involved. It then contructs a *'frame'* of all the existing tuples which involve the named concept.

This forms a basic low level editor tool for use when other specialised editors aren't appropriate for some reason.

Figure 7 shows the Plan Editor in which the user can quickly and easily draw up a definition of a process or plan, showing the tasks, user roles, and object interdependencies.

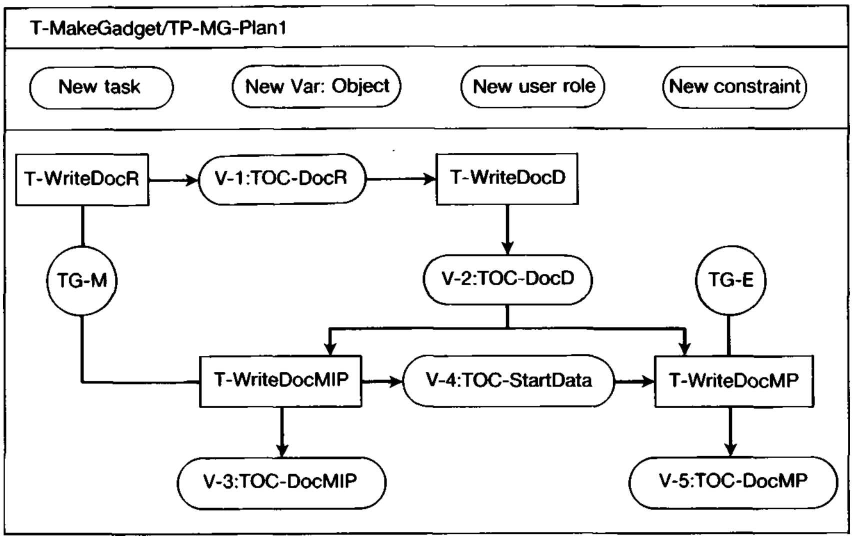


Fig. 7 Role and Object Dependencies for T-MakeGadget/TP-MG-Plan1

The Step Editor is another specialised tool for editing Plans which permits the AM Builder to easily contruct and maintain complex control and sequences structures. Figure 8 gives an illustration.

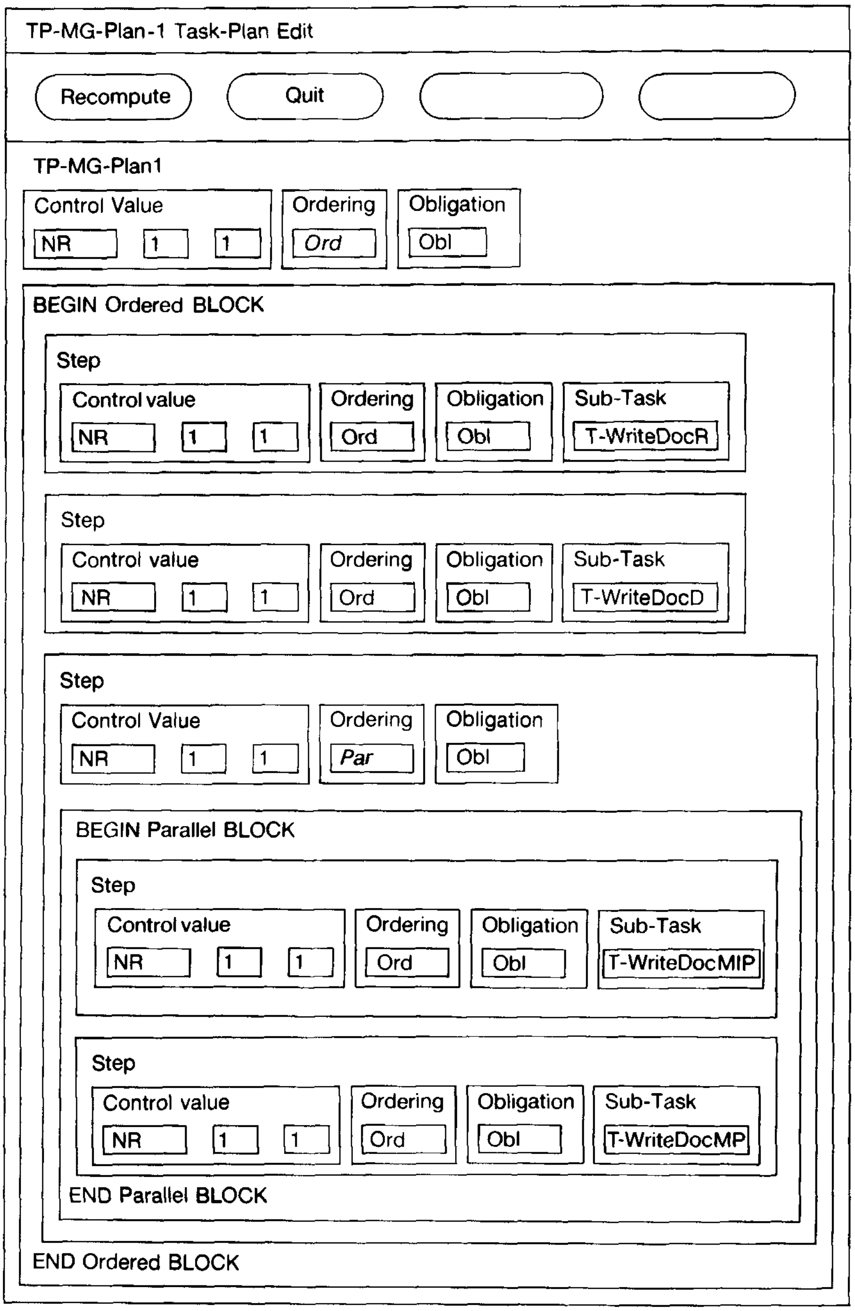
Figure 9 shows a typical TaskPlan decomposition and gives an illustration of

- how steps of a plan can be organised into *blocks*.
- how steps within a block can be specified to be ordered or unordered.
- how some steps contain control structures which control the block of steps below them.

2.7 The Help System Development Methodology

As described earlier part of the methodology for help provision is embodied in the architectural framework proposed by EUROHELP: First all application-independent knowledge of how to give help is realised in the form of the IHSS, thus removing one burden from the developer and in so doing preventing ad-hoc changes undermining the help provided. Second the IHSS is literally driven by a model of the application for which help is required. These two points combined mean that the developer *'simply'* has to concentrate on building an application model using the HSDS tools, (although an interface between the IHSS and the application is also required). This task is split into three phases[6] as listed:

[6]It should be noted that there are certain differences between dealing with extant systems and dealing with new ones. The EUROHELP project concentrated on extant system.


TP-MG-Plan-1 Task-Plan Edit

TP-MG-Plan1

Control Value: NR 1 1 | Ordering: *Ord* | Obligation: Obl

BEGIN Ordered BLOCK

Step
Control value: NR 1 1 | Ordering: Ord | Obligation: Obl | Sub-Task: T-WriteDocR

Step
Control value: NR 1 1 | Ordering: Ord | Obligation: Obl | Sub-Task: T-WriteDocD

Step
Control Value: NR 1 1 | Ordering: *Par* | Obligation: Obl

BEGIN Parallel BLOCK

Step
Control value: NR 1 1 | Ordering: Ord | Obligation: Obl | Sub-Task: T-WriteDocMIP

Step
Control value: NR 1 1 | Ordering: Ord | Obligation: Obl | Sub-Task: T-WriteDocMP

END Parallel BLOCK

END Ordered BLOCK

Fig. 8 The Block Structure of TP-MG-Plan1

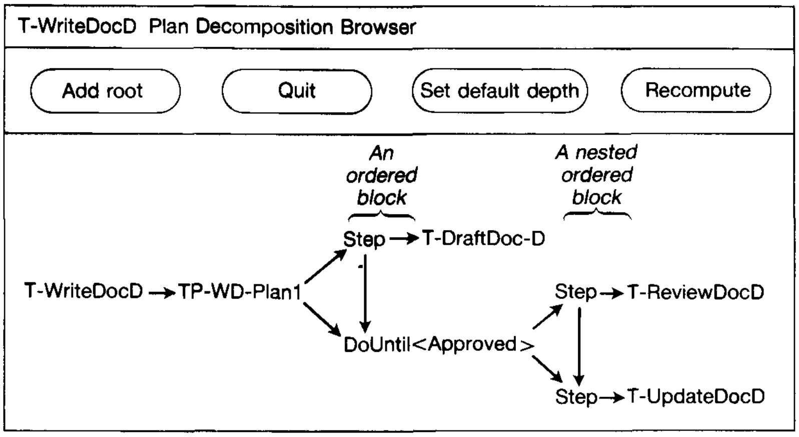


Fig. 9 The TaskPlan Decomposition Browsing Tool

1 User and Task Analysis.
2 The laying down of a *Conceptual Map* and *Model of Operation* of the application.
3 The actual construction of the model using the HSDS tools.

Phase 1 is not directly supported by the HSDS tools, and can in fact be undertaken independently of the provision of help using any currently available analysis technique [JDL84]. The output of the Marketing To Design methodology is particularly well suited to the EUROHELP approach (and vice versa) [Hut90], and includes descriptions of users, a hierarchy of their tasks, and the objects they manipulate. This information is later used to construct, and directly corresponds to the Task level of the Application Model.

Phase 2 can be viewed as a preparatory pencil and paper exercise for the later construction of the System and Interaction Levels of the application model (It is proposed however to exploit protocol analysis or hypercard tools such as KANT [SB89] from the Alvey DHSS Demonstrator, and/or the KADS tools [HWB87] from an ESPRIT project.) This phase itself is split into two stages. The first is a simple process of listing objects, operators, commands etc., and noting any groupings or other relationships between concepts. Such information can be gleaned from documentation, using the application, or talking to expert users. Its purpose is to set down a *'Conceptual Map'* and not only establish the required vocabulary and terminology, but also the overall scope of the modelling task. The second stage, of defining a *'Model of Operation'* is rather subjective in nature, and is intended to establish (at least) a *'mental picture'* of how the application actually works, which can be conveyed, via help, to the end user. Typically this Model of Operation is an *'As Is'* model, but need not necessarily be so. For example, in Unix Mail, how to describe the flags on messages and also the use of memory/filestore are two major decisions to be made in defining its Model of Operation.

The final phase, Phase 3, concerns the actual construction of the Model using the HSDS tool. There are no hard and fast rules about this, and each developer will arrive at his own way of working. However experience has shown that building small areas of the model, in depth, is preferable to adopting a more breadth-first approach. The reason for this is that it is nearly impossible to remember all the details typically left *'hanging'* in a breadth-first approach. For this reason it is thought best, for the System and Interaction levels, to

1 choose a particular facility,
2 define each command variant for invoking that facility,
3 define each object mentioned together with the means of referencing it,
4 define the *effect decomposition of the facility,*
5 place the facility in the Mode hierarchy,
6 and last, but certainly not least, define all the object mappings.

As building progresses more and more overlap is encountered, perhaps requiring some iteration, leading to less and less effort for each facility defined. Gradually the model fills out, rather like doing a jig-saw puzzle, until the defined facilities represent a framework for all future construction.

2.8 Types of Application

The project experience has shown that there are a number of factors, which give rise to a 'need for help', that need to be represented in an Application Model:

1 The range of facilities or commands an application makes available.
2 The range of Tasks for which the application can be employed.
3 The complexity of the *Model of Operation* needed to be understood by the user in order to make effective use of the application.
4 The depth of *domain knowledge* or *data content* which needs to be exploited in using the application.

The EUROHELP project addressed only factors 1 to 3 above. Factor 4, as shown by our experience with an SQL application, would require significant extensions of the Application Model definition to encompass far more *domain* knowledge. For example although there is some requirement for help on the SQL facilities themselves which can represented, the main requirement is for support in formulating data-base queries which specifically extract only the desired information. This requires *data dictionary* type knowledge, which in principle could be added to an Application Model, but would introduce an additional level of dependency, in that the data dictionary knowledge is concerned with the application of the application, for example the data in the database. Thus EUROHELP did not really address applications for which 4 is significant.

3 The Benefits from Intelligent Help

This chapter sets down the ways in which the IT Community can gain from the EUROHELP deliverables, by examining the potential beneficiaries of this technology:

The Application End Users – The people who employ the application in their everyday working life and who require help and training in doing so.

The Application Purchaser – The organisation which employs the end users and which should benefit from their effective use of the application.

The Application Developer – The organisation which undertakes the design, implementation and maintenance of the application.

The Application Vendor – The organisation which pays for the application development and which should profit from its sale.

These beneficiaries will be dealt with in turn in subsequent sections.

3.1 The Benefit to the End Users

Here we are not so much concerned with how help is generated, but more with its quality and effectiveness. In this respect we can assess the benefits from EUROHELP under a number of headings as follows:

The Nature of Help – The design of the EUROHELP IHSS is based on significant empirical studies and attempts to address the *specific needs* of an end user. Thus the Active and Passive help capabilities are quite complementary, and together give a comprehensive range of help, designed genuinely to assist the user in most situations. No conventional help system provides such a comprehensive range of help facilities.

The Context Sensitivity of Help – The EUROHELP IHSS maintains an emulated application state and also attempts to infer the intentions of the user. This information is not only used by the Active side in deciding when to intervene, but it is also used in answering a number of question types.

The User Sensitivity of Help – The EUROHELP IHSS maintains a user model in the form of an overlay of the Application Model. This is used in both the Active and Passive sides; on the Active side to govern the timing and topic of an intervention, and on the Passive side to select material to include in an explanation. Both Active and Passive sides also make use of the user model in the selection of the top level strategies for response formulation. The overall benefit is that the individual can proceed at his own speed, and can develop his knowledge of different areas of the application at different rates whilst at the same time receiving the appropriate level of help in each area. The net result, especially when combined with context sensitive help, is a much more flexible help system, genuinely reacting to real user needs.

The Accessibility of Help – Here there are a number of contributory factors:

- Questions can be asked at any point in time about *commands*, *objects*, *tasks* and also about *what happened.*
- Follow-on questions can be asked about any topic mentioned in any generated help text.
- Tasks and Objects can be *browsed* as a means of discovering what to ask and also what the application provides. In addition questions on commands can be asked via menus organised according to modes.
- The Help System is *Active* in the sense that it will intervene on appropriate occasions to offer help.
- Access to help does not interfere with the user's ongoing dialogue with the application itself.
- Appropriate information is dynamically gathered together according to researched schemes of help generation and addressing particular needs.

The Effectiveness of Help – Clearly a full-scale evaluation study and some live use is required to *prove* the effectiveness, or otherwise, of the EUROHELP IHSS design. Nevertheless it is felt that in general the help generated should be more effective than in conventional help systems for the reasons already discussed. One additional advantage of the EUROHELP approach is that if some deficiency in the generation of help is discovered, then the necessary improvement changes are confined to the IHSS and at worst some aspect of the Application Model. (This is in contrast to conventional help technology, where the changes would probably be spread over many individual help texts.) In this way there is every chance that the IHSS can at least evolve toward an effective help system, if it is not already.

3.2 The Benefit to the Purchaser

There are a number of factors concerning user effectiveness and user training which can be beneficial to the Purchaser:

- First help is designed to enable the user to *carry on* with his current task.
- Second it exploits various opportunities in both active and passive help to *expand* the user's knowledge of the application – a form of *on-the-job* training.
- The use of the IHSS is expected to reduce the amount of conventional training and printed documentation necessary.

3.3 The Benefits to the Developer

Conformance to Requirements and Acceptability will be strongly enhanced by any technology which genuinely supports the user in task performance.

Productivity and Maintainability – There is considerable overlap between application modelling and activities undertaken as part of the application development process. Clearly exploitation of such an overlap will undoubtedly lead to gains in these areas, especially when combined with the other benefits described in this section.

3.4 The Benefits to the Vendor

Benefits to the vendor are the following:

Development Costs – As previous sections indicate there is scope for both productivity and quality improvements.

Product Differentiation – Effective intelligent help would differentiate the application from ones without it.

Internationalisation – The use of an Application Model is a significant step toward the possibility of constructing multi-lingual applications.[7]

Customisation – This is similar in many respects to Internationalisation in that the use of an Application Model is a significant step forward.

4 Conclusions – A Critical Appraisal of EUROHELP Techniques

Section 2 described the EUROHELP project's results as they stood at its end in 1990. As mentioned earlier it had substantially met its objectives but fell short with respect to complete *productisation*. Section 3 then sets down the kind of benefits that can be expected from a help system along the line proposed by the EUROHELP project. This section critically appraises the results of EUROHELP, clearly indicating the authors' views on what should be *productised*, and what, at this moment, is only of theoretical interest.

4.1 Add-On Help versus Integrated Help

The EUROHELP project began with the mission of developing a means of providing help for extant systems. This led to a deep understanding of the nature of help, and a clear definition of the interface required between an application and a generic help facility. This interface revolves around two basic requirements over and above the basic requirement for an Application Model:

1 the need for an intelligent help system to have access to application state information.
2 the need to intercept easily **and** to interpret all input and output between application and user.

Requirement 1 tends to be very difficult to achieve for extant systems, and is dependent on how they are engineered. This essentially means that the help system has to resort to *'gleaning'* whatever information it can from the application's output and user's input, and emulating the effects of each command.

Requirement 2 is more easily met, but to avoid duplication of processing (eg parsing of input by both the application and the IHSS) it is better that the

[7]It is not a simple way forward, since language differences might necessitate substantial changes to the Application Model formalism, but nevertheless we are moving in the right direction.

application is modified to intercept input/output at an appropriate point and pass it on to the IHSS.

All this implies that the Add-On approach can only satisfactorily work with *'hands-on'* access to the application code so as to supply the required interface, and even then there can be problems.

The preferred approach is what is termed the *integrated* approach, in which help and the application are designed and implemented together. The benefit of this approach is that the necessary interfaces and access to information can be guaranteed in the design process. However for the integrated approach to be possible the application development methodology should subsume the methodology for providing help.

Clearly there are not simply two extreme approaches, the Add-On and the Integrated. The Integrated approach is really a matter of degree. There is therefore a whole spectrum of what are called *Hooked* approaches, in which the application is in some way modified to provide *hooks* for the provision of help.

4.2 Application State Information

As discussed in section 4.1 it is difficult to extract state information when help is added-on to an existing application. The reasons for this are that

- state information tends to be embedded throughout the application code.
- commands which provide state information often cannot be used transparently by the help system since they also change the state of the system.

Equally emulating the state is prone to difficulties:

- the emulated state gets out of synchronisation with the real state.
- it is typically only a partial model, with all the consequential problems.
- describing the effects of application facilities is difficult.

For these reasons it is best that the application is designed to supply an interface through which the IHSS can gain access to state information.

4.3 Performance Interpretation

Performance Interpretation can be broken down into three main components:

Parsing – the IHSS more or less has to replicate what the application itself has to do in the way of *parsing* user input. In some circumstances the parsing process in not simply a straight forward syntactic analysis problem. Application state information, such as the existence of a file, or whether a name refers to a directory or a file, often can influence the process. Isolated parsing by the IHSS is therefore somewhat disadvantaged, and clearly a single parser servicing both application code and the IHSS would be a preferred solution.

Plan Recognition – In EUROHELP the only plans which can be recognised are ones which are hand-built into the Application Model. Thus as well as representing all *'good'* methods for accomplishing a task, *'inefficient'* and even *'mis-conceived'* plans have to be constructed. This places an extra burden on the Model builder and in turn practical limitations on what can be achieved with such an approach. Additionally plan Recognition is a computationally expensive process which has to be undertaken even if the user is performing satisfactorily. It therefore constitutes a significant overhead on general task performance.

Planning – is used in testing whether the user is performing effectively, and also in answering specific Enablement questions. In both cases a heavy reliance is placed on state information, and effect descriptions.

The net conclusion for Performance Interpretation is that there is a heavy reliance on model building and application state representation, both of which are somewhat problematical. In addition an order of magnitude improvement in performance is required before it becomes a practical proposition. Therefore some rescoping is required which focuses sharply on specific help requirements.

4.4 Diagnosis

Diagnosis ranges from being straightforward at one extreme, to being almost intractable at the other. It is always dependent upon the accuracy of the Performance Interpreter and User Modeller, which is in turn dependent upon Diagnoser, thus leading to somewhat of a *'vicious circle'*. Nevertheless it has been demonstrated that certain useful diagnoses can be made, but the practicability hinges around the speed at which these modules can be made to perform.

4.5 Application Modelling

There appears to be an ever increasing set of requirements on Application Modelling, ranging from supplying inefficient plans to describing how to parse input. Whilst it is important to factor carefully non-generic components from the IHSS into the Application Model, there is a practical limit in terms of just how much the model builder can be expected to do.

Another major issue concerns the *'match'* between model and reality. First there is the question of how detailed or abstract the model should be. Then of how accurate a model can be made.

These issues should, at least, be alleviated by adopting an Integrated approach to help provision and application development.

4.6 Coaching and Question Answering

The combination of Active and Passive help seems a good one, which provides a comprehensive range of help facilities. Even if performance interpretation cannot adequately be rescoped so as to continue to support much of the Active side, then shifting similar functionality into the passive side certainly seems a plausible alternative and this can be done without significantly reducing the range of range of help provided.

4.7 User Modelling

The EUROHELP project employed relatively simple User Modelling techniques, involving straight-forward attachment of information to concepts in the application model as a form of overlay. Provided the performance interpreter and other modules can supply the necessary information this approach seems to be perfectly practicable. The overall conclusion is that something more akin to *'usage modelling'* as opposed to *user modelling* would suffice for help provision.

4.8 Summary

- The comprehensive range of help proposed and demonstrated by EUROHELP constitutes a definitive standard against which all (future) help systems should be judged. At the very least application developers should take note of the flaws in conventional help technology.
- The integration of help provision techniques into standard application development processes is essential if this new standard for help is to be put into practice.
- The EUROHELP project has successfully demonstrated the feasibility of intelligent help. It is has also defined the nature of the required interfaces for a general run-time component to be incorporated into future applications.

These last two issues will be addressed in subsequent papers.

4.9 Acknowledgements

The authors wish to acknowledge that the work reported in this paper covers results from the entire research project. Thanks go therefore to all participants of the EUROHELP project. Thanks also go to Charlie Portman for reading and constructively commenting on early versions of this paper.

References

[Bre90] BREUKER, J. editor. The EUROHELP Final Report. Publisher to be announced, 1990. Copies obtainable from the authors.

[Chi83] CHIN, D.N. Knowledge structures in UC, the Unix consultant. In Proceedings of the 21st Annual Meeting of the ACL, pages 159–163, June, 1983.

[DH82] DOUGLAS, R.J. and HEGNER, S.J. An expert consultant for the Unix operating system: bridging the gap between the user and command language semantics. In Proceedings of the Fourth CSCSI/SCEIO Conference, pages 119–127, Saskatoon, 1982.

[Equ80] Digital Equipment. VAX VMS Command Language User's Guide. Digital Equipment Corporation, 1980.

[Fin83] FININ, T. Providing help and advice in task oriented systems. In Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI '83), pages 176–178, 1983.

[FLS85] FISCHER, G., LEMKE, A. and SCHWAB, T. Knowledge-based help systems. In Proceedings of Human Factors in Computing Systems, pages 161–168, April, 1985.

[Heg87] HEGNER, S.J. Knowledge representation in Yucca-II: exploiting the formal properties of command language behaviour. In Proceedings of the Workshop on Knowledge Representation in the Unix Help Domain, Berkeley, December 1987.

[HKN*88] HECKING, M., KEMKE, C., NESSEN, E., DENGLER, D., GUTMANN, M. and HECTOR, G. The Sinix Consultant – A Progress Report. Technical Report, Dept. of Computer Science, University of Saarbrucken, October 1988.

[Hou84] HOUGHTON, R.C. Online help systems: a conspectus. Communications of the ACM, 27(10): 126–133, February 1984.

[Hut90] HUTT, A.T.F. and FLOWER, F. Marketing to Design. ICL Technical Journal, Vol. 7, No. 2, pp. 253–269.

[HWB87] HAYWARD, S.A., WIELINGA, B.J. and BREUKER, J.A., Structured analysis of knowledge. International Journal of Man-Machine Studies, 26, 1987.

[JDL84] JOHNSON, P., DIAPER, D. and LONG, J. Task, Skills and Knowledge: task analysis for knowledge based descriptions. In B. Shackel, editor, Interact84, North Holland, 1984.

[Kem87] KEMKE, C. What do you know about mail? Representation of commands in the Sinix consultant. In Proceedings of the Workshop of Knowledge representation in the Unix Help Domain, Berkeley, December 1987.

[Lim82] Microsoft Limited. Microsoft Multiplan. Microsoft Limited, Windsor, 1982.

[Mor83] MORAN, T.P. Getting into a system: external-internal task mapping analysis. In A. Janda, editor, Proceedings of CHI'83 Human Factors in Computing Systems, pages 45–49, ACM, New York, 1983.

[NWW87] NORVIG, P., WAHLSTER, W. and WILENSKY, R. Proceedings of the Workshop on Knowledge Representation in the Unix help domain. Berkeley, December 1987.

[OMa85] O'Malley, C.E. Helping users help themselves. In D.A. Norman and S.W. Draper, editors, User Centered System Design, pages 377–398, Lawrence Earlbaum Associates, 1985.

[RMH88] ROBERTSON, J., MAURO, T. and HELBIG, K. A Guide to Andrew, X Version11, Release 3. Technical Report, Information Technology Center, Carnegie Mellon University, Pittsburgh, 1988.

[S*89] SMITH, M.J. et al. The Application Model Definition. Research Report ICL-ULE/EUROHELP/041, ICL, Future Systems, 1989.

[SB89] STORRS, G.E. and BURTON, C.P. Kant – a knowledge analysis tool. ICL Technical Journal, May 1989.

[SF82] SHRAGER, J. and FININ, T. An expert system that volunteers advice. In Proceedings of the Second National Conference on Artificial Intelligence (AAAI '82), pages 339–340, 1982.

[Sha88] SHAFER, D. HyperTalk Programming. Hayden Books, Indianapolis, 1988.

[WAC84] WILENSKY, R., ARENS, Y. and CHIN, D. Talking to Unix in English: an overview of UC. Communications of the ACM, 27(6): 574–593, June, 1984.

[WCL*88] WILENSKY, R., CHIN, D.N., LURIA, M., MARTIN, J., MAYFIELD, J. and WU, D. The Berkeley Unix consultant project. Computational Linguistics, 14(4): 35–84, December 1988.

[Wil82] WILENSKY, R. Talking to Unix in English: an overview of UC. In Proceedings of the Second National Conference on Artificial Intelligence (AAAI '82), pages 103–106, 1982.

How to Use Colour in Displays[1]– II. Coding, Cognition & Comprehension

Darren Van Laar and Richard Flavell
The Management School, Imperial College, London

Abstract

Colour is often used in visual displays to improve human-computer interactions and to achieve a more natural and representative display than monochrome coding alone can accomplish. In order to use colour on displays effectively it is important to understand how colour is perceived, how colour coding can be used to support tasks, how many colours to use on a display, and when colour will be useful.

This is the second of two papers discussing the effective use of colours in CRT displays and covers task factors in colour displays, explains what is meant by colour coding, when to use the different forms of colour coding, and the effect of using colour on performance and preferences. Also covered are the aesthetic and environmental factors relevant to employing colour on displays.

This paper concludes with a summary of the physical and cognitive factors that have been highlighted by the two papers, together with some guidance as to how this advice may be implemented. A glossary of frequently used terms is also included.

1 Introduction

The first paper in this series (Van Laar and Flavell, 1990) set the scene by discussing and describing the physics, physiology and perception of colour vision. This paper will examine why colour should be used in displays, and how information should be colour coded. Although these papers have been presented in what may seem a natural order, from low level to high level factors, perhaps this paper should have been first in the series, as it is only when the programmer or designer has decided to use colour after a consideration of the task that decisions about which colours to employ

[1] This work has been supported by the University Research Council of ICL. We are grateful to A. J. Russell and members of the Strategic Systems Technology, Bracknell, for their continued assistance.

should be made. In reality only displays which have been designed with the user, the task and the environment in which they are to work firmly and uppermost in mind will be successful. The aim of this present paper is to give the reader an appreciation of the importance of task factors in using colour with CRT displays, and their relation to the user and the environment in which the task is to take place.

When colour displays were first developed it is fair to say that colour was used in a trial and error basis, with little regard for information enhancement or expression. One excellent example of this genre is the British Post Office's PRESTEL system. The PRESTEL logo itself was one of the worst examples, each letter being coded in one of the colours available (7 foreground or text colours plus one background = 8 colours). Other factors were the low resolution of the displays and the lack of experience of the page suppliers in designing pages for the system. Often the pages were individually designed by each contributor, whose aim seemed to be to produce the brightest and most garish page on the system (and therefore the most eye catching), but neglected that the pages might not be suitable for reading. Fortunately, from this low point the effective use of colour has since been considered more seriously by display designers.

Colour coding is used in visual displays in three basic ways (Teichner, 1979). First, it may be employed to provide a more pleasant display than a "bleak" monochrome presentation. However, there is conflicting evidence as to whether colour displays per se produce better performance, but they do seem to be consistently preferred over monochrome displays (see section 7.1). Second, colour may be used to reduce clutter or visual noise in the display by organising the display into perceptual units. Third, colour may be used to 'code' information, in the sense that the colour may be used to convey meaning in the same way that alphanumerics, graphs or meaningful symbols might be used. This paper will concentrate on the use of colour for coding, although the other two uses are also briefly discussed.

Note that many of the concepts and definitions used in this paper are explained in Van Laar and Flavell (1990), and it is assumed that the reader will be familiar with this. A large reference section is included, in an attempt to make the reader aware of the range and depth of research in this area, and to provide access to more detailed work. Many of the unfamiliar terms used here are included in the glossary, appendix A.

2 Task analysis and the use of colour

Before discussing the use of colours in CRT displays, it is important to place the display within a task context. The display does not exist for its own sake, but to transfer information to a human observer. When designing displays a number of factors have to be borne in mind, in particular the task to be performed, the nature of the observer, and the environment within which the activity is to take place. There is a substantial difference between a secretary peforming a word processing task in an office and a navigator of a battle tank trying to identify objects on a radar screen.

There have been relatively few studies applying task analysis to CRT displays. Early studies emphasised the motor tasks, eg. how rapidly were keys being struck, rather than cognitive tasks such as search or discrimination. An unpublished recent report (Fenn, 1986) surveyed various task analysis procedures to assess their relevance to CRT displays and concluded that none at that point in time were completely suitable, this conclusion has also been reached more recently by Sutcliffe (1988).

Possibly the most relevant methodology is that of Easterby (1986) who discusses the types of tasks, their implications and the likely psychological processes. The distinction between task and process is that the former is operationally defined, ie what somebody does such as searching or labelling, whilst the latter is a theoretical model of what is thought to be taking place within the person performing the task. The processes are important because they constitute the building blocks that describe how a task is performed and thus how information may be best presented. Examples of processes are:

- Detection
- Discrimination
- Identification
- Recognition
- Categorisation

Easterby discusses how the display and user attributes may be interfaced with the processes, the former under the display designer's control but the latter obviously outside it. Examples of the former are readability and conspicuity, and of the latter training and motivation. No mention is made of the environmental attributes that would also impinge on performance.

But even Easterby's work is still at a very descriptive level. Very few studies have investigated the relationship between task and colour coding in displays (Long, Eldridge and Carver 1983; Campion, 1989). Whilst these made use of task analysis, the formats of the displays were already determined and the different colouring alternatives extremely limited. The studies were designed to generate applicable colour knowledge that was objective and generalisable, but in their own admission fell short. The fundamental difficulty appears to be that the experimental tasks employed have to be seen to capture the important features of the real tasks; in the context of this survey, the features that are sensitive to colour coding.

While there is little doubt that task analysis is of major importance to the design and colour encoding of displays, it is a developing technique that is really still at a descriptive and exploratory phase and cannot be widely applied to cognitive processes without great investments in time and money. This type of in-depth task analysis takes a large degree of experience to implement properly, and is generally only available within human factors departments of large companies or colleges. The display designer who has a smaller project is therefore recommended always to think first about supporting the goals and tasks of the user when applying colours to the display.

3 Colour coding of information

Colour is useful as a non-redundant coding dimension where it can be employed to reduce clutter on a display by attributing meaning to colour codes rather than introducing another symbol. For example, air traffic controllers use colour to indicate direction of flight, alphanumeric symbols show height and position and speed is spatially encoded. The danger is that the user may forget just what meaning the colour represents, or confuse one colour with another especially if no intuitively representative colour is available. Colour coding in such situations (especially in high risk, life sensitive tasks) should therefore only be used with people trained specifically for such tasks.

It is sometimes valuable to use colour within a multidimensional colour display, where information in a particular colour in one part of the display can mean something different from other colours on the display, or the same colour elsewhere on the screen. Egeth and Panchella (1967) have shown how, in general, the greater the number of coding dimensions in a display, the less accurately each dimension, and the meaning associated with it, can be identified. Common coding dimensions other than colour in a display are shape, position (spatial coding), size, blinking, alphanumeric (textual) and icons.

Colour should therefore be used as a means of making the logical structure of the information in a display more visible. Other factors such as making the display more attractive or of conforming to previous conventions are secondary.

Colour is used widely to separate information into different classes but without implying any relationships between the classes. For example, the lines in a map of the London Underground are generally coloured differently to aid discrimination, and there is no additional information to be deduced from the fact that the Piccadilly and Victoria lines are different shades of blue. A colour scale, eg. increasing saturations of red, may be used to imply numerical information. A classic example is in cartography, where the deepness of the sea is indicated by the shade of blue. The deeper the sea, the darker the blue, and so one may be able to deduce at a glance the relative depths of parts of the sea.

The use of colour in the London Underground map is a type of 'qualitative' coding and is used to represent nominal data. This form of coding is very widely used in information displays, and can be employed in three ways:

highlighting, discrimination or relating. In the cartography example, 'quantitative' coding is used to code differences in degree between depths. In quantitative coding colours are coded along a colour dimension in a meaningful way, as in some infrared film preparations where colder temperatures are represented by colours nearer the blue end of the spectrum, warmer temperatures by colours towards the red. Quantitative coding is used for ordinal and even interval data to indicate the amount of, or level of, difference between codes. There are a number of problems associated with quantitative coding, not least the choice of scale or colour dimension to use when displaying data.

3.1 Qualitative use of colours

The qualitative use of colour may be classified into supporting one of three types of task: highlighting, discrimination and relating.

Highlighting: In order to emphasise or draw attention to important fields of information, a word, a title, an error message, a graph axis, etc may be coloured differently from other information.

Relating: Colour may be used in visual displays to indicate a relation between fields of information. A sense of relation may be produced by using similar colours eg pink and red (varying degrees of saturation) or red and orange (neighbours within the spectrum), or by relating a given interpretation of the colour to that used in the display (danger and cautionary messages in red and amber).

Discrimination; This is the use of a cue or coding dimension to distinguish between two or more different sets of data. An example of this would be the use of a black and red coloured text to indicate credits and debits on a spreadsheet program.

3.2 Quantitative use of colour

The use of colour to imply quantitative information was, until recently, relatively uncommon on displays. This was principally due to the limitations of the displays to generate a sufficient range of colours. When necessary, the limitation was in part overcome by the use of 'pseudocolouring', namely the assignment of colours to numbers (or ranges of numbers). For example, a positron emission tomography (PET) scan may be used to measure the glucose energy requirements of differing parts of the brain (and hence differing levels of activity). It might be displayed using the following code:

Units	Colour	
1–100	blue	
101–200	yellow	
201–300	red	etc.

There are two major problems with this method of virtually arbitrary assignment of colours to codes. First, there is no obvious intuitive relationship between the numerical and the colour scales, and hence the relationship has to be learnt by rote. Second, two quantities may be relatively close numerically, but because they lie either side of a colour boundary would be translated into different colours, ie. artificial discontinuities are created. Perceptual problems are also not usually taken into account, with yellow for example appearing much brighter than an equally saturated blue of the same luminance and will therefore stand out as more important or more different than the numerical value it represents.

Despite difficulties in display interpretation poor and arbitrary pseudocoding still persists in many application areas such as medical imagery and satellite data interpretation. A more effective way to pseudocolour a display may be to use a continuous path through a colour space such as RGB or L*u*v* in such a way that the numerical change in the data is represented by a correlated physical change in the colour stimulus or code. For example, using the brain scan data above, if it is assumed that the maximum number of units is 300, then dividing the red voltage range into 300 steps and then showing units of glucose uptake with the appropriate voltage on the red gun (with the green and blue guns off) would result in higher readings being represented by increases in red. Unfortunately, whilst such a transformation would step evenly through the RGB space, the user would not perceive colours on the scale as changing evenly as brightness and saturation are confused together (see Van Laar and Flavell, 1990). Thus stepping instead through a perceptual colour space such as L*u*v* or HLS would be far more appropriate and intuitive for the users (Robertson, 1988; Levkowitz and Herman, 1986).

Even when using a perceptual space some problems still exist. Probably the most obvious is that there is a wide range of intuitively 'sensible' paths through such colour spaces, and it is often difficult to determine which is best. For example, which is better (based on HLS space):

a. Fix lightness at 50% (L = 50), saturation at 100% (S = 100) and then step through the hues (H)?
(Sometimes called the gamut boundary or rainbow scale).
b. Fix H (say to red), and S = 100, and then evenly step through L?
(Lightness scale).
c. Fix H, and L = 50, and then evenly step through S?
(Saturation scale).

The first scale will pass through all of the hues in the spectrum, and the non-spectral purples, the second will run from black through fully saturated red to white, whilst the third from mid-grey to fully saturated red. These are three obvious paths and Flavell and Heath (1989) found that the rainbow scale was the most easily interpreted but it was difficult to extract relative information from it rapidly. The lightness scale was nearly as good whilst also possessing an intuitive preceptual logic. The third was significantly worse, which was surprising given that a saturation scale is frequently suggested in the literature. However, it was found that a grey scale (ie. any H, S = 0 and step through L) was nearly as accurately interpreted as the rainbow or lightness scales which shows that well chosen monochrome displays can be as good as colour in tasks of this type.

A second issue is whether the scale should be as continuous as technically feasible, or should there be a relatively small number of discrete graduations or steps in the colour space. It can be computationally expensive to step evenly through a perceptual space (because of the computing overhead involved in transforming the many small steps into RGB gun values). However, it has been found that untrained people intuitively use a small number of graduations, probably no more than 10 (Jones, 1962; Oborne, 1982). Above this number, people tend to 'chunk' graduations, ie. create bigger graduations mentally, which can decrease task performance and increase error rates.

The National Physical Laboratory (Clarke and Leonard, 1989) have produced recommendations on choosing colours for pseudocolour scales based on their work on thermal imaging. Robertson (1988) concludes that colour coding using two dimensions ie. cross-sections through the space maintaining constant hue or constant lightness, is best. However, Ware and Beatty (1988) in an experiment testing the explicit use of three dimensions of colour found that problems impelementing HSV space and an opponent process space meant that the colours used in their experiment had to be based on the non-perceptually uniform RGB (monitor) space.

4 The effect of colour on task performance

The impact of colour on the performance of tasks is not well understood. Most studies have used experimental tasks whose extension to real tasks is debatable. A recent study (Chechile, Eggleston, Fleischman and Sasseville, 1989) which looked at the cognitive complexity of displays found that 99% of the variation in performance between complex military displays was predictable from variations in the task alone. They concluded that perceptual factors not related to the task, such as an increase in contrast of items in the display or the use of different colours for objects, would have very little effect on performance. This next section will look at the use of colour coding in various tasks that have been examined experimentally.

4.1 Colour vs monochrome coding.

Much of the research on colour coding has used as a control, or been compared with, monochrome displays and codes. This section tries to summarise the most important findings of this work according to whether colour was found to help, to hinder or whether it had other effects on the task performed.

4.1.1 Performance advantages Christ (1975), in a survey of work on colour coding up to that time, concluded that for many tasks concerned with identifying objects on displays, colour was a better form of coding than shape or size, but was not as good as alphanumeric coding, and when the task was to locate (search) or count codes in a display colour coding was superior to all of the other codes in the experiments reviewed. However, it was observed that colour sometimes produced longer search and identification times than shapes, alphanumerics or size of object codes when colour coding was present as an irrelevant display stimulus.

According to Fisher and Tan (1989) colour is a consistently more effective means of highlighting information on a display than boxing, blinking or reverse video. Although the exact effect will be determined by the probability that the message/object highlighted is the one that is important (highlighting validity) and the probability that subjects will attend first to the highlighted parts of the display.

Kopala (1979) showed that redundant colour coding significantly reduced both response times and error rates in pilot flight performance. Carter (1982) obtained fast, predictable search times from colour coded displays when the number of items in the target class of colours in a search task was small, although the effectiveness of the colour coding was found to diminish as the target class size increased. In a study by Sidorsky (1980), colour was seen to substantially reduce processing times of complex information in battle displays.

Van Laar (1989) found that redundantly colour coding programming displays improved performance in a programming maintenance task over monochrome presentations of the same information.

4.1.2 No clear advantage or disadvantage With very high levels of practice the usefulness of colour coding displays diminishes. Christ (1983) reports a series of nine experiments which investigated the use of the coding dimensions of letters, digits, familiar geometric shapes and coloured dots in visual displays. The experiments used search, choice reaction, location and identification-memory tasks. Christ wrote that "These experiments provide no basis for concluding that any particular code has a general advantage or disadvantage over any other.", and "The relative effectiveness of different visual codes is a function of practice, other display conditions, the tasks, and the dependent measure used to make the comparison." (Christ, 1983 p. 71).

Foster and Bruce (1982) conducted an experiment which looked at the effect of three colour formats and a monochrome control on tabular presented information. Although colour coding was found to be advantageous in some conditions, colouring groups of data in a non-task-related way can have detrimental effects on interpreting such displays.

According to one of the latest theories of visual search (Duncan and Humphreys, 1989) search performance decreases as targets and non-targets become more similar, and, less obviously, as the non-targets become more similar. This implies that colour coding non-targets (such as the background or unimportant screen details) in a similar fashion to any colour item (target) will have a detrimental effect on search performance.

Knapp et al (1982) found that performance on colour and achromatic codes was equal in a study of the ways in which information should be highlighted in complex displays.

4.1.3 Performance disadvantages Teichner (1979) in an early summary paper thought that the use of colour as a coding dimension was overrated given the present evidence, and that colour coding offered no real advantage over achromatic or monochrome coding dimensions.

Beck, Sutter and Ivry (1987) found that relative to luminance contrast, hue is a poor method of segmenting images into their constituent parts. Furthermore when both luminance contrast and hue contrast segmentation information is given, the hue information can be ignored.

4.2 Legibility

A number of studies, especially in the early 1980's, sought to discover which combination of colours on a display was the most legible, both when on their own (monochrome displays) and when in combination with other colours. Unfortunately most of these studies employed displays which had a much lower pixel resolution and a smaller colour gamut than is usually employed today. These studies also usually regarded names of the colours used as sufficient descriptions of the physical stimulus, which for the reasons described in Van Laar and Flavell (1990) is wholly inadequate. Therefore the following descriptions of studies and others found in the literature should be treated with caution.

Small (1982) found that in a legibility task employing an eight colour display where users had to spot spelling errors in familiar texts, legibility was highly positively correlated with the log luminance contrast between the colour combinations. However, the poor quality of the display used meant that these effects were possibly effected by misconvergence effects, especially with secondary colours (ie cyan, magenta and yellow). Sawyer and Talley (1987) provide results summarizing legibility of characters based on non-colour characteristics.

4.2.1 Calculating the perceptual difference between colours Bruce and Foster (1982) required subjects to read aloud three randomly sorted alphabets and three sets of digits (0 to 9) presented in 56 foreground-background combinations of eight colours. From their results they recommended that the relative luminance index between the background colours should be higher than 0·3 to maintain maximum character visibility. The relative luminance index, also known as the CIE luminance contrast formula, is given by:

$$\text{Contrast} = (L_o - L_b)/L_b$$

where L_o = foreground/object luminance, and L_b = background luminance. Note that a positively contrasting display gives a positive number, and a negatively contrasting display a negative number (see section 4.3).

As mentioned in Van Laar and Flavell (1990), luminance is not a useful dimension to represent information in displays for a number of reasons, and a better method is to use lightness (L*) which gives a truer estimate of the relative perceived luminous intensity between colours. Thus the lightness contrast coefficient (LCC) is calculated as above but with luminance (L) replaced with the perceptual measure L*. To illustrate lightness differences between colours, Table 1 below shows the actual luminance and lightnesses of patches of colours as measured with a tele-spectroradiometer for the sixteen colours available on one of the most popular types of colour display currently in use. This table shows how the perceptually based L* scale illustrates the perceived colour difference much better than the luminance reading. For example, bright yellow might well be nearly four and a half times better in physical luminance, but actual appearance is much nearer to the less than twice as bright figure predicted by the L* data.

Table 1 Actual luminance and relative lightnesses of colours on an IBM CGA monitor, with a reference white of $x = 0{\cdot}2823$, $y = 0{\cdot}2952$, $Y = 77{\cdot}3$.

IBM number	Colour	Luminance Cd/m^2	Lightness L*
0	Black	0·67	7·83
8	Grey	4·95	30·4
11	Blue	5·45	31·92
4	Red	8·79	40·2
5	Magenta	11·52	45·5
9	Bright blue	11·9	46·17
6	Brown	15·95	52·55
12	Bright red	18·02	55·39
2	Green	26·33	65·01
3	Cyan	26·91	65·6
13	Bright magenta	27·3	65·99
7	White	30·0	68·65
10	Bright green	49·8	84·19
11	Bright cyan	61·63	91·56
14	Bright yellow	70·8	96·64
15	Bright white	77·3	100·0

Although the figures above are useful for calculating the LCC of text-background combinations on an IBM CGA display, it cannot be stressed enough that these figures will only be useful without major modification by other CGA owners of a monitor of approximately the same age (four years) and usage, with similar brightness and contrast settings. Nevertheless, lightness contrast and therefore discriminability may be roughly estimated using the above table for any pair of colours.

Table 2 shows the LCC for a number of the foreground and background colours given in Table 1. Differences in lightness are not the whole story, and differences in hue and saturation make a large contribution to the perceived difference of a colour, although exactly to what extent they affect legibility is unknown and the subject of current research. Table 2 also shows the perceptual colour difference (ΔE) and it is clear that this can differ markedly from the LCC. When lightness alone is considered combination number 2 is nearly ten times more discriminable than no. 3, but when hue, lightness and saturation are considered together then 3 would appear to be nearly twice as contrasting as 2.

Table 2 The LCC and ΔE colour difference for three colour combinations on an IBM CGA display.

No.	Foreground	Background	LCC	ΔE
1	Yellow	Blue	2·02	228·48
2	Cyan	Blue	1·05	73·9
3	Red	Magenta	0·11	130·7

The observation, that the CIE ΔE formula for measuring approximate perceptual colour difference has little bearing on how different colours look in displays, is very important. In an unpublished study conducted in our labs the ΔE colour difference formula, based on actual measurements from a monitor, only began giving reasonable correlations of subjective estimates of colour difference when the luminance part of the equation (ΔL^*) was multiplied by 50. Other support for this observation comes from Matthews (1987) and Lippert (1986).

4.3 Positive vs negative polarity

The argument as to whether positive (black text on light backgrounds) or negative (light text on dark backgrounds) should be used when displaying information on VDUs is often studied, and is often the starting point when designing colour displays. Positive displays are supposed to be better than negative displays because of their greater overall luminance which increases acuity and decreases the effect of illuminant glare. However, with bright, wide-angled, displays with low refresh rates, flicker can be a problem. For further information on specific situations see Pawlak (1986).

5 The number of usable colours on a CRT

It is impossible without specific information to answer the frequently posed question "how many colours can be used on a graphics display?". The question can only be answered when both the task and physical environment contexts are also considered. However, some general guidance can be given on how to estimate the number in a particular situation. This guidance depends on two factors: the perceptual and display factors that determine the 'discrimination' of colours in the specific display, and the number of colours people can categorise or 'identify'. The next two sections deal with these factors in turn.

5.1 Perceptual and display factors

Common advanced colour systems have the ability to produce 256 steps in voltage on each of the three colour guns. As the guns are (generally) independent, this implies that 256^3 or roughly 16·7 million different colours may be generated on the display.

Distinguishing between colours is a relative judgement, and is assessed by the 'just noticeable difference' or 'jnd' metric. This refers to the smallest perceivable difference between two colours and one which will be discriminated at least half of the time by an observer. Ideally this difference, if measured as a distance in a perceptually uniform space, should be constant throughout the space, and equal in all dimensions. In practice there is a ratio of about 4 to 1 between the size of the maximum and minimum jnd in the theoretically perceptually uniform $L^*u^*v^*$ space. It has been estimated that there are probably about 1·2 million just noticeably different colours on a typical high resolution advanced colour display (Tajima, 1983).

In practice, in a normal working environment, greater discrimination would be required. Heath (1986) estimates, by separating colours by about 10 jnd's and thereby ensuring easier discrimination, that an upper limit of 1000 distinct colours might be useable for colour coding purposes. If the ambient lighting is very high then this number will reduce rapidly (because as L^* approaches 100% the size of the space reduces rapidly) down to double or even single figures.

Van Laar and Flavell (1990) in the first paper in this series summarise the perceptual and display factors that will affect the choice and number of colours to use.

5.2 Cognitive factors

When the task is to tell whether two colours are different people can perform remarkable feats of fine discrimination. However, when the task is to categorize or remember colours, relatively few colours can be managed without extended training or practice.

5.2.1 Colour code set size Performance has often been shown to be affected by the number of colours used in a display. Cahill and Carter (1976) showed how the time taken to search for a coloured item in a display decreased as the number of colours increased to seven, but increased significantly as the number of colours exceeded ten. Luria, Neri and Jacobsen (1986) however, found that the time required to say whether a colour was included in a set of colours increased linearly up to set size of 5 or 6, after which it increased much more rapidly. In the same study error rates increased sharply with set sizes of 8 to the maximum set size considered of 10. Luria et al (1986) noticed that reaction times to individual colours remained the same throughout the conditions, with dark blue, red, purple and white being the quickest, and yellow orange and cyan being the slowest matched. These differences in performance were not due to brightness differences, but seemed to be based on hue (eg. orange most often confused with red and yellow) and saturation differences (with reaction time decreasing as saturation increased).

However, the task may necessitate not merely distinguishing a colour difference, but in recognising a specific colour. Some relevant experiments were performed in the 1950's but obviously not using CRT's. Heath and Flavell in their experiments report that most subjects appeared to be able to use only about 6–7 colours or steps, although this number increased if a reference scale was presented (Heath and Flavell, 1985). This agrees with the earlier work, such as Jones (1962) who suggests using only 8 colours for reliability, and Oborne (1982) who recommended the use of a maximum of 10 colours.

5.2.2 Colour naming Related to colour recognition is colour naming, in other words how somebody categorizes colours using familiar terms. Boynton, in a series of experiments (Boynton et al, 1989; Uchikawa and Boynton, 1987), found that both Japanese and American subjects used only 11 basic terms plus a range of adjectives to describe most colours. These eleven basic terms are:

> white, black, red, green, blue, yellow, brown, purple, pink, orange and gray

Given this limited vocabulary, it is important for displayed colours to be perceived as members of these categories if they are to be readily identified and remembered (see also the next section).

5.2.3 Memory for colours Although memory can be thought of as being a collection of many separate stores, each with characteristic properties, the basic distinction between iconic (sensory), short-term (working) memory and long-term (semantic memory) is still valid.

Iconic memory is characterised by being able to store a literal copy of a visual scene for a very short time (about one second). This type of memory is very susceptible to interference but can give the other stores enough time to focus on a particular feature which can then be remembered. Interestingly with very short-duration stimuli people seem to be able to report either the location or the colour of an object but not both at the same time (Clark, 1969).

The advantage of short term memory for colours over other code types is not that the colours themselves are better remembered, but that they allow the display to be divided into meaningful areas more easily. In this way colour coding the number of aircraft by altitude would not help recall the position or the identity of an aircraft but would help in reporting the number of aircraft within each colour (altitude) segment (Clark, 1969; Davidoff, 1987).

Short-term memory can usually only hold 7 ± 2 items at one time. If a task requires that more than seven different codes are to be remembered without the support of a displayed reference or key, then people will have difficulties and begin to make mistakes, especially when the task itself is demanding (Heath, 1986). The exact hue of an object seems to be very well remembered up to thirty seconds after presentation of a stimulus, and after this period colours are recalled less accurately. However, after this point performance remains at this level indefinitely.

One of the most interesting effects with long term memory for colours is that unlike shapes or drawings which are usually remembered in a pictorial form, long-term memory for colours is usually encoded in a verbal form. Boynton et al (1989) provide a possible explanation for this by suggesting that after a short period of time colours are no longer directly comparable on their physical characteristics, but become categorised and labelled into one of the eleven basic colour terms (see section 5.2.2). Long-term colour memory appears to be independent of shape and texture memory, and seems to depend on the type of stimulus to be remembered (verbal or pictorial, natural or made-up object, etc). Memory for colours is reviewed in more detail by Davidoff (1987).

5.2.4 How many colours to use? In conclusion, the maximum number of colours that should be used in a display seems to depend on the task. If the task requires a non-redundant colour code to be employed then the number of colours used should not be more than the eleven familiar colour categories, plus one or two other easily identifiable 'qualified' colour codes such as 'Royal blue' or 'Sky blue'. The identification and use of colours is always helped by a reference scale, and where possible such a scale should be displayed.

If the display is cluttered, ie it has many codes and types of information, then fewer colours should be used. Long (1984) recommends up to a maximum of eleven colours with a low density (ie. uncluttered) display, a maximum of seven for medium density (around 45 items), and as few as five colours on a very dense display.

If the task is to identify relative changes along a scale, and the actual colours will not be used without a reference scale, then 1000 different colours may be used on a display. If the task involves the representation of a natural scene, for example when looking at photo-fit pictures or digitised images of houses (an estage agent's database?), then as many colours can be used as are in the actual images.

6 Environmental factors in using colour in displays

6.1 Health and colour displays

Matthews (1987) reported poorer reading performance, higher ratings of discomfort and higher incidence of reported symptoms of discomfort from people using combinations of colours from the extremes of the spectrum (red, blue), rather than the middle of the spectrum or white. However, this experiment did not adequately control for brightness contrast, and the results might well be due to the lack of brightness contrast rather than that the colours were from opposite ends of the spectrum.

Lovasik, Matthews and Kergoat (1989), in a better controlled experiment, found that when subjects performed three 4-hour search tasks using a colour monitor with different colour combinations no differences in visual discomfort occurred, although colour combinations did lead to differences in search time.

6.2 Lighting

Ostberg (1975) referrring to negative information presentations and low resolution displays recommends that environmental illuminations should not be over 150 lux in order to minimize glare. However, in order to see papers clearly it is important to have brighter illumination. Consequently the lighting levels recommended by the lighting industry for use with CRTs is a compromise of 300–500 lux (see also section 4.3). Bright illumination also serves to desaturate colours on a display, therefore possibly changing the intended meaning of colour codes or scales, and imparting a ‘washed out’ appearance to the colours.

7 Subjective factors and aesthetics in colour displays

One of the most difficult topics to deal with when assigning colours to displays is that of colour aesthetics. Despite sensible and scientific application of colour guidelines the displays may still be poor or even unusable without a proper understanding of what makes a display attractive, acceptable and pleasant to use as well as being useful.

7.1 Colour preferences

It has been observed that when a choice is given computer users prefer colour displays over monochrome (Tullis, 1981). This is, at least in part, due to the aesthetic appeal of colour which can be used to improve user satisfaction in what might otherwise be a boring or repetitive task (Hopkin, 1977). People may feel more secure when using colour CRTs, and rate their task as easier. This can occur even when performance itself is not improved (Jeffrey and Beck, 1972) or is even degraded (Hopkin, 1983), compared with monochrome displays.

In one study, subjects trained on flight deck displays incorrectly believed that colour improved their ability to detect details, even though it had no such effect (Witt and Strongman, 1983). If users believe that colour displays will ease their workload, they may (given that people tend to work a set amount within a certain time) relax and complete less work, as they think they are being more effective in their job.

Colour has been found to improve acceptance of a new system within an organisation (Knapp, Franklin and Gellman, 1982), and of the colours used in a study of colour codes, computer operators preferred blue-green colours, even though performance with these same colours was not generally improved (Jones, 1962).

Vickerstaff and Woolvin (1950) found that the legibility of colour text/background combinations was improved by attractive combinations. Tinker (1969) also found that preferred colour combinations were correlated with higher reading speeds. However, in studies of colour preference it must always be remembered that often people realise how well they perform in a task, and when asked afterwards which colour they preferred may well pick the colour they felt they had performed best with.

7.2 Human response to colour

Many people believe that colours can have an effect on how people act and feel (Birren, 1983, Bellizzi, Crowley and Hasty, 1983). Display designers may be tempted to design colour displays in the same way in which walls are painted different colours, but often this advice is based on subjective feelings, and not on research data. For example the artist De Grandis (1986) gives advice on what colours to paint walls of different types of hospital, and recommends for within operating theatres that "... the most widely used colour (should be) green – both for the walls and for the surgeons gown – because the surgeon's eye, strained by the red of the patients blood, finds this restful." (p 102)! In a survey of the literature, Davidoff (1987) makes the point that no experimentally sound study exists to support the generally held opinion that hue can actually make a person feel warmer or colder, or happier or sadder, but that many studies exist which refute these claims. However, luminance has been found to exert some influence on people's feelings about coloured objects, and there is some evidence to show that saturation may also have an effect (Kunishima and Yanase 1985). According to De Grandis (1986) to produce a perceived harmonic relationship between highly saturated colours so that the luminance of one does not dominate, one should vary the surface area of colours in inverse proportion to their brightness at the same luminance.

People often have a 'favourite' colour, and this may be important for determining the preference of isolated colours in displays, but in one study where subjects were asked to rate how much they liked a car, hue was found to have no effect on preference (Saito, 1983).

One thing that may be concluded from people's colour preferences and their likely effects on performance is that whatever occurs will be complex and related to isolated colour preference, the object or display context, culture, fashion and to how people feel at the time.

7.3 The semantics of colours

Meanings and stereotypes ascribed to colours should not be overlooked when designing a display. For example, a misleading impression may be gained if a section of text is misconstrued as denoting 'danger' instead of its real meaning such as 'on'. Table 3 shows the most popular colours associated with a selection of concepts given to American college students. Ehlers (1983) reported that users of a videotext system showed frustration when common colour conventions of this sort were not observed.

Table 3 Colour stereotypes of American college students (Bergum and Bergum, 1981).

Concept	Colour	Response (%)
stop	red	100
go	green	99
cold	blue	96
hot	red	94
danger	red	89
caution	yellow	81
safe	green	61
on	red	51
off	blue	31

8 Guidelines for the effective use of colour and colour coding in displays

Tufte (1989) recommends using colour conservatively, and following examples of good colour use such as the "cartographic excellence" to be found in many maps. The use of natural colours or colours found in nature is also recommended for screens, with local emphasis using more saturated or brighter colours acceptable if used in moderation. Colour can also provide aesthetic, pleasant displays when used with thought and care. This section summarises the work described in this series of two papers, and our experiences in designing colour displays.

The following guidelines are in the order in which they appear in the papers, the paper (I or II) and section (eg 2.4.1) are also given for reference.

8.1 Physiology, physics and perception guidelines

1 Care must be take when assigning colours to a display that the environment in which they were designed will be similar to that in which it will appear, as illumination and monitor differences can make the colours look completely different. (I, 2.2.1; I, 4.1)

2 8% of males and 1% of females in the west have some type of colour defective vision. The major effects of this can be overcome by maintaining large luminance differences between colours which are likely to be confused (especially reds, greens and browns). (I, 2.3)

3 Colours produced on a CRT are additive. This implies that a secondary colour such as cyan is likely to be brighter than its constituent primaries of blue and green. Brightness is an extremely important perceptual dimension when colours are coded in hue as well as lightness. (I, 3.3)

4 A perceptual space should be used to choose colours where possible, the next best is a device-dependent pseudo-perceptual space, eg HLS, then RGB gun values, then arbitrary assignment of colours to values (I, 3.4.1; I, 4.3)

5 The use of highly saturated colours from opposite sides of the colour circle can result in a number of visual and perceptual illusions, such as the fluttering hearts and chromatic aberration effects. In addition, such colours will induce colour shifts, eg. a foreground colour will be perceived as having moved towards the complementary colour of the background. Highly saturated colours should therefore be avoided unless they are inherent in the coding dimension. (I, 5.0)

6 As we age our colour discrimination abilities become less good, especially affected are yellow discriminations. Care should also be taken when designing displays for users who are likely to suffer from colour deficiencies due to illness or disability. (I, 5.2.5)

8.2 Coding, cognition and comprehension guidelines

1 In tasks requiring rapid search of a display, colour is generally superior to many other forms of coding, but with identification tasks colour has a smaller advantage. (II, 4.1.1)

2 The more similar the colour of targets, and of targets to non-targets in a display, the longer search tasks will take. (II, 4.1.2)

3 In legibility tasks the relative lightness of foreground and background colours is more important than differences in hue and saturation. (II, 4.2).

4 The greater the density of the display, in terms of number of codes and items of information, the fewer the number of colours that can be used without performance decrements. (II, 5.2.1)

5 If colours are to be absolutely identified no more than 10 graduations should be used in a quantitative coding scale, and only 6–7 if a reference scale is absent. If only relative judgements are to made, then many more graduations may be used. (II, 5.2.1)

6 People use only a limited number of basic colour names to describe and to remember colours in general. Only colours corresponding to the basic colour categories should be used, especially with untrained subjects, in order to optimise performance. (II, 5.2.2)

7 The range (gamut) of colours that may be perceived is determined by both the display system and the ambient lighting. High ambient lighting will desaturate colours and make them less discriminable, thus reducing the gamut of useable colours. (II, 6.2)

8 Beware that colours may imply a particular interpretation due to the use of colour coding in other contexts. The use of a colour in a display should avoid being contrary to its broader interpretation. This interpretation is not always portable across different cultures. (II, 7.3)

8.3 Discussion of the use of guidelines

These guidelines are deliberately not specific, unlike those given in many earlier studies which tended to make broad assumptions such as that: only highly saturated colours would be available, the background would be black or ambient lighting absent. There is no realistic reason for these assumptions to be true. The above guidelines are therefore based upon more general findings about the underlying processes of physics, physiology, perception, cognition and comprehension, which in turn should mean that they are more widely applicable.

Notwithstanding the theoretical and experimental basis underlying these guidelines, they should not be followed and interpreted blindly. There will be times when they will interact, and then a judgement on priority will have to be made (based, it is suggested, on an understanding of the task). The final arbiter of the success or failure of a display is the end user, and guidelines cannot supplant him or her. They may however help tremendously in reducing gross errors in designing an initial version of a display.

Guidelines are for guidance, and should not be thought of as legally binding, or written in stone!

9 Conclusions

This paper, together with its predecessor published in the previous edition of this journal, has been written as an introduction to the effective use of colour in CRT displays. Only three years ago the typical business display possessed only 8 colours and low spatial resolution. Since that time very high resolution systems have become commonplace, many with the ability to select from a palette of thousands of colours. This has meant that a designer has to be provided with a different style of information and assistance from the early, quite rigid and dogmatic guidelines. Because there is considerably more control over the display, it is important for the designer to understand the underlying processes so that they may be applied in the literally infinite number of possible displays that might be designed. It is hoped that these papers will provide part of that assistance.

It may be of interest to speculate on the next generation of assistance. We ourselves believe that it is likely to be on-line, in the form of multi-layer, context sensitive help and assistance, with the ability for the designer either to apply colour to a display and get comments on it, or alternatively to get the system to make an initial suggestion. Such a system would be sophisticated, but not intelligent. It would also possess the ability to guide the user through colours and colour spaces, to show in real time coloured displays and permit the user to alter on-display colouring and see the effect of such changes.

Until this time the best advice we can offer is to design your displays with the user, the task and the environment in which they will interact firmly in mind. This is the key to the successful design of any interface.

Appendix A: Glossary

Absolute judgement	A task where the user has to decide upon the value represented by a colour without the context of other colours.
Categorisation	The assignment of a code to a class or category based on the attributes of the code.
Conspicuity	The property of a stimulus which makes it stand out from other stimuli.
Detection	A task which requires the user to say when and/or whether a stimulus has appeared on a display.
Discrimination	The telling apart of two codes by their physical differences along some dimensions.
Hue scale	A quantitative scale of colours, which is associated with a change in hue, usually from a reference hue, around a hue circle and back again, for a colour of a given lightness and saturation.
Identification	Linking a code with meaning, whether by using reference scales or by remembering the values.
Lightness scale	A quantitative scale of colours, which is associated with a change in lightness, usually from black to the white for a colour of a given hue and saturation.
Non-target	Objects on a display which are not targets, which are not associated with a task.
Qualitative coding	Coding information by differences in type, eg coding a column of credits black, and a column of debits red.
Quantitative coding	Coding information by differences in degree, eg a brain scan, showing temperature by colour.
Recognition	Identification of a stimulus code after being presented with it, rather than before it is presented.
Redundant coding	The use of colour to code information that can be gleaned from another code set. Eg the London underground map is spatially as well as colour coded.
Relative judgement	A task where the user interpolates between two colours of known physical and numerical value.
Saturation scale	A quantitative scale of colours, which is associated with a change in saturation, usually from grey to the most vivid example for a colour of a given hue and lightness.
Search task	The scanning of a display until the object sought is found.
Target	Usually an object on a display, which must be discriminated or identified by the user.
Task	Something that has to be done or an act to be accomplished.

References

BECK, J., SUTTER, A. and IVRY, R.: 'Spatial frequency channels and perceptual grouping in texture segregation'. Computer Vision, Graphics and Image Processing, 37, 299–325, 1987.

BELLIZZI, J.A., CROWLEY, A.E. AND HASTY, R.W.: 'The effect of colour in store design'. Journal of Retailing, 59, 21–45, 1983.

BERGUM, B.O. and BERGUM, J.E.: 'Population stereotypes: an attempt to measure and define.' In Proc. of the Hum. Fac. Soc., 25th Ann. meeting. 622–65. Human Factors Society, 1981.

BIRREN, F.: 'Colour and the human response'. Color Res. and Appl., 8, 75–80, 1983.

BOYNTON, R.M., FARGO, L., OLSON, C.X. and SMALLMAN, H.S.: 'Category effects in color memory'. Color res. and Appl., 14(5), 299–34, 1989.

BRUCE, M. and FOSTER, J.J.: 'The Visibility of coloured characters on coloured backgrounds in viewdata displays. Visible Language. XVI (4), 382–90, 1982.

CAHILL, M.C. and CARTER, R.C.: 'Colour code size of search in displays of different density.', Human Factors, 18 (3), 273–80. 1976.

CAMPION, J.T.: 'Interfacing the laboratory with the real world: a cognitive approach to colour assignment in visual displays.', 35–66, In Long, J. and Whitefield, A. Eds 'Cognitive Ergonomics and Human Computer Interaction'. Cambridge University Press, 1989.

CARTER, R.C.: 'Search time with a colour display: analysis of distribution functions.' Human Factors 24(2), 203–12. 1982.

CHECHILE, R.A., EGGLESTON, R.G., FLESICHMAN, R.N. and SASSEVILLE, A.M.: 'Modelling the cognitive complexity of displays'. Human Factors, 31(1), 31–43, 1989.

CHRIST, R.E.: 'Review and analysis of colour coding research for visual displays.' Human Factors, 17(6), 542–70, 1975.

CHRIST, R.E.: 'The effects of extended practice on the evaluation of visual display codes.' Human Factors 25(1), 71–84. 1983.

CLARKE, F.J.J. and LEONARD, J.K.: 'Proposal for a standardised continuous pseudo-colour spectrum with optimal visual contrast and resolution'. IEE Conference Proceedings publ. No. 307, 687–91, 1989.

CLARK, S.E.: 'Retrieval of colour information from perceptual memory'. J. of Expt Psychol., 82, 263–6, 1969.

DAVIDOFF, J.: 'The role of colour in visual displays'. Int. Review of Ergonomics, 1, 21–42, Ed Oborne, J., Taylor and Francis. 1987.

DE GRANDIS, L.: 'Theory and use of colour'. Abrams, NY, 1986.

DUNCAN, J. and HUMPHREYS, G.W.: 'Visual search and stimulus similarity'. Psychological review, 96, 3433–58, 1989.

EASTERBY R.: 'Tasks, processes and display design', Information Design, Ed. Easterby and Zwaga, Wiley, 1986.

EGETH, H. and PANCHELLA, R.: 'Multidimensional stimulus identification.' Perception and Psychophysics. 5 (6). 31–43, 1967.

EHLERS, H.J.: 'How colour can help visualise information.' In 6th Int. Online Info. Meeting. Learned information. Oxford, 1983.

FENN, P.S.: 'Literature survey of task analysis with special reference to CRT displays', Management School, Imperial College, Dec 1986.

FISHER, D.L. and TAN, K.C.: 'Visual displays: the highlighting paradox'. Human Factors, 31(1), 17–30, 1989.

FLAVELL, R. and HEATH, A.: 'Further investigations into the use of colour coding scales', Dec. 1989, Working Paper, Imperial College Management School. 1989.

FOSTER, J.J. and BRUCE, M.: 'Looking for entries in viewdata tables, a comparison of four colour formats.' J. App. Psy. 67 (5) 611–15. 1982.

HEATH, A.: 'Colour coding scales and computer graphics', PhD thesis, U of London, 1986.

HEATH, A. and FLAVELL, R.: 'Colour coding scales and computer graphics', Graphics Interface '85, 321–8, 1986.

HOPKIN, D.: 'Colour Displays in air traffic control.' Institution of Elec. Engin. Conf. 1977.

HOPKIN, D.: 'Use and Abuse of Colour.' Proceedings of Int. Computer graphics Conf. 101–10. 1983.

JEFFREY, T.E. and BECK, F.J.: 'Intelligence information from total optical colour imagery.' US Army BSR lab. National Research. 1972.

JONES, M.R.: 'Colour coding.': Human Factors. 4, 355–65. 1962.
KNAPP, B.G., FRANKLIN, L.M. and GELLMAN, L.H.: 'Information highlighting on complex displays.' In Badre, A. and Scheiderman, B. (Eds.) Directions in HCI. Ablex. N.J. 195–217. 1982.
KOPALA, C.J.: 'The use of colour coded symbols in a highly dense situation display.' In Compass for technology, proceedings the Hum. Fact. Soc. 397–401. 1979.
KUNISHIMA, M. and YANASE T.: 'Visual effects of wall colours in living rooms'. Ergonomics, 28, 869–82. 1985.
LIPPERT, T.M.: 'Colour difference prediction of legibility for raster CRT images', SID Digest, Vol 17, 86–9. 1986.
LONG, J., ELDRIDGE, M.A. and CARVER, M.K.: 'The use of colour on the AMTE/AIO Demonstrator', Ergonomics Unit Report, University College, London, 1983.
LONG, T.: 'Human Factors Principles for the design of computer colour graphics displays.', British Telecom Tech. J., 2 (3), 1984.
LOVASIK, J.V., MATTHEWS, M.L. and KERGOAT, H.: 'Neural, optical and search performance in prolonged viewing of chromatic displays'. Human Factors, 31(3), 237–89, 1989.
LURIA, S.M., NERI, D.F. and JACOBSEN, A.R.: 'The effects of set size on colour matching using CRT displays'. Human Factors, 28 (1), 49–61, 1986.
MATTHEWS, M.L.: 'The influence of colour on CRT reading performance under operational conditons'. Applied Ergonomics, 18.4, 323–8, 1987.
OBORNE, D.J.: 'Ergonomics at work', Wiley, 1982.
OSTBERG, O.: 'Crt's pose health problems for operators'. Int. J. Occ health and safety, Nov-dec, 24–6, 1975.
PAWLAK, U.: 'Ergonomic aspects of image polarity'. Behaviour and Information Technology, 5(4), 335–48, 1986.
ROBERTSON, P.K.: 'Visualising Colour Gamuts: a user interface for the effective use of perceptual color spaces in data displays', IEEE Comp Graph Appl, Sept, 88, 50–64, 1988.
SAITO, T.: 'Latent spaces of color preference with and without context: using the shape of an automobile as the context. Color Res. and Appl., 8, 101–13, 1983.
SAWYER, D. and TALLEY, W.T.: 'Display legibility guidelines: a design aid'. Information display, 12/87, 13–16, 1987.
SIDORSKY, R.C.: 'Colour coding in tactical displays: help or hinderance?', (US army research institute tech. paper), Alexandria, VA. October, 1980.
SMALL, P.L.: 'Factors influencing the legibility of text/background colour combinations on the IBM 3279 colour display station'. IBM Human Factors Report HF066, Hursley, 1982.
SUTCLIFFE, A.: 'Human-computer interface design'. Macmillan, London, 1988.
TAJIMA, J.: 'Optimal color display using uniform color scale', NEC R&D, Vol 70, 58–63, 1983.
TEICHNER, W.H.: 'Colour and visual information coding.' Proc. of the Soc. for information display, 20(1). 3–10, 1979.
TINKER, M.A.: 'Legibility of print'. Iowa University Press, Iowa, 1969.
TUFTE, E.: 'Visual design of the user interface', IBM Corp. Armonk NY, 1989.
TULLIS, T.S.: 'An evaluation of alphanumeric, graphic and colour information displays.' Human Factors, 23(5). 541–50. 1981.
UCHIKAWA, K. and BOYNTON, R.M.: 'Categorical color perception of Japanese observers: comparison with that of Americans', Vision Res, 27(10), 1825–33, 1987.
VAN LAAR, D.L.: 'Evaluating a colour coding programming support tool.' In Sutcliffe, A. and Macaulay, L. (Eds) 'People and Computers V' CUP, Cambridge, 217–30. 1989.
VAN LAAR, D.L. and FLAVELL, R.: 'How to use colour in displays I. physiology, physics and perception,' ICL Tech. J. 7 (1) 154–179 May 1990.
VICKERSTAFF, T. and WOOLVIN, C.S.: 'Investigation of the legibility and aesthetic value of printing inks on paper'. Colour, ICI. 1950.
WARE, C. and BEATTY, J.C.: 'Using color dimensions to display data dimensions', Human Factors, 30(2), 1988.
WITT, N.W. and STRONGMAN, E.: 'Applications and expertise of colour CRT flight deck displays.' Displays, April. 77–82. 1983.

Eye Movements, for A Bidirectional Human Interface

Richard Epworth
Technology Strategy Manager, STC Technology Ltd.,
London Road, Harlow, Essex CM17 9NA, UK

Abstract

When we look at a screen, we are unaware of precisely where our eyes are looking, yet our eye movements contain valuable information. An optoelectronic system which monitors our eye movements, may be used to communicate with a machine. Items on a screen may be selected simply by looking at them, and actions initiated by deliberate eye movements. Such a system can also be aware of our interests and difficulties, and may respond to our needs, even those of which we are unaware. This paper describes the techniques of eye movement measurement, possible applications of this information, and several practical demonstrations of an eye-controlled interface.

1 Introduction

Human computer communication is far less intimate than a meeting between two human beings, where eye movements, facial expressions and body language communicate a considerable amount of the information. As a greater percentage of the population becomes involved in the use of computers, it is natural to expect the manner of controlling computers to move away from the programming model and closer to the perceptual process we use to accomplish our goals in the physical world [Krueger et al, 1985]. A more intimate relationship with computers should bring a greater awareness of our real needs. The use of eye movement information is a step towards a closer relationship.

This paper will explain the meaning of eye movements and describe techniques for their measurement. The range of applications of eye movement controlled technology will be presented, and finally details will be given of several practical demonstrations.

2 The Human Eye, its capabilities and limitations

2.1 The Physiology of the Eye

The human visual system is very complex. For the purposes of this paper it is necessary to describe a little of the physiology of the eye. Those who wish greater detail, are recommended to read "Eye and Brain" by Richard Gregory [1966]. The eyeball is approximately spherical, and is rotated within its bearing socket by means of a number of muscles. An optical system images the external view onto the retina which is a dense matrix of light sensitive cells, inside the back of the eyeball (Fig. 1). The dominant imaging component is the bulge in the front surface of the eye, the cornea. The lens within the eye provides some additional focusing which is controllable, enabling the eye to focus on objects at different distances.

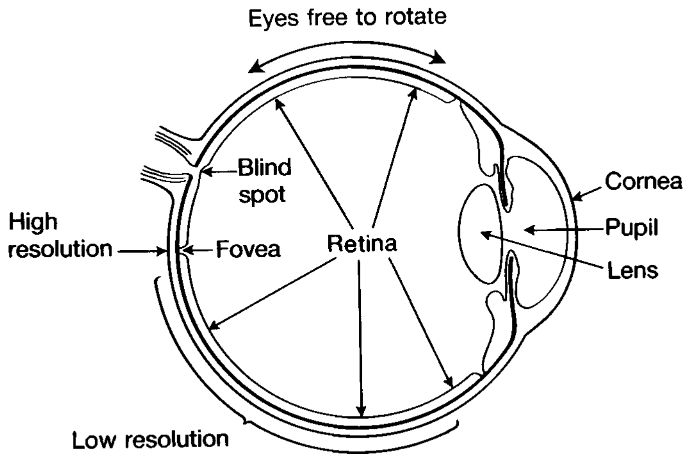


Fig. 1 The human eye. Note, that only that small part of the image which falls on the Fovea, can be seen with high resolution. The rest is "blurred"

Whilst it is attractive to imagine the eye as a sort of camera, this is not an appropriate analogy. Firstly it begs the question "What looks at the picture?". Secondly, the eye is in effect an extension of the surface of the brain, and much signal processing occurs within the back of the retina itself, in groups of cells optimised for specific purposes, such as moving edge detection. The image we see of reality is a software simulation updated by features sensed by our eyes.

2.2 The Resolution (Visual Acuity) of the Eye

The human eye has very high resolution, about one minute of arc, close to the diffraction limit for the size of the optics used. However the eye does not provide high resolution over a wide field of view simultaneously. Central vision is used for the acquisition of fine detail. Only coarse detail is discernible away from the central region (see 4.3.1.1 and [Yager and Davis, 1987]), and peripheral vision is used largely for motion detection. We achieve high resolution perception by sequentially moving the small central high resolution view of the eye, over the scene of interest.

2.3 *Fovea Fixations and Point-of-Gaze*

The fovea is the small, roughly disc shaped area on the retina, which provides the highest visual acuity. It is located close to the central optic axis (the line through the centre of the eye, lens and cornea). The diameter of the fovea corresponds to about 1 degree visual angle. In other words if a person looks at a disc whose diameter subtends one degree visual angle, the image of the disc that is focused on the retina will cover an area about equal to that of the fovea. The word is sometimes used as a verb, to "foveate" meaning to position the eyes so that the image of a certain target or element of the visual scene falls on the fovea. To "fixate" a point in the visual field, implies foveation of that point. In this paper, when we talk about the "point of fixation" or "point-of-gaze" we are indicating the part of the scene being imaged on the fovea.

2.4 *The Blind Spot*

The normal eye has a region which has no response to light, known as the "blind spot". This is where all the connections are made to the eye, i.e. where the blood supply enters and leaves, and where the optic nerve leaves the eye, carrying the visual information to the brain. What is intriguing about the blind spot is that we do not normally know it is there, we are blind to its blindness. It is not perceived as a hole in the visual field, but instead appears to be filled with a stimulus similar to whatever surrounds it. Blind spots caused by damage or disease also fill-in in this way. The blind spots are offset horizontally from our centre of vision by about 15 degrees. The blind spots of left and right eyes are displaced to the left and right sides of our centre of vision respectively, so that the image from one eye provides details of the image missing from the other. However, the fact that we are ignorant of our loss of information when using one eye alone suggests that vision (and in fact all our perception) is in effect an ongoing simulation of reality, which is continually updated by our senses. In Reference 2, Gregory writes: "The large brains of mammals, and particularly humans, allow past experience and anticipation of the future to play a part in augmenting sensory information, so that we do not perceive the world merely from the sensory information available at any given time ...". In other words, what we are familiar with, influences what we actually see.

3 Eye Movements

3.1 *Point-of-Gaze as an Indicator of Point of Attention*

When performing a visual task, the point-of-gaze is a reliable indicator of where our attention lies, especially when the task involves resolving some visual detail. It has been shown that, in the absence of peripheral stimulation, it is not possible to make an eye movement without making a corresponding shift in the focus of attention [Shepherd et al, 1986]. Conversely however, it is possible to shift one's attention without making an eye movement, for example one might start thinking about something else while staring at a word in this piece of text. To summarise, an eye movement usually indicates a shift of attention to that new location.

3.2 Types of Eye Movements

Because of the intimate connection between eye and brain, the study of eye movements is now providing a unique insight into human information processing [Groner et al]. There are two distinct types of eye movements, first a jerky motion jumping from one fixed pointing direction to another, and second a smooth tracking motion [Yarbus, 1967], [Carpenter, 1989]. These may occur independently or simultaneously, and perform very different functions. In addition, eye movements when reading text are highly distinctive.

3.2.1 Fixations and Saccades During normal scanning of a visual scene, eye movement is characterised by a series of stops and very rapid jumps between stopping points. These stops, which normally last at least 100 milliseconds, are called "fixations", and it is during these fixations that most visual information is acquired and processed. The rapid jumps or flicks between fixation points are called "saccades". Saccades are conjugate eye movements (both eyes move together) that can range from 1 to 50 degrees visual angle. They generally have durations from 30 to 120 milliseconds, and achieve angular velocities as high as 600 degrees per second [Carpenter, 1989]. Very little visual information is acquired during saccades, mainly due to blurring caused by the fast motion of the image across the retina, and because the brain partially suppresses information just prior to and during a saccade. When our gaze is attracted to a new spatial location, our gaze jumps towards the new location, but typically undershoots by about 10%, followed by a second or even third corrective saccade (see Fig. 2). Each of these jumps takes time, and delays the moment when any high resolution detail from the scene can be perceived. This highlights the need to reduce the number of events which cause eye movements if the speed of performing a task is important.

This jerky motion of the eye can be felt by placing ones fingers on the closed lid of one eye, whilst looking around with the other. Except for unusual mental states such as unconsciousness, the eyes are rarely completely still for more than a few moments. When we are thinking, we do fixate for quite long periods, but even then the eyes are not completely still. The eye exhibits "micro-saccades", tremor and drift, all of which maintain the image in motion on the retina. This motion is necessary for image perception, for if the image is perfectly stabilised on the retina, it will fade totally within a few seconds [Tulunay-Keesey, 1982]. It is possible to observe this fading, by staring fixedly at a scene which contains only low spatial frequencies.

The reason for this surprising fact can be traced to our evolutionary past. Early creatures used their eyes simply as motion sensors, they lacked the

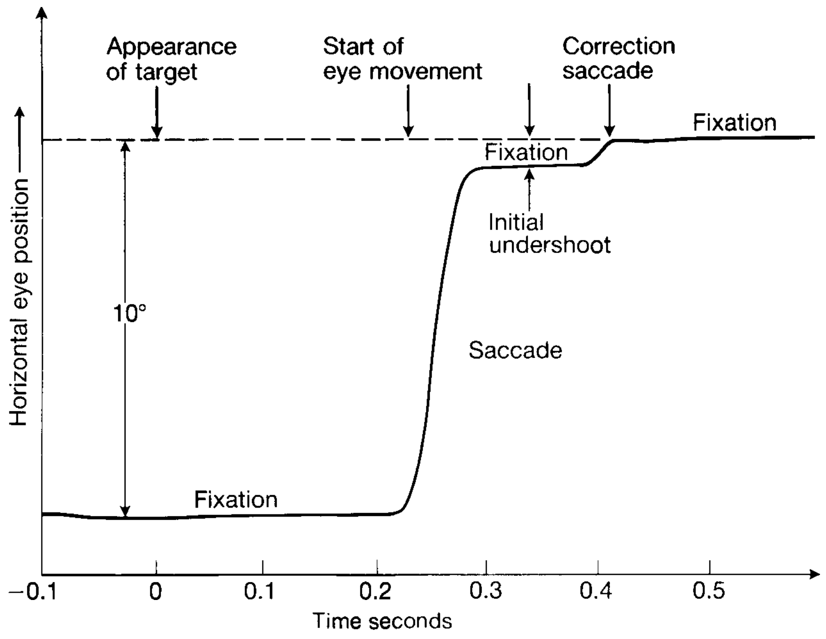


Fig. 2 Plot of the eye pointing direction versus time, when a target suddenly jumps 10 degrees to the right.

This clearly illustrates the abrupt jumps (or Saccades) between Fixations. Note that it is 0.2 of a second before anything happens, and more than 0.4 seconds before the point-of-gaze finally falls on the target. The subject however, thinks he has responded "instantly"

computing hardware to perform any more sophisticated image processing such as object recognition. They kept their eyes still with respect to the surroundings, and the sensitive cells in their eyes rapidly adapted to whatever light fell on them. Cells would then only be triggered if part of the scene were moving, alerting them to pounce or run in the direction determined by the particular cells triggered.

Many higher creatures still stabilise their head and eyes when specifically looking for movement, to suppress all the stationary part of the image (e.g. chickens looking for worms, kestrels looking for mice). We are generally above that sort of thing, and are capable of sophisticated image processing; however we still have light sensitive cells which have no DC response. The solution is to "chop" the signal to get it to fall within the passband of our eyes, and we do this by moving our eyes, well before the image fades. However, we are not normally conscious of this strategy.

3.2.2 Smooth Pursuit The eye can smoothly track targets that are moving in the range of 1 to 30 degrees a second. These conjugate slow tracking eye movements are usually called "smooth pursuit" and their function is partially to stabilise slowly moving features of interest, onto the retina. This is necessary if we are to extract high resolution detail from a moving image. These slow smooth eye movements cannot in general be executed without a slowly moving target. We do however possess another mechanism which compensates for movements of the head. Inertial information from the balance organs in the inner ear is used to provide an equal and opposite rotation of the eyes when the head rotates, to ensure that the eye tracks the image even when the head is not stationary[7]. A consequence of this is that we can also produce slow pursuit eye movements by gazing at a fixed object and turning our head.

3.3 Eye Movements During Reading

Though normal eye movements are not taught, the act of reading is unique, in that it requires a specific eye movement strategy to be trained. Except for very short words, or those anticipated by context, words can only be read by fixating the point-of-gaze upon them [Rayner, 1983, McConkie, 1983, O'Regan, 1987]. When reading at a reasonable speed, the eyes must make a sequence of saccades, most of which are towards the text that is as yet unread. A smaller number of saccades, called "regressions", move the eyes backwards towards a location that was passed earlier in the course of reading. Early readers train their eye movements, usually by following a finger which points at the words in sequence. This provides a target to direct the next saccade. With practice, these directed eye movements become programmed in the brain, and the pointing finger is no longer required.

When reading multi-line text, the eyes make a large saccade to the left and down slightly, to the start of the new line. More often than not, this saccade undershoots, and an additional correction saccade is required to locate the point-of-gaze on the first word of the next line. This is in part a consequence of the poor resolution of the eye off-axis. We cannot accurately predict what size a large saccade should be, because we have no accurate detail about the target until the point-of-gaze gets near to it.

Reading aloud is a slow process limited by the speed of speech, and almost every word is fixated in turn. In contrast, normal reading is for comprehension, not for translation into speech. Reading speed is vitally important and surprising strategies are employed. On average there is one fixation per word, however only 60% of the words are fixated, the rest, often short words like "the" are not fixated at all. This accounts for errors in proof reading short words. In addition, short function words such as "and" and "for", are skipped more often than short content words such as "ate" and "fog" [Hogaboam, 1981].

Long words, of ten or more characters in length, receive on average two fixations per word. Inexperienced readers make more fixations and regressions than fast readers. The fixation durations are about 250 milliseconds on average, and show little reduction with increased reading speed. Fast, and so called "speed" reading are accomplished by fixating fewer words, not by spending significantly shorter time per word. The fact that the reader can "understand" the text, despite acquiring such a small fraction of the image, is a consequence of the high redundancy in language. To be understandable, text must conform to the rules of spatial structure, sensibility, spelling, and context. The rules, known to both writer and reader in advance, reduce uncertainty, making messages partially predictable.

The understanding of reading behaviour is a new science, the study of which is providing valuable insight into the architecture of the brain. There is much debate over the influence which words not being fixated have on the reading process, and to what extent the mental processing of a word is completed prior to the next saccade [Rayner et al, 1987]. There are however measurable factors which can be correlated with reading and comprehension difficulties. Multiple fixations and regressions are frequently a sign of reading difficulty. Multiple fixations and unusually long fixations occur on words which are misspelt, less contextually predictable, or unfamiliar [McConkie, 1983, Rayner et al, 1987].

3.4 Are we aware of our eye movements?

We do not need to be taught how to fixate, make saccades or slow pursuit eye movements, and many who read this paper may previously have been unaware of the very nature of their own eye movements. When we perceive our world, our eye movements secretly assist in acquiring data on the reflectance and spectral properties of key features in our surroundings.

We are also unaware of many of the limitations of our senses, these include the poor resolution away from the centre of vision, the long delay times in responding to visual stimuli, and the blind spot. Presumably, it has been of little evolutionary value to be distracted by knowledge of those limitations which cannot readily be overcome. The processing speed associated with vision, has been at a premium for ensuring our survival. Many of the programs are "hard-wired" to ensure the minimum latency (delay).

Note that in everyday life, we very rarely use eye movements for sending information, apart from the occasional raising of the eyes as a sign of exasperation, or looking away in avoidance of eye contact.

4 Applications of Eye Movement Information

4.1 Eye Movements as a Measurement Tool

4.1.1 Medical Simple eye movement measurement systems, have been used quite widely in the medical field for investigating visual impairments and reading difficulties. Until recently, their use for non medical purposes has been comparatively minor.

4.1.2 Human factors

4.1.2.1 Sports Eye movement analysis has provided a new insight into sports training. For example, it has revealed that in fast ball games, good players do not keep their "eye on the ball", contrary to advice by their coaches. Instead of tracking the ball, they take a few "snapshot" fixations, which enable them to regularly update their predictive model of the ball's trajectory.

4.1.2.2 Usability In order to develop new software and hardware systems which are easy to use, it is important to be able to measure their "usability". In a present day usability evaluation laboratory, video recordings are made of the subject, display, keyboard etc., and keystrokes and mouse movements are recorded, all for subsequent analysis. This provides a record of what the subject is doing physically, but one would really like to know where the subject's attention is located.

During a typical trial, there are significant periods of time when nothing appears to be happening, when there are no body movements or keystrokes. During these times it is vital to know whether the subject is simply daydreaming or whether they are struggling with the system. One would ideally like to know where their attention is focused. The subject can be asked to give a verbal commentary, but speaking interferes with the thinking processes, and is subject to significant delay errors.

There is no record of the point-of-gaze to provide an indication of the point of attention. With appropriately positioned cameras it is possible to judge which quadrant of the screen the subject is looking at, but only if head movements rarely occur. However the video image of the subject's face does not provide sufficient resolution to show what the eyes are looking at.

4.1.2.3 Application to usability measurement Point of gaze measurement systems are increasingly being used for the study and evaluation of human factors [Grat, 1987]. This information may be used at two levels:

i) Simply as an indicator of the location of the subjects attention.
ii) From analysis of the sequence of the fixations and their durations, one may make detailed inferences about the mental processes taking place.

Eye movement measurements can provide a strong indication of which mental processes are taking place. Eye movements during reading are distinctly different from other eye movements, such as searching or staring vacantly (thinking?). Thinking produces long fixations, therefore we can discriminate thinking from reading.

Point-of-gaze measurement provides a technique for following the moment by moment processing of visual information during a task. In tasks where the acquisition of visual information is associated with the spatial arrangement of the environment (for example a vdu display, reference manual layout or position of control knobs) eye movement data should be able to contribute to the assessment of the usability of the system.

The following measurements would be of particular interest: the spatial distribution of fixations over the visual field, the sequence of fixation points, the interval between fixations on the same point of interest in space, the number of fixations required to carry out a particular task, the fixation times etc.

The number of fixations over a given time can be used as a global index of the information acquisition function of a task; the relative frequency of fixations to different points in the visual field, can indicate which sources the subject finds most important. They should provide a powerful tool for understanding difficulties in searching for items on the screen. One indication would be a return of the point-of-gaze to previously fixated items, without having made a keystroke. For example one could determine what is the problem when the subject is looking at a screen full of pulldown menus, but not making keystrokes or any other conventionally observable action. Vertical saccades within a vertical menu list can be distinguished from horizontal saccades between menus.

We are largely unaware of our visual strategy. Many of the perceptual limitations associated with eye movements conflict with the common sense view. We need to develop an understanding of eye movements associated with screen text, windows, icons, and documentation as this will enable the performance of future HCI systems to be improved [Findlay, J. M. et al 1988].

4.1.2.4 Is a Head Mounted System Suitable? There are several practical problems in using eye movement information for usability evaluation. Anything which impedes the subject or obstructs their field of view may interfere with their natural behaviour and be less acceptable. The cheaper presently available spectacle mounted systems are rather cumbersome; however, the remote point-of-gaze measuring systems are considerably more complex and expensive.

Ideally one requires a record of the absolute point-of-gaze versus time; however this in general precludes the use of a simple head referenced measurement system. The exceptions are when used with the head mounted display shown in Fig. 3 and described later, or with correction from a head movement sensor. It is possible, however, that a simple record of the time, orientation and magnitude of the saccades (and not the absolute point-of-gaze), may be sufficient to discriminate between different visual activities.

4.2 Eye Movements for a Real Time Interface

If a computer with a visual display unit, is interfaced to a fast real-time point-of-gaze monitor, some exciting possibilities arise.

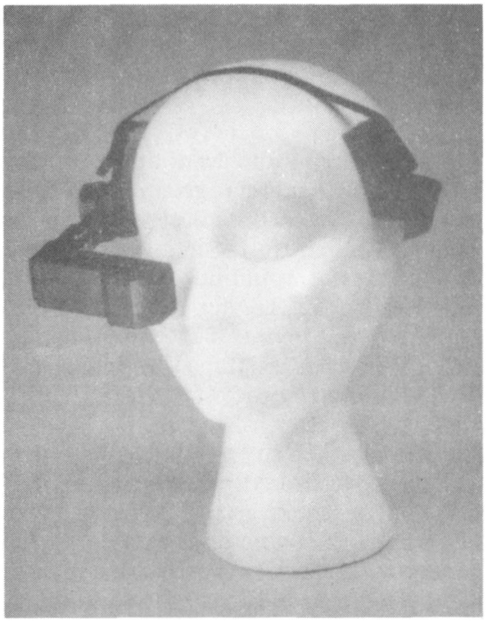


Fig. 3 The "Private Eye", a miniature headband mounted display, which produces a virtual 12 inch image, located 2 feet in front of the wearer. It should be ideal for eye movement controlled applications, such as hands-free Expert systems

4.2.1 Eye Movements for Deliberate Selection Items on a screen can be selected simply by looking at them. Eye movement controlled technology could be used to increase usability in applications like form filling, where data is typed in at various tabbed locations. By using an eye-controlled cursor, the operator would simply have to look at the appropriate box, and start typing.

4.2.1.1 The Eye is Faster than a Mouse Input devices such as the Trackerball and the Mouse are proliferating. If we consider how they are used, we realise that they only go where the eye has gone before.

A typical sequence of actions is:

(i) Look at the point on the screen we wish to move to.
(ii) Look at the present cursor position.
(iii) By a sequence of successive iterations involving both eye and hand movement, bring the cursor to the point on the screen we originally looked at.
(iv) Press a button.

Clearly it is much simpler and faster if we only need to perform the first and last action.

In Ware et al, 1987, a commercial remote point-of-gaze monitor was evaluated as a device for computer input. They found that the time to make menu selections was only two thirds of the time required when using a mouse.

While the maximum resolution of the eye is approximately 1 minute of arc, typically we can only reliably fixate our gaze upon a target to within about one quarter of a degree; the eye unconsciously scans around the object being looked at. This would set a limit to the number of degrees of freedom that can be addressed by the eyes; it would however be quite adequate for many applications [Levine, 1984]. Much greater accuracy can be achieved by using the point-of-gaze of the eyes to control the position of a visible cursor or pointer on the screen, which has a damped response. However, this will be much slower, as it is an iterative process.

In some applications, selection by eye movement may be much more convenient than using a keyboard. People performing physical tasks which require manual skills or hygiene, e.g. aircraft service engineers or hospital surgeons, may wish to access a computer or a database. Conventionally they must either interrupt their task or use speech as an interface. Eye movement controlled selection would be much faster, as it is less ambiguous than speech recognition except for situations with a very limited vocabulary.

However *initiation* by speech ("Yes", "No" etc.), as opposed to *selection*, might still be appropriate [Glenn, 1986].

4.2.2 Unconscious Input via Eye Movements We need not be directly conscious of the information available to an eye monitor. As a simple example, a point-of-gaze measuring system can know which sections of the displayed text have been read, and more specifically which words have been read. This information is very valuable for proof reading. Words not yet fixated upon could be highlighted to draw attention to them. It can also know what part of the screen image we are interested in[6], if any, and respond to that.

4.2.2.1 Responding to Interests and Difficulties An intelligent eye movement monitor system can deduce our interests and difficulties, by watching the trajectory of our point-of-gaze. It can therefore respond to our needs, even those of which we are unaware. One example might be a specially equipped bank cash dispenser which, following the cash transaction, could present a list of topics of possible further interest, e.g. cheap loans. Even if the customer did not respond by pressing a key, the system would be able to present a sublist matched to the customer's interests.

When reading text, the fixation time on each word, and the occurrence of regressive saccades to an earlier point in the text, provide a measure of the degree of difficulty that the subject is experiencing [Rayner et al, 1987]. Long fixation durations on particular words in a text display can be correlated with comprehension difficulties. A computer armed with this knowledge could become powerfully responsive to the needs of both the user and the task. When a "difficult" word is identified, a dictionary or encyclopedia database could be accessed, to provide a simpler but extended description of the word. This could be presented to the user as education, or transparently replace the difficult word in the text. Similarly, difficulties in interpreting the sentence structure could be recognised. So help could be provided whenever assistance seemed to be required, not just when it was specifically requested.

Similarly, software utilising this information could continually adapt menu structures etc., to suit the level of knowledge, skill, experience and fatigue of the operator, in a much more transparent way than is achieved conventionally. It could ensure that the task is challenging and not tedious. This, after all, is what an understanding human communicator would have done, with sufficient time to spare!

4.2.3 Eye Movement Initiated Commands Given that selection is performed by eye, initiation can be achieved by a variety of means. The simplest is by pressing a button, other possibilities include speech, by blowing (as used in the Possum system for the handicapped), or as will be discussed here, by deliberate eye movements.

The benefit of both selection and initiation by eye is that the eyes then become a complete bi-directional human interface. They receive information from the display screen, and control the system via eye movements related to the displayed image.

Though *selection* by eye is a natural act, being almost synonymous with giving something our close attention, deliberate *initiation* of an action by eye movements is not part of our normal experience. Frequently we are unaware of eye movements such as blinks and glances, and so must ensure that the act of initiation is unambiguous.

Although most eye movements are involuntary, we can learn some control. Reading and gesturing with raised eyes are both examples of this. What is needed is one or more simple eye gestures, which are distinguishable from normal eye movements. These would be equivalent to pressing a button. A complex sequence of deliberate eye movements would overcome the ambiguity problem, but could take too long. Possible solutions [Ware et al, 1987] to this are:

(i) The Dwell technique, in which one consciously stares at the item to be selected for a minimum duration.
(ii) Use of an on-screen "button", in which the item is selected by looking at it, but the action initiated only if the gaze subsequently moves to a patch of screen designated as the "button".

There is a further problem: if other items are located on the screen between the item to be selected and the "button", the gaze will pass over them and some means must be found to prevent their accidental selection. By checking that a minimum fixation time is exceeded, we can determine that an item has been consciously fixated. This same test can be applied to the "button" also.

4.2.4 Eye Movement Controlled Technology for the Handicapped The impact of this technology for the physically handicapped will be enormous. They could obviously use such an eye-controlled system provided they have the use of their eyes, and the majority of physically handicapped people do retain full eye movements. Downing [1985] found that with a limited vocabulary of 1000 words, an average of 2·1 selection steps per word are required, and that communication rates of 45 to 50 words per minute should be achieved. For severely handicapped users this is a ten to twenty times increase in communication speed over that achievable by other means.

There is also the exciting possibility of providing a limited sense of "sight" to the blind who still have control of their eye movements. The resulting pointing information could direct an optical or acoustic sensor, and the information returned to the user, either as an acoustic signal, or to a matrix of implanted electrodes.

4.3 Eye Movements for Displays

The author originally developed an interest in eye movement controlled technology through considering what would be the technical requirements for an electronic display which was indistinguishable from reality.

4.3.1 Eye Movement Synchronised Image Generation (EMSIG)

4.3.1.1 The Limitations of Images A conventional image generation system "broadcasts" its image: the whole image can be seen from all possible viewer locations simultaneously, as with a physical object. This has the benefit of being multi-user, but results in serious technical limitations in the resolution of the display. The human observer perceives the world in far greater detail than present day display technology can provide, and over a very wide field of view. Our field of view is about 180 degrees horizontally, and 100 degrees vertically, without including head movements.

Even High Definition TV provides only a thousandth of the perceptual detail which we as humans can observe [Nussbaum, 1987]. Full resolution, full field of view, would require operation at several thousand Gigabits per second, for the camera, display and transmission path. Consequently visual realism is unlikely to be possible for many decades using the conventional "broadcast" image approach, which assumes that the resolution of the display matches that of the human eye over the full field of view.

However the eye only has high resolution (1 minute of arc) over the central 20 minutes of arc. Outside this, the minimum resolvable feature size increases linearly with eccentricity from the centre of our view [Yager et al, 1987]. It is therefore evident that our visual perception information capacity is much smaller than the vast figure implied above. We achieve high resolution perception by sequentially moving the small central high resolution view of the eye around the scene to acquire important feature details.

A conventional display is not energy efficient. Even when sitting close to a TV screen, less than one part per million of the generated light enters the pupils of the viewer's eyes. Furthermore only a small fraction of that light falls on the high resolution region of the retina. This is all as a consequence of an approach which imposes negligible restrictions on the number and location of viewers. For a single viewer it is highly redundant in both optical power and pixels.

4.3.1.2 The EMSIG Technique The large mismatch between the properties of a conventional image and the capabilities of the human eye, suggests a radically different approach: Provide the eye/brain only with that part of the image which it can perceive.

This approach, which we call Eye Movement Synchronised Image Generation (EMSIG), requires different image information to be transmitted to each observer (and to each eye for 3D). The portion of the scene to be transmitted is determined by the pointing direction of the observer's eye(s) and this information must be transmitted from the viewer to the image source to control the pointing direction of the camera. Note that multiple viewers would require multiple sets of hardware.

It is interesting to note that this technique will be capable of providing far greater realism (and sense of depth) than is possible using a hologram. A hologram simultaneously "broadcasts" a large number of different images. It requires vastly greater information, to project a two-dimensional image to all possible viewing positions and directions, than is required by a single observer with just one pair of eyes.

The EMSIG technique will enable the pixel and power redundancy to be traded in exchange for greater visual realism (or reduced information capacity), thus overcoming the fundamental technical barrier which is inherent in the conventional method. It has far reaching implications for high resolution computer displays, Viewphone, remote inspection in hazardous areas, surveillance, simulators, entertainment etc., and ultimately offers complete visual realism.

4.3.1.3 Hardware Required for EMSIG

(i) A point-of-gaze monitor.
(ii) A display which tracks the point-of-gaze.

If the display is head mounted, then an additional head orientation sensor is required. The miniature head-mounted personal display devices described later (see Figure 6), are probably very suitable.

4.3.1.4 Applications of EMSIG Consider two different kinds of source image material:

4.3.1.4.1 Viewphone Present day approaches to viewphone do not allow high quality video without needing prohibitively high capacity, hence viewphone is not a very exciting step, as indicated by the slow rate of growth. Viewphone is ideally suited to EMSIG for the following reasons:-

Firstly communication is always bi-directional, enabling the control signals to be sent to the camera.

Secondly the improvement in definition, together with a perceived field of view of 360 degrees, could provide an undreamed of level of realism; true stereoscopic vision would also be possible. For the first time, technology could provide the experience of "being there".

The sound of course could also be stereo, sensed in the same orientation as the camera, providing the realism associated with "dummy head" recordings, plus the ability to explore the acoustic scene by moving the observer's head in the normal way. The orientation of the sensing "head" (camera + mike) would track the orientation of the eyes of the remote viewer, however its position might also be variable and controlled by the movement of the observer's head. Ultimately the system could control, and experience through, a robot which "became" the particular viewphone caller. These techniques would also be invaluable for surveillance and use in hazardous environments.

4.3.1.4.2 Computer Graphics Its use is also envisaged in any application requiring high resolution graphics. For example a complete VLSI chip could be displayed with as much or greater resolution than is provided by hard copy. Word processing could provide a WYSIWYG (What You See Is What You Get) display, on any size "paper", even full newspaper size.

Whilst EMSIG Viewphone would allow reality to be experienced at a distance, EMSIG computer graphics will allow the simulation of any visual scene which we can both imagine and compute. The resolution and field of view limitations of the display will be overcome, the limitation will then be the speed of computation. The enormous power of this technique for real-time simulation is now being exploited by many of the flying simulator manufacturers. Modern combat aircraft like the F16 provide the pilot with an almost all-round, unobstructed field of view. To simulate this view realistically is a far harder task than for earlier aircraft with restricted field of view. The approach being pursued, is to project a low resolution wide-angle image, and to patch-in high resolution detail around the point-of-gaze [Lewis et al, 1976].

4.3.2 Future Avionics, Displays, and Control by Eye Movements In the cockpits of modern aircraft, the many different display devices are being replaced by a few electronic displays; however it is now widely believed that in the next century, the pilot's display will replace the cockpit itself. This "virtual cockpit environment" is being developed by several avionics companies in the USA, and through "Cockpit 2000", a collaborative programme in Europe. These programmes are looking at EMSIG techniques which will combine physical images from cameras, with computer graphics in a single display.

The benefits include:

(i) Integration of data from all types of sensor, onto a single image, e.g. infra-red, radar, computed target status, plus visual image from cameras.
(ii) Optimally positioning the pilot to:
    A. Endure high G forces, prone position allows tighter turns.
    B. Eliminate the weak-point in the airframe resulting from the need for a transparent canopy.
    The pilot may even be located in another remote stand-off aircraft or on the ground.
(iii) Allow unrestricted view, e.g. seeing through the underside of the aircraft.
(iv) Ease of reconfigurability.

It is envisaged that eye movements will also be used to operate synthetic cockpit switches and for weapons aiming. This avoids the problems of hand and arm movements in a high G environment.

These defence applications will accelerate progress in the technology and techniques of head mounted displays and eye movement monitors, irrespective of their use in civil applications.

4.3.3 Gaze Controlled Conference Vision Lastly, consider the application of gaze information to conference vision. In a normal group conversation, we look at someone to indicate that a statement is addressed to them. In conventional conference vision, several individual images from each location are patched together to form a group. Now when we talk to anyone in this "group", none of the individuals know which one we are looking at. We cannot establish eye contact in the normal sense. A point-of-gaze monitor would be able to recognise which person is being looked at, and this information could be transmitted to that person. For example, at my monitor, the image of whichever individual is addressing me could be highlighted.

If sufficient bandwidth were available, more than one image of each member of the group could be transmitted, e.g. one slightly to the left, one face on, and one slightly to the right. The conference vision system could then synthesise a "seating arrangement" in which I would see face of the person who is looking at me, while all other members would see that same person looking to their left, or to their right.

5 Point of Gaze Measuring Techniques

5.1 How do We know when We are being looked at?

In group situations, we are usually aware when someone is looking at us. It is an important part of human communication to know whether a statement such as "how are you today?" is addressed to us or not. We do this by observing the orientation of the face and eyes. We see if they are facing us directly, we also look for asymmetry in the whites of the eyes, and ellipticity of the iris. It is quite surprising what accuracy we do achieve. Some of the opto-electronic measurement systems described in this section use similar methods.

5.2 Practical Point-of-Gaze Measuring Systems

For most of the ideas proposed in this paper, some means of measuring where the subject is looking is required. There is no means of directly measuring where the subject's attention is located, but there are a variety of methods of measuring where the eyes are pointing, as described in the very comprehensive review by L.R. Young and D. Sheena in Reference [Young et al, 1975]. Some of these measure the orientation of the eyes with respect to the head, whilst others measure their orientation with respect to an external frame of reference. Some of these systems are only laboratory research tools, whilst others have been widely used for evaluating the eye movements of naive subjects.

5.2.1 Measurement Frame of Reference If the objective is to determine the point-of-gaze within a displayed image, then the measurements must be referenced to that display. Most displays are in a stationary frame of reference, (with the exception of some avionics systems and the recently announced headband mounted PC display shown in Fig. 3), and most cheap eye monitors are head referenced. This problem can be resolved in a laboratory environment by keeping the head still, and can be achieved either by i) some kind of physical restraint such as a chin rest or a bite board, or ii) as in the case of our own experiments, simply by consciously keeping the head still during the experiments, together with frequent recalibration of the offset error. In some experiments it is convenient to provide a real-time indication of where the system thinks the subject is looking. This allows the subject to minimise the error, simply by tilting the head.

One easy way to refer head-referenced measurements to the stationary frame of reference is to mount a small forward-facing scene camera onto the head. The eye monitor signals may then be superimposed onto the video image as markers. The advantage of this technique is freedom of movement, the wearer can turn through any angle and walk around without interfering with the measurement. This approach has been used widely for the study of human factors, in situations where the required output is an image with a superimposed point-of-gaze marker. It is less attractive where an electronic signal is required at the output which tells us precisely where the subject is looking.

5.2.2 Calibration of Gain and Linearity None of the systems is perfect; high accuracy is achievable only by some initial calibration procedure which adjusts the raw data. Furthermore the reflectivity and dimensions of eyes, vary from subject to subject, as does the head shape. Calibration may be done manually by adjusting potentiometers, but the procedure is easily automated. A mark flashed on the screen attracts the subject's attention, and the system reads the measured point-of-gaze. This sequence is repeated for several screen locations, and a correction algorithm derived, which is applied to all subsequent point-of-gaze measurements. This may be updated automatically whenever the system knows what the subject is supposed to be looking at.

5.2.3 Commonly Used Eye Movement Measuring Systems As one might expect, most of the systems measure the eye orientation optically, but there are a few non-optical techniques [Young et al, 1975]. All the systems described are commercially available. Most of these systems have been made only in small quantities, so as one would expect, prices are significantly higher than the component costs.

A. Limbus tracking system
Most cheap spectacle mounted eye movement monitors are of this type, shown in Fig. 4. The Limbus is the boundary between the white of the eye and the iris. When the eye rotates in the horizontal plane the amount of white exposed on each side changes. The simpler limbus tracking systems flood the eye with infra-red and monitor the reflected light on each side of the iris. The differential signal is a measure of the angle of the eyeball in the head. Apart from the disadvantage of the measurement being head referenced, the major problem is how to measure vertical movements, as the upper and lower edges of the iris are usually obscured by the upper and lower eyelids respectively. For this reason many limbus trackers are sold as horizontal eye movement monitors only. Using sensors on both eyes enables the angle of stereopsis to be measured. This is a measure of the distance in space at which the two images converge.

Fortunately and rather surprisingly, the eyelids track the vertical rotation of the eyeball. If the infra-red sensors are adjusted to monitor the boundary between the eyelid and the eyeball then vertical eye movements can be measured. This is not as accurate as the horizontal measurement, as changes in facial expression cause the eyes to "narrow", i.e. the lids to move. Smiling, for example, causes the lower lid to rise and the upper lid to fall. Blinks of course interfere with all the optical measuring techniques, but the tiniest blink will produce a large error if we are monitoring the upper lid, consequently the lower lid is usually used, though smiling affects this one more.

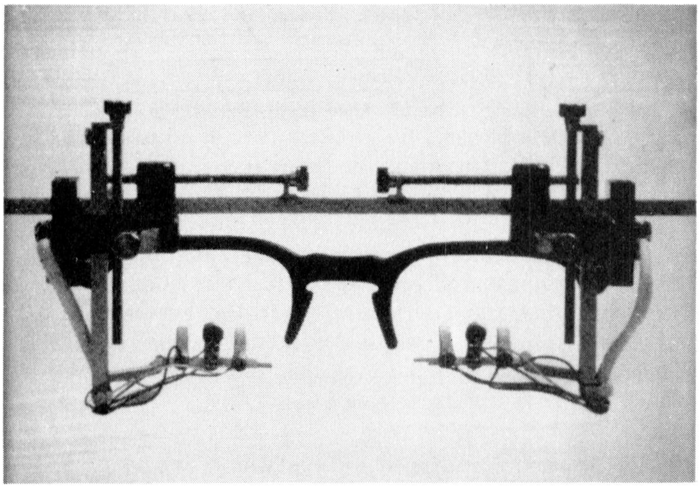


Fig. 4 Spectacle mounted commercial Limbus tracking eye movement monitor. The eyes are illuminated by infra-red emitting diodes, and the light reflected from the whites of the eyes, adjacent to the iris. Eye movement is sensed by pairs of photodetectors

Horizontal and vertical movements are usually measured on different eyes. In normal subjects this is acceptable and it results in less obscuring of the field of view by the sensors. It offers reasonable accuracy for horizontal measurements and is cheap, so is ideal for measuring reading behaviour. Range + & − 15 degrees: accuracy 1 degree horiz., 2 degrees vert: referenced to head: cost £2k upwards.

B. Pupil-centre Corneal-reflection system

In this an image of the eye is processed to determine the separation between the centre of the pupil and a bright reflection from the cornea, the outer surface of the eye [Eizenman, 1984]. Typically a miniature TV camera produces an image of the eye, and the system computes the centre of the pupil (C), from the pupil–iris boundary. Simultaneously an infra-red light is reflected off the cornea to produce a bright "glint in the eye" (G). When the eyeball rotates, the relative position of these changes. The pointing angle of the eye is computed from the separation and relative orientation of C and G.

This system is one of the few that can provide accurate point-of-gaze signals without requiring laboratory conditions. It has been widely used in both head mounted and remote configurations. In head mounted configuration the measurements are head referenced. The remote system uses tracking optics which follow the subjects eye over a limited range of head movements, allowing it to be used in informal conditions, such as for evaluating advertising material. As a side effect of calculating the centre of the pupil, this technique also measures pupil diameter which responds to mental workload and emotional response. The cost is from £50k upwards.

C. Double–Purkinje–Image (DPI)
This monitors and tracks the reflections from the surface of the cornea, and from the lens/aqueous humour interface, using infra-red [Crane et al, 1985]. The separation of the two images is proportional to the angular rotation of the eye. Servo control is used to track eye rotation by forcing image separation to zero. This technique has been refined to produce quite remarkable performance as follows:

Generation V (DPI) eyetracker:

Resolution	20 seconds of arc RMS
Tracking bandwidth	500 Hz
Slew rate	2000 degrees/sec, (peak for eye is 700)
Tracking range	20 to 30 degrees

This is the only system which optically tracks the eye, the others simply measure the movement of the point-of-gaze or the physical rotation of the eye ball. This system is sufficiently fast and accurate to track anything the eye can do, and hence is ideal for experiments with image stabilisation; however it is difficult to set up and is bulky and expensive. Cost £50k upwards.

D. Scleral coil contact lens method.
This senses the eyeball orientation magnetically with reference to an external magnetic field, by means of a sensing coil mounted in a special contact lens. It provides very high resolution accuracy (1 minute of arc), and is insensitive to head movement. Unfortunately it is inconvenient, as the subject is required to wear a contact lens connected by wires, and sit with his head in the space between a pair of large exciting coils. The signal picked up by the coil in the contact lens is a measure of the pointing angle of the eye. As it provides high accuracy at low cost, the hardware is available in several universities; however it is only suitable for research. Other problems are contact lens slip: cost, £1k.

E. Electro-oculography.
The eyeball naturally generates a DC electrical potential from front to back due to the difference in the metabolic rate, and this potential is detectable at the surface of the head. Electrodes placed on the skin beside the eyes allow measurement of the eye position. Drift is a problem when the scene brightness changes because this changes the metabolic rate, and hence the voltage. The technique is inconvenient as electrode jelly must be used. It does however have one major advantage in that blinks do not interfere with the readings.

The sensitivity varies from 10 to 40 microvolt per degree with increased brightness. The accuracy is plus or minus 1/2 degree.
The measurement is head-referenced.
The system is very cheap: cost, less than £1k.

6 Experimental Demonstrations

We felt it was important to gain some first-hand practical experience with eye movement controlled technology. We therefore decided to establish a hardware demonstration of some of the possibilities discussed earlier.

6.1 Hardware

An initial survey was conducted, to determine the most suitable means of obtaining electrical signals corresponding to the point-of-gaze of the eyes, with sufficient speed of response for our purposes [McConkie et al, 1984]. We chose to use a "limbus tracking" eye monitor system, being low cost and simple to use. The model was an Eye-trac 210, made by Applied Science Labs. This spectacle frame mounted system is head referenced, so is not referenced to the image on the display. Movement of the head therefore produces large errors. A head restraint frame was supplied with the system, however this was inconvenient to use, and it proved quite easy to just hold the head still during the experiments. This was assisted in many of the experiments by the presentation of an indicator on screen of the computed point-of-gaze. It was then possible to see the direction and magnitude of the error and correct it by tilting the head slightly.

The output signals from the eye movement monitor were interfaced to an Acorn Archimedes computer, this being chosen for its speed, graphics capability and low cost. The complete set-up is shown in Fig. 5. The basic eye

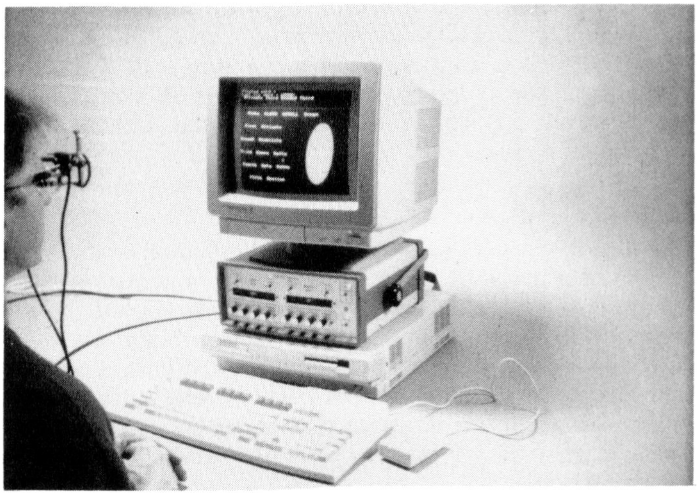


Fig. 5 The author using the experimental eye movement controlled system. Items are selected simply by looking at them, and actions initiated by subsequently looking at the "on-screen button". The hardware consists of an Archimedes computer and monitor, interfaced to an Applied Science Labs spectacle-mounted eye movement monitor

monitor is crude and must be re-calibrated for each subject and each occasion, if reasonable correlation between the measured and actual point-of-gaze is to be achieved. Potentiometers are provided on the front panel of the eye movement monitor for adjusting the gain, offset, crosstalk and linearity for Horizontal and Vertical signs respectively.

6.2 Auto-calibration

Unfortunately, this manual setting up procedure was time consuming and inconvenient, so an automatic calibration procedure for offset, gain and crosstalk was provided via the software. This functions by presenting a sequence of five fixation points at the top, bottom, left, right and centre of the screen. The subject simply presses the space bar whilst looking at each point as it is presented. The software then computes a correction algorithm, and corrects all subsequent inputs for offset gain and crosstalk. Because head movements produce offset errors, the offset may be recalibrated at any time during some of the subsequent demonstrations, by pressing the space bar. A central fixation point then appears and a second press corrects the offset and returns to the original demonstration.

6.3 Set of Demonstrations

For simplicity, the following four items use only horizontal eye movement information. They provide autocalibration of offset and gain alone, by the initial presentation of two fixation points, on the left and right hand sides of the screen.

1. Eye Controlled Menu. All the demonstrations, with the exception of the ones on usability, are selected using an eye-controlled Menu program. This uses eye movements to select the subsequent demonstration program. The computer knows which program name is being looked at, and pressing the space bar causes that program to be run or to access alternative menus.

2. Various programs to show the fundamental properties of eye movements and measurements. These provide a live demonstration of eye movement behaviour, graphically illustrating the long delays in our response to visual information and the difference between saccades, fixations and slow pursuit.

3. Simple fixation duration monitor. This identifies horizontal saccades when the point-of-gaze jumps to a new fixation point, and it times the fixation durations. This program displays a graph of the eye position, and velocity, plus a list of the fixation times. When text is read by the subject, it clearly demonstrates the differences in eye movements for easy text, difficult text, reading aloud, and comprehension difficulties.

4. Eye movement controlled scrolling of text within a one line window. This program displays a file of text to the operator, one line at a time. The next

line of text automatically overwrites the previous line, when the point-of-gaze has first got near to the right hand end of the line, and then subsequently flicks back to look for the start of the next line. The result is that multi-line text may be read on a single line display.

In addition, the previously read text is displayed as a scrolling chart display, upon which the points at which the eye fixates are recorded. This provides a record of the recent eye movements. The word being fixated from moment to moment is also identified and displayed.

The following demonstrations all make use of the full autocalibration routine described earlier in 6.2.

5. Point-of-gaze display. This monitors both the X and Y coordinates, and plots the fixation points on the screen. It allows us to see the trajectory of eye movements, which are produced by looking at different types of image and in different ways. Multiple fixations occur around features of interest. This kind of information is useful in evaluating usability.

6. Identify word fixated within multiple lines of text. This is a much more demanding task for the eye monitor for two reasons: Firstly, a typical word is much wider than its height, and so, in normally spaced text, the vertical separation of words is less than the horizontal spacing. Secondly the vertical accuracy of our Limbus tracking eye monitor, is inferior to the horizontal accuracy. This is because the vertical signal is derived from the movement of the lower eyelid, and facial movements, such as smiles and frowns, cause large errors.

   For this demonstration we therefore chose to increase the line spacing to three. A screen of text is presented, and the sequence of words fixated and the fixation durations, are written to an array. On exit, the sequence of words fixated and their durations is displayed on the screen, together with the original text. This dramatically demonstrates that a large proportion of the shorter words are skipped over, and that abnormally long fixation durations occur for difficult words.

7. Advanced Menu, totally controlled by eye movements. The initial program selection menu described earlier, i) could only select from a horizontal line of words, and ii) required a key press to initiate the action. This menu allows selection from items anywhere on the screen, and provides both selection and initiation by eye movements alone. Initiation is achieved by means of an eye-controlled on-screen "button".

   Several problems had to be overcome. Accidental selection and initiation when the gaze scans across an unwanted item, must be prevented. Command by eye is not a natural act, and it is undesirable for the operator, to have to be aware of where their eyes are looking all the time, in order to avoid an erroneous action.

We are only aware that we are looking at something if we have fixated it, that is if the point-of-gaze has been stationary on it for a minimum length of time. As a conscious fixation lasts at least 200 milliseconds, the menu ignores any items looked at for shorter durations. So if you didn't know you had looked at it, you could not select it.

Any item consciously fixated is "selected", but we do not wish to initiate an action by this alone. The act of choosing one from several items will probably involve fixating all the items in turn. Possible alternative eye-controlled initiation techniques [Ware et al, 1987] are:

(i) a deliberately long fixation on the item selected.
(ii) fixation on an on-screen "button", immediately following fixation on the item required.

We chose to use the latter, using a coloured patch for the button. Once again, to avoid false initiations, the button could only be activated by fixating on it for more than 200 milliseconds.

When using this menu, the procedure is to look at the item required, then look at the button. Because we check for conscious fixations, the scan path can pass over other menu items without false selections or initiations. This allows items to be placed anywhere on the screen. Passing over an unwanted item between selecting an item and looking at the button does not produce errors. The total delay (>400 mSec) may seem long, but it is considerably faster than when using a mouse. This particular demonstration is shown in Fig. 5.

This Menu would be suitable for hands-free expert systems, and for the physically handicapped.

8. Eye Movement Synchronised Image Generation (EMSIG). A full demonstration of the concept using moving images was beyond the scope of this project. We therefore decided to demonstrate the EMSIG concept, with a moving window on a stationary image. We used two static screens, one of text or a picture, and the other a spatially low pass filtered or blurred image of the first screen. The high resolution screen is viewed through a small window in the low resolution screen, and this window moves around the screen to track the point-of-gaze as shown in Fig. 6. To the normal observer, the greater part of the image is blurred, with a small non-blurred region which darts around. The text is blurred beyond comprehension. However, to an observer wearing the eye monitor, the image is always sharp where it is being read or looked at, and the text can be read without difficulty. This demonstration is a crude attempt to mimic the gradual fall-off in resolution of the eye with eccentricity. In our demonstration, the abrupt step in the resolution is visible but does not interfere.

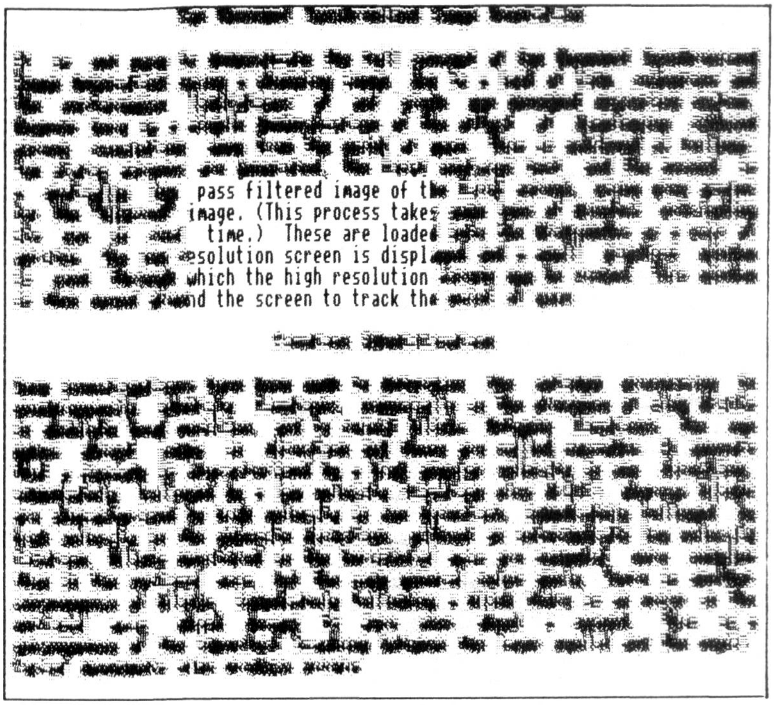

pass filtered image of t
image. (This process takes
time.) These are loade
esolution screen is displ
which the high resolution
d the screen to track th

Fig. 6 Demonstration of the concept of Eye Movement Synchronised Image Generation.

The high resolution window, moves around within the low resolution screen, to track the point-of-gaze of the subject

The blurred image only had 1/64th of the information content of the high resolution image, so this shows the potential benefits in information capacity. The high resolution window needs to be at least one character high, by about 15 characters wide; however the errors in the eye monitor forced us to use a larger window. The problems of vertical errors mentioned earlier, forced us to set the window to be several lines in height.

When the low resolution image is blanked, leaving only the high resolution window which tracks the point-of-gaze, reading is very seriously impaired. This demonstrates the importance of low resolution information in non-central vision, despite its illegibility.

The effect of additional delays was investigated, to see whether transmission delays in viewphone applications would be a problem. An extra 30 milliseconds was just noticeable, with 50 msec being acceptable. The latter would correspond to 5,000 kilometres of round trip transmission delay through optical fibre.

9. Usability. A set of three programs were written to illustrate the potential value of eye movement information in usability evaluation. The first pair of programs write and read the eye movement data, to and from a file, and provide some statistical analysis of the data. These show the distinctive characteristics of eye movements during reading, compared with searching.

   The third program shows that eye movement data can be acquired from our head referenced point-of-gaze monitor and interpreted, despite simultaneous head movement. This is achieved by recognising the rapid flicks of the eye (saccades) from the acceleration profile, and logging the eye position immediately before and after each saccade. This enables automatic recognition of the act of reading, from the ratio of small saccades to the right to the number of large saccades to the left. Text reading is therefore easily distinguished from searching and thinking, even during times of head movement.

7 The Future

The sales of professional point-of-gaze monitors continue to increase as people become more aware of the commercial value of eye movement information in product design. Ease of use of future systems should be considerably enhanced by awareness of the delay penalties associated with eye movements.

Athough the use of eye movement controlled technology is in its infancy, rapid progress is being made in some specific areas. The defence applications will force the pace of development of eye monitor and display technology, as cost is not a premium. However the potential for explosive growth in consumer applications should not be overlooked, the Compact Disc and the Walkman being good examples, where complexity has not been a barrier.

At the start of this project we believed that the key enabling component for many of the applications described would be a cheap headband-mounted display device. As eye movements are specific to one individual, it makes sense to have a personal display, with the benefits of mobility, small size and weight. This avoids the need for an expensive remote point-of-gaze monitor. It should be a simple task to mount a limbus eye monitor onto the display, avoiding the problems of head movement mentioned earlier.

Recently just such a display has been announced, suitable for use with a laptop PC. Though this is the first of its kind, the "Private Eye" is priced at less than $500. It provides a virtual image of a 12 inch monitor by means of a headband mounted module only 1 by 1·2 by 3·2 inches in size, and weighing less than 2 ounces, and is shown in Fig. 3. It should find applications wherever a high resolution display is required in a small space, or where the operator requires both unrestricted movement and continual access to a display.

Eye controlled input would enable hands-free use of expert systems. The addition of a simple head movement monitor, would allow a virtual display with a much larger field of view to be generated.

8 Conclusions

The limitations and capabilities of the human eye have been described. In order to acquire high resolution detail we must image that part of the scene onto the small region of the retina with high resolution, the fovea. Because of this, the pointing direction of the eye is, in the majority of situations, an indicator of where our attention lies.

Eye movements are complex in their nature, consisting of rapid jerks interposed by brief periods when the gaze is stationary or "fixated" upon a part of the scene. The location and duration of these fixations are an indicator of the area of interest and of comprehension difficulties. The availability of this information in real-time, makes many things possible. Eye movements may be used instead of a mouse, to select items on-screen and actions may be initiated by gestures of the eye. This should have great value in hands-free expert systems. As the difficulty of a task may be sensed automatically, software could become powerfully responsive to the needs of the operator. Our eyes therefore become a bidirectional interface with a machine.

There is a vast mismatch between the information rate required for conventional images, and the information rate which we can perceive. Eye Movement Synchronised Image Generation enables this imbalance to be reduced, a technique now being applied in the latest flying simulators.

A range of demonstrations were developed to show the potential of eye movement controlled technology, and these have been described. Direct experience and understanding of visual perception and eye movements, will allow future products, in which vision plays a part, to be more effective and competitive.

Acknowledgements

The author would like to thank the management of STL for their support, and colleagues from the User Centred systems department and Optical Systems department, for helpful discussions.

References

CARPENTER, R.: *Eye Motion Machinery.* Physics World 2, February 1989 pp. 41–44.
CRANE, H.D., STEELE, C.M.: *Generation-V dual-purkinje-image eyetracker.* pp. 527–537 Applied Optics 15 Feb. 1985, Vol. 24, No. 4.
DOWNING, A.R.: *Eye controlled and other fast communicators for speech impaired and physically handicapped persons.* Australasian Physical & Engineering Sciences in Medicine (1985) Vol. 8, No. 1, pp. 17–21.

EIZENMAN, M. *et al.*: *Non contacting measurement of eye movements using the corneal reflex.* Vision Research, 1984, 24(2), pp. 167–174.

FINDLAY, J.M. *et al.*: *Optimum display arrangements for presenting visual reminders.* Proceedings of HCI'88.

GLENN III, F. *et al.*: *Eye–Voice–Controlled Interface.* Proc Human Factors Soc. 30th Annual meeting 1986.

GRAF W. *et al.*: *Methods for the ergonomical evaluation of alphanumeric computer-generated displays.* In Human Computer Interaction-INTERACT'87, H. Bullinger and B. Shackel (Editors), Elsevier Science Publishers B.V. (North-Holland), IFIP, 1987, pp. 349–353.

GREGORY, R.L.: *Eye and Brain, The Psychology of Seeing.* World University Library. Weidenfeld and Nicolson, 1966.

GRONER, R., McCONKIE, G.W. and MENZ, C.: *Eye Movements and Human Information Processing.* Proceedings of Symposium on "Eye Movements and Psychological processes", Vol. 6, XXIII International Congress of Psychology. Publishers, North Holland, Elsevier Science Publishers B.V.

HOGABOAM, T.W.: *The rocky road from eye fixations to comprehension.* (Technical report No. 207), Urbana, University of Illinois, Centre for the study of reading, May, 1981.

KRUEGER, M. *et al.*: *VIDEOPLACE, An Artificial Reality,* Proceedings of CHI'85., 1985 ACM 0-89791-149-0/85/004/0035.

LEVINE, J.: *Performance of an Eyetracker for Office Use.* Comput. Biol. Medic Vol. 14, No. 1, pp. 77–89, 1984.

LEWIS, E. and AMOS, B.: *A high resolution vision system for aircraft and trainers.* NAECON'76 RECORD pp. 894–902.

McCONKIE, G.W.: *Eye movements and perception during reading.* New York: Academic Press, 1983.

McCONKIE G.W. *et al.*: *Instrumentation considerations in research involving eye-movement contingent stimulus control.* GALE, A.G. and JOHNSON, F. (Eds.): *Theoretical and Applied Aspects of Eye Movement Research.* Elsevier Science Publishers B.V. (North-Holland), 1984, pp. 39–47.

NUSSBAUM, E.: *Integrated broadband networks and services of the future.* Paper MB2, Proceedings of OFC/IOOC 1987, Reno Nevada, Jan. 19–23, 1987.

O'REGAN, K. *et al.*: *Eye Movements and tactics in word recognition and reading.* In "M. Coltheart (Ed.), Attention and performance XII; The Psychology of reading", Erlbaum, Hillsdale, N.J., 1987, pp. 363–383.

RAYNER, K. (Ed.): *Eye Movements in reading: Perceptual and language processes.* New York: Academic Press, 1983.

RAYNER, K. and DUFFY, S.: *Eye movements and lexical ambiguity.* In Eye Movements: From Physiology to Cognition, J.K. O'Regan and A. Levy-Schoen (Eds.), Elsevier Science Publishers B.V. (North-Holland), 1987, pp. 521–529.

SHEPHERD, M., FINDLAY, J.M., HOCKEY, R.J.: *The Relationship between Eye Movements and Spatial Attention.* Quarterly Jnl. of Experimental Psychology, 38A, pp. 475–491, 1986.

TULUNAY-KEESEY, ULKER.: *Fading of stabilised retinal images.* pp. 440–447, J. Opt. Soc. Am. Vol. 72, No. 4, April 1982.

WARE, C. and MIKAELIAN, H.H.: *An Evaluation of an Eye Tracker as a Device for Computer Input.* SIGCHI Bull. (USA) spec. issue. 1987 pp. 183–188.

YAGER, D. and DAVIS, E.T. (Eds.): *Variations of visual functions across the visual field.* Special feature in Jnl. Opt. Soc. Am. A, Optics and Image Science, Vol. 4, No. 8, Aug. 1987.

YARBUS, A.L.: *Eye movements and vision.* New York: Plenum Press, 1967.

YOUNG, L. and SHEENA, D.: *Methods and Designs, Survey of Eye Movement recording Methods.* Behaviour Research Methods and Instrumentation 1975, Vol. 7(5), pp. 397–429.

Government IT Infrastructure for the Nineties (GIN); an Introduction to the Programme

Peter Wiles
ICL Central Government & Defence Business Unit, Slough, Berks, UK.

Abstract

This article is the first of a series which will describe various technical aspects of a major ICL project, the GIN Programme. GIN stands for Government Infrastructure for the Nineties. It is a programme aimed at delivering an integrated, interworking set of UNIX* and PC products managed via a range of Systems Management products.

The GIN programme responds to the future Information Technology (IT) strategies of several major customers, in particular some of the largest Government Departments. Government doesn't have much in the way of common industry-wide applications, but does have requirements for an integrated set of infrastructure products which are in advance of the market as a whole. Infrastructure, in this sense, is used to describe the set of vendor-supplied hardware, software and networking products which enable the customer to implement and exploit his own applications. Government Departments often wish to install many hundreds of UNIX systems in many different locations, all providing similar applications to their local staff and all interconnected via a Wide Area Network. The interworking and management requirements of networks of this type go well beyond the free-and-easy environment for which UNIX was originally developed, and pose an interesting set of development requirements.

This article is intended as an introduction to the GIN programme. It describes the most important of these requirements, and discusses some of the technical issues which arise when planning a response to them.

1 Introduction

This article describes the justification and requirements for a major ICL strategy known as GIN, which stands for Government Infrastructure for the Nineties. It discusses the customer requirements which led to this programme, and some of the technical issues which arise from it. Later issues of the Technical Journal will contain a number of papers covering some of these technical issues in more detail; the purpose of the present paper is to articulate the requirements, to set the scene and to emphasise the need for an integrated approach to a wide range of product strategies.

The GIN programme is a response by ICL to the emerging needs of a number of large Government Departments. Many of these Departments have spent the 1980s constructing large Transaction Processing systems on mainframe computer systems. These systems are used by many thousands of their staff, and in many cases they are crucial to the objectives of the Department. However in parallel with these developments, other staff within the Departments have been buying PCs, usually to run some specific package. Few of these PCs are connected together in any way. But now a number of new and sometimes conflicting requirements are beginning to emerge, and have a profound influence on the types of computer systems that these Departments expect to procure in the future.

- Users whose sole source of IT is the large corporate Transaction Processing service are now demanding access to other services.
- Based on their experience with the sophisticated graphical interfaces often presented by their PC applications, they are now demanding equally sophisticated interface styles to their mainframe applications.
- Users want to be able to connect their PCs together, and use them (among other things) to access the corporate mainframe services.
- However managers in charge of security are very wary of this latter requirement, because of the scope for data corruption and virus invasion that it brings.
- Customer IT managers are looking for insurance against the potentially disastrous loss of one or more of their mainframe computer systems; they see the gradual spread of X.25 networks as offering the opportunity to be able to switch a large population of users from one mainframe to another very quickly, and look to their suppliers to propose ways in which this requirement can be met.
- Departments are concerned about being tied to a single supplier. They see the emergence of "open systems" as a way of avoiding this situation, and look to the industry to support the appropriate standards for operating systems and communications facilities. Of particular interest, they look for open systems in the communications cluster controllers and application servers located within individual offices. In practice this means that they are seeking to install UNIX systems in large numbers in distributed offices, and look to the industry to migrate existing functionality onto UNIX.
- However, the complexity of UNIX systems imposes a support requirement which was not present with cluster controllers.

When they are examined in detail, these requirements are not unique to Government. They provide an early indication of the direction in which many other large IT users will wish to move in the next few years. This paper examines these requirements, and shows how emerging technology trends also influence our customer's expectations. It discusses an architectural framework within which it is possible to meet the requirements both of Government and of many other customers. This framework forms the basis for a major ICL programme which will deliver a sensible and integrated set of products over a period of time.

Although developments are underway in many of the areas described, nothing in this article should be read as implying a commitment to deliver any particular product or facility by any particular date. One important lesson learnt during the early stages of the programme is the incredible speed with which the industry is moving at the workstation and departmental systems end. The programme is subject to constant monitoring and redirection to ensure that it remains relevant to customer needs. In addition, development timescales are influenced by customer requirements and priorities. However the GIN programme provides an overall framework within which a number of different product strategies will evolve, and thus indicates a statement of intent which customers can discuss with their local ICL representatives.

2 Major Customer Requirements

The GIN programme aims to provide a response to a number of important customer concerns. Its target is to produce a coherent architecture which meets the needs of large corporate customers during the 1990s, and to deliver a sensible and integrated set of hardware, software and networking infrastructure products which conform to this architecture. It is built around a number of themes, each of which is derived from clear customer requirements.

2.1 Single Terminal Access

The largest Government Departments tend to have many thousands of users, often carrying out similar tasks in a wide variety of offices. Each user has his or her own areas of responsibility, yet it is important for the public perception of the Department that all use the same applications, apply the same rules (particularly if they are in contact with the public) and are subject to common constraints on what they may and may not do.

These staff tend to be organised into workgroups, containing up to (typically) 60 people, working in the same office, and sharing some common purpose, work objectives, and often some local data. Members of a workgroup need to access services provided by servers either within the office, or at some remote location.

Almost all these staff now require access to IT facilities, usually via a terminal or workstation on their desk. The great majority of them carry out what are essentially clerical rôles – form-filling and data retrieval functions in response to mail or telephone prompts. The range and sophistication of the facilities that they wish to use has increased considerably in the last few years and shows no signs of abating. Very few of these staff have space for more than one terminal, nor could their organisation afford for them to have more than one. Thus this single terminal must provide access to a very wide range of facilities.

It is often necessary to be able to access two or more applications concurrently, with easy switching between them and, if appropriate, the ability to view each in a separate "window".

2.2 Security

The growing use of IT facilities throughout large organisations places an increased emphasis on the integrity of the corporate data owned by that organisation. In addition, modern concerns about data privacy in many countries mean that it must be possible to guarantee to the public that information they divulge (especially to Government Departments) will only be used for the purposes for which it was intended. Computer systems used by Government Departments hold large amounts of data about people – taxpayers, vehicle owners, benefit claimants, convicted criminals and so on. Government looks to its suppliers to provide adequate mechanisms to protect this data.

Because of the sheer size of some internal Departmental sub-networks, Government is concerned about the integrity and continued availability of the network itself. It should not be possible for semi-trained staff in district offices to interfere with the configuration information, applications or other aspects of the computer systems they use and thus reduce the availability of these systems to other users. Indeed, there is a growing trend within Government for IT Departments to prepare "service level agreements" with their user departments, and so any lack of availability from such a cause could have financial penalties for the IT Department.

These concerns mean that it is important to provide suitable security facilities which can protect the integrity and confidentiality of the organisation's data, both against external bodies such as "hackers" and against the organisation's own staff who have no need or right to access that data. The end-user must only be permitted to use those applications and services made available to him by his organisation. He must be required to identify himself (and prove his identify) before he can do anything, even access PC applications. And the data used by these applications should only be accessible to the users or applications which need to access it.

Defence organisations have requirements which go a long way beyond this, and some of these were discussed in [Jones 1989]. The GIN programme is concerned primarily with security in commercial organisations, though some of these more stringent requirements will undoubtedly apply in due course.

2.3 Common Look and Feel

IT facilities are the tools used by today's clerical workers to carry out their job. They are not a *part* of that job, and hence should be as unobstrusive as possible. There should be a common "look and feel" to how they get into and out of the applications which meet their real needs, and where practical to the image presented by these applications themselves. It must be possible to use several applications concurrently, and to switch between them easily. These applications must be limited to those which are available to a user in the rôle in which he is logged on.

The route to a remote application or service can be very complex, especially in large networks with many servers and alternate communications paths. In addition, some existing application may impose an additional "logon" process, and it should be possible to hide this from the user once he has been adequately identified to the network. It must be possible to shield the user from these complex details, so that as far as he is concerned, service selection operates in a "seamless" way and he need not be aware whether the server is within his local office or in a remote DP centre.

Most users spend all their time in their "home" offices, and only ever use their IT facilities from their own desks. However some users in some organisations need to be able to "log on" from any workstation, wherever they happen to be at the time. Such a login must give the user access to his permitted list of applications and services, and to any personal data maintained by these applications.

2.4 Systems Management

As indicated above, there is a significant move within Government to the provision of UNIX-based systems within very many offices. UNIX is notoriously complex and difficult to manage, administer and suport, and many UNIX users (particularly in the academic environment in which UNIX originated) would think nothing of dedicating highly skilled staff to this purpose.

Though this may be practical with a small number of free-standing UNIX systems, it is out of the question for large organisations with large numbers of similar UNIX systems. It shouldn't be necessary to employ a UNIX "guru" with each of these systems. Most customers cannot afford them, and would be wary of the implications for the integrity of these systems if they could. In large Government Departments, tens or even hundreds of these UNIX systems will be identical in all important respects, having similar configurations and running identical software packages. This should make it possible to simplify the support requirements by centralising support staff in one support centre. There is a need for a range of applications which make it possible to manage these replicated systems effectively. For example, it must be possible for the support staff to monitor the well-being of the distributed systems and their applications, to diagnose and (if possible) fix them when they have problems, to gather information from them relating to their performance, and to distribute new software and configuration data to them.

2.5 Using Modern Applications

Users need to be able to use any application or service, wherever it may be located in their network, from just a single workstation. But the rapid spread of computing facilities, and the technological explosion of the last few years, may mean that it is no easy task to define what an application is and where it is located. At its simplest, an application consists of three parts:

- A Human-Computer Interface (HCI) component, concerned with the way in which the user sees the application and how he drives it.
- An Application component, which carries out the functions required to support the user's intentions.
- A database component, which contains the data need to support the application, and manages this database if it is also accessed by other users.

Traditionally, large mainframe applications located all three of these components in the mainframe, and PCs located them all in the workstation. Now, PC networks permit some application and server functions to migrate to other boxes on the LAN. Relational databases such as INGRES enable the HCI and Application components to be located in the local server, accessed from dumb terminals, with the database located in one or more remote mainframes.

Figure 1 illustrates some of these application structures, and shows the logical end-point of this evolution, in which all three components are located in different systems. It must be possible to support any sensible combination of locations for the three components, and there should be no unnatural boundaries limiting where any particular function may be located. In particular, it should be possible to separate the HCI component from the application, and locate it close to the user.

It is interesting to note that Eprit-funded projects such as RICHE and RIBA [see Drahota 1990] are concentrating on the kinds of modern application architectures made possible by the emergence of Open systems and OSI protocols. The GIN programme must provide a platform able to support the needs of these new architectures.

2.6 Open Systems and Standards

Increasingly, organisations are demanding that their IT systems are based upon Open Standards. They do not always specify exactly what they mean by this, though the underlying impetus is to be able to move towards a measure of application portability from one vendor's hardware to another's.

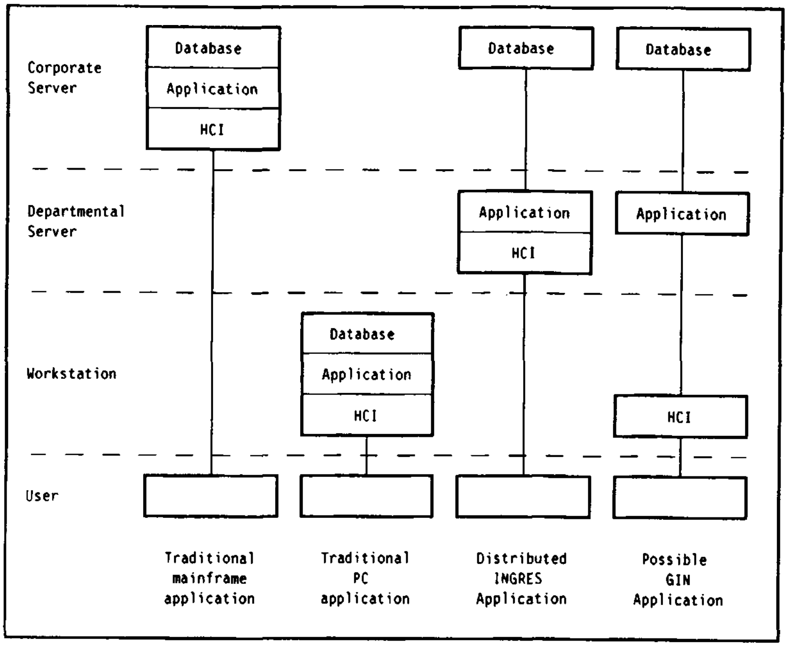


Fig. 1 Application Architectures.

Government has taken a lead by defining under the aegis of the GOSIP programme (Government OSI Profiles) a set of OSI standards which are increasingly becoming mandatory in large procurements. In addition, Departments demand conformance to standards defined by bodies such as X/Open. And, recently, the UK Government GOSIP Terminal Requirement Taskforce has accepted the need for a terminal architecture which underpins the move towards open systems in the use of terminals to access IT systems.

3 Meeting the GIN Requirements

All in all, that is quite a large variety of requirements. It will require solutions very different from those of the 1980s. One fundamental difference is that these solutions cannot be confined to one specific component or product. The wide and free-ranging nature of these requirements means that ICL must provide solutions which are integrated across products at a number of different levels – at the corporate mainframe, the departmental server and the desk-top workstation. It is a major challenge.

The GIN programme responds to this challenge. It defines an architectural framework within which it is possible to meet all of these stated requirements, and yet which is flexible and open-ended enough to respond quickly to a rapidly changing market place and to the pace of technological change.

igure 2 illustrates the overall IT network structure around which the rchitecture is based.

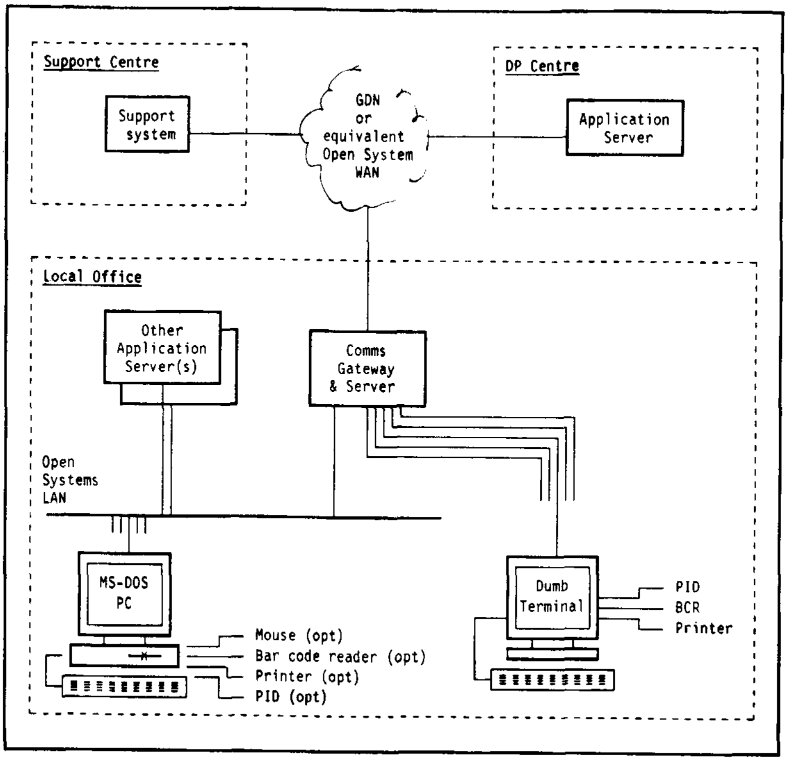


Fig. 2 Components of the GIN Architecture.

- It identifies a number of local offices, each containing a number of workstations which may be either PCs or dumb terminals. Where PCs are used, they are connected using Local Area Network (LAN) cabling standards.
- Each office contains one or more UNIX servers. One of these acts as a communications gateway to an external network, and "owns" the PCs and terminals. There may be other servers offering services such as mail, applications, printing and so on, though given the power of modern UNIX servers it is becoming fairly common to offer all of these services from a single system.
- The UNIX servers conform to the major Open Systems standards such as POSIX and the X/Open Portability Guide (XPG).
- A number of "DP centres" may exist elsewhere in the organisation. These contain the customer's existing corporate mainframes. Increasingly these will be augmented by UNIX servers, either in the same centres or distributed to some local offices. They may also include

services provided by Information Centres as these develop in response to the evolving needs of end-users for advice on new applications.

- The organisation also contains a system support centre which has overall responsibility for managing these servers and workstations.
- The various offices and centres are linked together by a Wide Area Network (WAN) which in the case of many Government Departments will be the Government Data Network, or GDN. This network has the benefit to the customer that it enables any office to access a server in any other office around the network, and in particular enables the Department to switch a service from one server to another in a different location in an emergency – so long as such a move has been planned for and any necessary data is held off-site.

Much of this architecture is by now fairly conventional, and will not be discussed further. But there are some areas – mainly within the local office itself – where the GIN programme requires a number of new technical directions, and the remainder of this Section concentrates on these issues.

3.1 Workstations

The end-user's workstation provides access to a wide variety of services and applications. Most existing UNIX systems are accessed by "dumb" terminals supporting protocol such as VT220.*

However, the GIN programme focuses on the demand for PC functionality from end-users, and the implications of this demand. These "industry-standard" PCs run MS-DOS*, and a multi-tasking graphical windows-based interface can be provided to the user by products such as Microsoft Windows.* Increasing numbers of the common PC applications are now available in versions which run under windowing environments.

In addition to being able to run standard PC applications, these PC terminals can also run protocol emulators for protocols such as VT220 or X/Windows. This enables their users to access applications located on the local UNIX server. However, much more significantly, the PCs can also run emulators for proprietary mainframe protocols such as ICL's ICAB-05. This means that the same terminal can be used to access a variety of existing mainframe or UNIX services, without the need to make any changes to the applications themselves. This has considerable implications for proponents of newer, more "open" protocols such as the OSI Virtual Terminal Protocol (VTP). It may often be more cost-effective for a customer to implement an existing protocol in a new terminal, than to go to the trouble and expense of adapting an old application to use a new protocol.

Microsoft Windows provides multiple "windows" enabling several such protocol emulators to run simultaneously, thus enabling a user to access many mainframe or local applications concurrently with easy switching between them. The only limitations on the number of sessions arise from the power and memory size of the PC itself. Microsoft Windows* also provides a modern graphical-based HCI which can be exploited by the protocol emulators to provide a rather better interface to the user than may have been possible with the original terminal protocol.

Although MS-DOS-based PCs are expected to satisfy the requirements of many customers, there is expected to be an increasing demand for UNIX workstations, especially for uses such as software development where sophisticated Computer Aided Software Engineering (CASE) packages are becoming available.

Let us now look at how the rest of the GIN architecture supports the use of PCs in this flexible way.

3.2 Imposing Network Security

The GIN programme supports a number of security features.

3.2.1 User Authentication A mandatory "login" process operates on both dumb and PC terminals. This process goes beyond the facilities provided as standard in UNIX.

Two systems components are involved. The first is a "User Sponsor". For dumb terminals this runs in the server to which the terminal is connected. For PCs, it is located in the PC itself and runs under Microsoft Windows. It is loaded whenever the PC is switched on or reinitialised.

The User Sponsor does not permit any applications or services to be accessed until the user has undergone an authentication process. It requests the user's ID and other authentication data such as a badge or password, and passes these to "Person Server" located in a UNIX server for verification. The Person Server contains a database which records all the members of the workgroup and information such as their User IDs, encrypted passwords, badge contents (if appropriate), and any limitations of users to particular applications, workstations, times of day, and so on.

The information supplied to the User Sponsor is checked against this database. If it is valid, the server constructs a packet of information about the user which specifies his identity and other information which is of use to applications. In due course, as the architecture develops, this packet will evolve into a Privilege Attribute Certificate (PAC) as described in [Parker 1990], and at that stage it will be possible to allow a user to log in at a system other than his own.

The User Sponsor can be configured such that repeated login failures cause either the user identity, or the workstation, or both, to be disabled and a security Alert to be generated. In addition, a user can be forcibly logged out if his terminal is inactive for too long.

3.2.2 Use of Badge Readers The customer can enhance this authentication process to include the use of physical objects such as badges or smart cards. If he does, a number of additional security-related features come into play, including the ability to log the user out or suspend all his sessions if the badge is removed. The main user of the badge contents is the authentication process itself. However, some existing applications also wish to use the badge contents, and thus it is necessary to devise enhancements to the VT220 and other appropriate protocols to make this information available to them.

3.2.3 Access Rights and Access Control Once a user has successfully logged in, the User Sponsor displays a menu listing those applications and services which he is permitted to use. The user can choose any of the available services, and the User Sponsor makes a connection to it, using a "script" appropriate to the application. This script hides from the user the details involved in making the connection, and thus gives a seamless service selection interface.

The menu presented to the user is in fact associated with a "rôle". The organisation can define any number of rôles, and associate each with a set of applications and services which a user acting in that rôle may be permitted to use. These may overlap, and will usually be chosen to give a sensible set of the applications which a user may need to access in a single workstation session.

When a user is introduced into the system, the local administrator defines the rôles which he or she may use. If there is more than one, then one of the rôles must be specified by the user at login time.

The user is not permitted to access any service or application which is not available at the current rôle. And because the user's access right is to a rôle rather than to a particular application, it is easy to withdraw the right to access a particular service when the user is no longer able to act in that rôle.

3.2.4 Using Multiple Concurrent Sessions The user can run a number of different applications concurrently (so long as they are available in the current rôle). Each operates in a discrete "window", the position and size of which can be varied on the screen by the user. When a window is opened, control is passed to the User Sponsor which displays the menu for the rôle. The user cannot avoid or bypass this menu. And whenever he exits from an application, the window and any communication connections it may have established must be closed.

3.2.5 Discless PCs PCs with diskette drives offer an opportunity for users to introduce rogue software or viruses into the network, or to take software or data away for private use. PCs with fixed discs may also be difficult to "manage" for reasons which are discussed below.

Thus the GIN programme supports the use of discless and "floppyless" PCs. As well as offering increased security, these can be cheaper, quieter, and more reliable than PCs with discs. However they do impose a number of requirements on the server.

- The server must take responsibility for "downloading" the operating system and ancillary software into the PC when it is switched on or reloaded.
- The workstation software, as well as any files generated or used by applications running on the workstation, must be held elsewhere in the local network (for example within the UNIX server's own filestore). Amongst its other benefits, this enables these files to be subject to the filestore access facilities of UNIX. For example they could be restricted so that they were only accessible to their owning user.
- The remote filestore facility must map names and disc drives used by MS-DOS applications onto appropriate parts of the UNIX filestore. Three different kinds of PC files are required:
  - The Operating Systems and other control software which is the same on every workstation. It is not necessary or desirable to hold these more than once in the UNIX server.
  - Files, used by these products, which must be tailored in some way, for example because they define the characteristics of a particular workstation (such as its identity).
  - Files which belong to the user himself. These may include data files used by applications (such as word processing documents), or in some circumstances applications which are only used by single users.
- The server can provide local administrative facilities, such as incremental archiving, which are difficult to organise (and hence often omitted) if files are held on individual PCs scattered around the office.

The use of a file server to hold the PC filestore offers increased flexibility in office accommodation as there are no constraints on moving end-users from one desk (and hence workstation) to another.

3.3 Communications Strategies

GIN systems will require interworking between the following end points.

- A workstation and an application in a server on the same LAN
- A workstation and a remote application via the server
- The server and a remote management centre

Government is committed to the use of OSI protocols. The Central Computer and Telecommunications Administration (CCTA) chairs a number of GOSIP committees, attended by representatives of Government Departments and the major suppliers, which define sensible profiles, or subsets, of the most useful protocols. The GIN programme provides a set of common platforms on which these profiles can be introduced in a co-ordinated way. GIN supports transport-level profiles which underpin each of the communication modes listed above. These include WAN profiles for use across an X.25 network, and both connectionless and connection-oriented OSI LAN profiles.

In addition the following GOSIP Application-level profiles are supported.

- File Transfer, Access and Management (FTAM) is supported in the server. It enables the transfer of files or parts of files into and out of the local office, for example by a support centre.
- Message handling System (MHS) is supported in the server, both as part of an integrated office support environment (OFFICEPOWER) and as free-standing service and set of interfaces which enable other applications to use messaging facilities. In particular this can provide a "user agent" capability so that, for example, the user is informed as soon as he logs in if there is mail waiting for him.
- Virtual Terminal Protocol (VTP) is the projected OSI replacement for proprietary terminal access protocols such as ICL's ICAB-05. VTP emulation can be provided both in the server, for use with dumb terminals, and in the PC. However the widespread adoption of VTP will entail changes to host applications and their infrastructure which probably cannot be justified given the ability to support a number of existing proprietary terminal protocols simultaneously from one PC terminal.

The server can also support non-OSI protocols, such as File Transfer Facility (FTF) or Interactive Video (ICAB-05) which are required by existing host applications.

PC workstations are connected to the server via LANs, and applications can use both OSI and TCP/IP transport protocols. The workstations can act as terminals accessing ICL, IBM and other mainframe services. Terminal emulators should, whenever possible run on the workstation itself, rather than on the server. This has a number of benefits.

- The workstation itself, rather than the server, responds to each keystroke typed by the user. This provides a faster and more effective response, similar to that which can be achieved when using conventional PC-based applications such as WordPerfect.
- The terminal emulator is responsible for only one terminal, and hence can be simple and easy to support.
- Terminal emulators can exploit modern HCI facilities available at the workstation, and present a high-quality interface to existing mainframe applications.
- Character traffic is kept off the LAN, so that LAN loadings are reduced and the response time to the user can be improved.
- A large load is taken off the server. This enables the use of less powerful servers than would otherwise be required for the same terminal connectivity. Alternatively, it leaves a greater amount of the server power available for handling local applications.

In addition to these facilities for accessing mainframe applications, PC workstations can communicate with applications in the local UNIX server using de facto or OSI LAN standards.

Servers communicate with remote services across the WAN, and with both workstations and other servers across the LAN. Again, OSI transport profiles are used, though TCP/IP protocols are also available for use by applications which have not yet been reworked to use OSI protocols. In some situations the server acts merely as a Transport-level Relay, providing a connection route from a workstation to an appropriate remote service.

The flexibility which is provided by the use of X.25-based WANs makes it reasonably easy to redirect connections to a particular service should it be necessary to migrate that service to a different server or location. This is particularly useful when preparing "disaster standby" contingency plans.

3.4 Systems Management

3.4.1 The GIN Approach to Systems Management The term Systems Management is used to cover the tasks of monitoring, updating and configuring a number of discrete (but often similar) end-systems from a single management or support centre. Some of the major requirements for systems management of departmental systems arise from Central Government Departments, and their concerns were discussed in Section 2.

The ability to carry out systems management tasks implies two needs:

- a set of "management" applications, for use by staff at the support centre.
- a set of facilities on the distributed servers in local offices which communicate with these management applications and thus permit the local office systems to be "managed".

The systems management architecture is designed to permit the management of a wide variety of different types of objects, not just the UNIX servers and their workstations which we have discussed above. The facilities which any managed object may support will vary depending on the nature of that object, its intelligence and capabilities, and availability of local human support and other factors. Here we are concerned with servers which have a good deal of processor intelligence but little human operations capability nearby.

It is important to differentiate between the characteristics of a systems management approach, and the use of remote operation or administration techniques which are increasingly common in networks of UNIX systems. In the latter, operations or administration staff handle the specific needs of the management end-systems individually, without being co-located with those end-systems. Systems Management, on the other hand, is required where there are too many end-systems, or too few support staff, to give each end-system individual care so long as it is behaving normally. The day-to-day administration and management functions must be automated so that only exception conditions are brought to the attention of the support staff. To a great extent it relies upon the end-systems having a high degree of uniformity in their hardware and software configurations, but that is exactly the type of situation which the GIN programme is intended to address.

3.4.2 Major Systems Management Functions Managing many distributed UNIX systems from a support centre requires a number of features not commonly found in UNIX. These include the following:-

- Managed systems must be able to report their status. This information should be displayed at the support centre in an easy-to-understand form.
- Statistical information generated on the managed systems must be fed back to the support centre for analysis or display.
- Problems occurring on a managed system must be reported back to the support centre so that they can be recorded in a problem database (preferably automatically). Support staff should be able to track the resolution of these problems. Remote diagnostic facilities are necessary so that skilled support staff do not have to travel to remote locations to investigate the problem
- Software distribution facilities are required so that the management centre can send new software releases, bug-fixes or other types of product upgrade to many remote sites, and to ensure that these are installed on every affected system in a simple and coherent way.
- The task of setting up and managing the configuration data and operational parameters in a large network can be extremely complex, especially if the network is constantly changing. Configuration generation facilities are required so that this task can be made manageable, and to allow co-ordinated changes to be made to many end-systems.

3.4.3 The Management Centre A number of management applications are required in the support centre to carry out these tasks. These are built around a common INGRES relational database. As well as providing a coherent view of the network and its components to all of these applications, this will enable customers or service providers to tailor the management applications, the data that is stored and the ways in which it is accessed, to their own unique requirements.

The management centre will be in constant communication with a large number of managed systems, each of which can transmit status or error information back to the management system on a regular basis. It must be powerful enough to handle the number of concurrent network connections, and the incoming message traffic.

At times the management centre may transfer data in bulk to many managed systems. While the load on the management centre may be reduced by the use of "store" and "forward" and similar techniques, a high communications capacity will be required at the management centre to meet peak loads.

3.4.4 The Managed System The managed system will not be subject to some of the more onerous performance requirements of the management system, but will need to be tuned so as not to overload the management centre. It must support a range of facilities which interact with the management centre applications, and at the same time work with and exploit those features of UNIX and of the server architecture which make remote management viable. These facilities include the following.

- **A messaging system** based upon ICL's Community Alert Management (CAM) facilities. This may be used by any application which wishes to pass a message back to the support centre (or to another process within the managed system). A *filter* which is part of the CAM is used to decide what to do about any particular Alert. Options include passing it to the management centre, logging it in a file or journal, passing it to another process, or ignoring it.
- **A bulk file transfer responder** is used on the managed system to allow the management centre to send it new software releases, configuration changes and so on, and to extract statistical data.
- **Status monitoring processes**, which can be invoked periodically or polled by the support centre to report on the status of the managed system as a whole or of some part of it. The status information returned can be a simple "I'm alive" message, or it can be more sophisticated, such as a record of the current number of logged-in users.
- **A statistics collector**. Some applications such as the X-25 and OSLAN handlers can generate statistical information on demand. In addition, Alerts which are written to a file, standard UNIX log files, and a host of other sources can generate what are loosely termed "statistics". This information enables the support centre to monitor the behaviour of the network, to spot any potential bottlenecks before they become serious, and to plan any necessary extensions to the network based upon accurate information.

  An application running within the server can take information from these various sources and return it to the management centre for analysis or display. Each source of statistics provides what is called a "sponsor" to interface between the supply of information and the format in which it is required at the support centre.
- **A workstation manager**, which is necessary when discless PCs are used. This is responsible for "bootloading" the PCs when they first switch on. Subsequently it establishes a management connection of each connected PC; this connection is used to check the status of the PC, and to enable the PC to pass Alerts and statistical information back to the server.

- **A software distribution agent**, which takes responsibility for the installation of new software products, upgrades to existing products, and new configuration data. UNIX SVR4.0 supports a concept known as "product packaging", which permits application developers to package their products and distribute them in a uniform way, similar to the "shrink-wrapped" style common to PC applications. The package includes a description of the product, its filestore structure, and how it is to be installed on the UNIX system. UNIX allows packages to be installed or deleted, and the facilities are designed primarily to be used by a local system administrator on the system on which the product is to be used. GIN expands on this in three ways. Firstly, it enables packages to be transmitted over a network into a buffer area on the managed system, and then installed by commands from the management centre. Secondly, it allows the distribution of product upgrades, which take a currently-installed product from one version to another (later) version. This reduces the amount of data which must be transmitted to the managed systems; unnecessary network traffic is expensive. It also permits the use the upgrade actions on the managed system if, for example, an upgrade from one version of a product to another means that it is necessary to reformat data owned by the product. Thirdly, before an upgrade is applied, the distribution agent constructs a "regression package" which reverses the effect of the upgrade. This is kept available in case it is needed.

  Once a package or a package upgrade has been delivered to an end-system, it is necessary to install and "activate" it. The software distribution agent understands the package structure and can map the product onto the UNIX filestore structure. It can invoke package-supplied "upgrade hooks".

  The distribution application in the management centre can distribute the same product to many tens or hundreds of similar end-systems concurrently, and monitors the progress of the installation. It is possible to group together end-systems with common properties, such as that they all run the same application; the support centre can then force the distribution of a product to all members of a given group.

  In most cases the amount of bulk data involved in a software distribution process will be sufficiently small to enable the customer's network to be used for its dissemination. However there are some cases in which the sheer bulk of new software, or some other reason, means that it may be preferable to employ an alternative upgrade mechanism, such as a streamer tape.

- **Configuration Generation Hooks** enable the management centre to make co-ordinated changes to the configuration information and tables used in many end-systems. The management centre maintains a single database which defines the network topology, the managed units, their interconnections and their characteristics, including installed software versions. It can thus identify all the objects affected by a particular configuration change. Thus when some aspect of the configuration changes, a management application generates a number of files, each of which gives a view of the changed network looking out from one of the systems affected by the change. These files are transmitted to the appropriate end systems, using the software distribution mechanisms, and at a specified time each is used to generate and install all the configuration data required by affected components within that end system. Each managed component needs to provide a configuration hook. These include the X.25 and OSLAN handlers, the Message Handling System (MHS), and the CAM subsystem.

3.4.5 Fault Management Fault management covers those aspects of system management concerned with the detection, recording, diagnosis and rectification of faults occurring in remote locations.

Within a managed system, components which detect faults indicate the fact by generating a CAM Alert. This is handled by the CAM filter; usually the Alert will be passed back to the support centre where it will be recorded in a problem database or "help desk" so that support staff can monitor and chase it.

Support centre staff can access the managed systems across the network to diagnose faults. A number of diagnostic tools are provided. Those which are used in a particular case will depend upon the nature of the fault, and whether it is sufficiently serious to prevent the end system from operating. They include standard UNIX tools as well as some specifically developed for GIN. These offer facilities for dump analysis, displaying audit trail files and so on. UNIX systems traditionally maintain a number of audit files, and adopt conventions such as mailing messages to the system administrator following any unusual incident. In addition, much statistical information related to performance, device utilisation and error events is gathered. All of this information can be accessed by the support person; much of it can also be returned to the support centre periodically.

Hardware faults will usually require a visit by an engineer to replace or service a faulty part. Software faults may be fixed by patches which would normally also need to be applied to other similar systems. These are normally issued in the form of "upgrade packages" to the affected product, and are distributed and installed using the software distribution facilities described above.

3.4.6 Performance Management Any organisation responsible for managing a large network will be concerned about the performance of the components of that network, and the response-times experienced by the users of the network. In the past, this information would be required to handle complaints from users about poor response times or low availability.

Now, more and more Government computer departments are entering into Service Level Agreements with their customers, and this information is required to support and confirm their ability to meet these agreements.

In addition, it is necessary for ICL to be able to demonstrate that systems proposed to customers are capable of meeting the customer's performance requirements.

GIN systems provide facilities to log response-time information. This information is collected both as near as possible to the user, so as to be most precise, and at points such as the interface with a WAN or LAN to enable problems affecting performance to be identified. It is returned to the support centre via the standard statistics collection mechanisms.

Acknowledgements

The GIN programme affects a large number of people in both the marketing and development organisations of ICL. Particular mention must be made of David Teague and Mike Coombs, Director and Programmes Manager respectively of ICL's Central Government and Defence Business Unit, who understood the need for a strategic programme of this type and encouraged the process which led to its definition and adoption as a corporate programme. Mark Davies, Dave Weitzel, Bernard Pierce and Phil Bates provided invaluable advice on the needs of their specific customers.

Once these requirements are articulated, staff in Product Operations must respond to them. John Scott, Bill Marcham, Peter Dawe and John Jones of the Major Projects Central Design Team will ensure that the delivered products form a consistent and coherent whole. Others, including Belinda Fairthorne, Andrew Hutt and Malcolm Butler, commented on early drafts of the requirements specifications, helped the author understand the industry directions which constrain any solutions, and defined the developments necessary to meet them.

MS-DOS is a trademark of the IBM Corporation;

UNIX is a registered trademark of AT&T in the USA and other countries;

X/Open is a trademark of the X/Open Company Ltd;

Windows in a trademark of Microsoft Inc.

References

1 JONES R.W., Security Classes and Access Rights in a Distributed System. ICL Technical Journal, 1989, 6(4), p 694

2 DRAHOTA A, [title tbs]. ICL Technical Journal, 1990 [Vol & number tbs] RICHE, Réseau d'Information Hospitalier Européen. ICL Technical J. Vol 7, no 2, p. 298.

3 PARKER T.A. Standards for Secure Interfaces to Distributed Applications ICL Technical Journal, 1990 7(1) p 141.

A1 Glossary

CAM	Community Alert Management
CCTA	Central Computer and Telecommunications Administration
CG&D	ICL Central Government & Defence Business Unit
FTAM	File Transfer and Management
FTF	File Transfer Facility; proprietary ICL file transfer protocol.
GDN	Government Data Network
GIN	Government Infrastructure for the Nineties
GOSIP	Government OSI Profiles; a CCTA-led subset of the major OSI protocol stacks to evolve a set of standards which are demanded from IT suppliers in Open Tender procurements.
HCI	Human-Computer Interface; it is a description of all aspects of the interaction between a human being and the computer systems that he/she uses.
ICAB-05	ICL proprietary protocol for mainframe communication.
MHS	Message Handling System
OSLAN	Open Systems LAN; ICL's implementation of ISO 8802.3 LAN protocols.
PAC	Privilege Attribute Certificate; an encrypted record of a user's User ID and other security attributes produced as a result of a successful authentication process.
PID	Personal Identification Device; a device such as a credit card which contains personal identification information usable by a computer system.
POSIX	Portable Operating System for Computer Environments; an IEEE standard for a set of UNIX-like interfaces.
SVR4.0	System V Release 4.0 (of UNIX)
TCP/IP	Transport Control Protocol/Internet Protocol; group of networking protocols included in UNIX SVR4.0.
VT220	Screen-handling protocol common on UNIX systems.
VTP	Virtual Terminal Protocol; an ISO protocol, adopted by GOSIP, aimed at standardising communication between applications and workstations.
X.25	CCITT recommendation for packet-switched WAN services.
XPG	X/Open Portability Guide

Book Review

Realistic Compiler Generation, by Peter Lee, The MIT Press, Cambridge, Massachusetts, USA, 1989, xviii + 246 pages, ISBN 0–262–12141–7.

This book outlines the theory and practice underlying a compiler generator. When given a definition of the syntax and meaning ("semantics") of a programming language, this compiler generator will create a compiler for the language. Its capabilities go well beyond those of a parser generator such as YACC, because it automates not merely the construction of the parser but also the construction of the code generator. Of course the user must still input the syntax and meaning in forms acceptable to the compiler generator, but these particular forms are written in a high level language and have other uses besides compiler generation. This review discusses these other uses briefly, before returning to the subject of this book.

Just as BNF is now the standard approach to describing programming language syntax, so denotational semantics is becoming the standard approach to describing meaning. Both these approaches take several different syntactic forms; the concepts, rather than the syntactic forms, currently constitute the standards. Moreover, denotational semantics remains a subject of active investigation, especially in relation to concurrency and type theories. Nonetheless, denotational semantics is well enough established to be able to cover a very large area of programming, including Algol 68 and the better thought out parts of Ada, for instance.

Among the uses generally envisaged for denotational semantics are the following:

- allowing the concepts underlying languages to be specified, so that the languages may be better designed, described and understood;
- allowing laws about programs to be formally justified, so that reasoning about programs can itself be validly formalised;
- allowing implementations of languages to be proved correct relative to the semantics;
- allowing implementations of languages to be generated from the semantics, thereby providing correctness by construction.

Many of the techniques needed in these uses have long been available as "laboratory bench" prototypes. Some have been adopted for "pilot plant" experiments. The current concern with safety critical systems (and other high integrity systems) will certainly lead to their greater exploitation: a high integrity kernel requires both a justification that the program meets its specification and a guarantee that the code generated for the program can be trusted.

The compiler generator described in the book under review uses denotational semantics to ascribe meanings to programming languages. In this respect it is similar to several other compiler generators. It differs from these others in that it is structured to encourage the use of more than one version of the semantics of a language. These versions of the semantics are written in essentially the applicative subset of the programming language ML; this is one of the best designed languages in current use and is itself a by-product of work on denotational semantics. Different versions of the semantics can offer different compromises between, on one hand, their reliance on general abstract concepts and, on the other hand, their closeness to the structures of an intended target machine for compilation. Of course, one of the versions of the semantics (presumably the most general and abstract) must be chosen as defining *the* meaning of the language, and the other versions must be shown to be equivalent to it. The book does not explain how to do this (though methods are known); yet the need to do it prevents a compiler generated by this system from being automatically correct by construction. However, the system offers the possibility of trading the simplicity of proofs of compiler correctness against the efficiency of compiled code on conventional target machines.

The examples given in the book indicate that adequately efficient compilers can be generated quickly by systems like this one, even running on a PC. However, this efficiency is achieved at the expense of the simplicity of the proofs that the compilers are correct. Only if the target machines were declarative would both simplicity and efficiency be attained.

Because the input to the compiler generator is written in ML the task of generating a compiler should be made more straightforward by using systems like this one. Hence such systems could well provide a preferred approach to creating compilers, though much of the immediate benefit would come from the use of ML rather than the particular facilities of the compiler generator. Moreover, where either the target machine is declarative or a suitable equivalence proof is performed, the compilers thus created can have their correctness guaranteed. This fact may be central to future high integrity systems.

The particular compiler generator described in this book is perhaps unlikely to be central to such systems, as at various points in the book the grasp of denotational semantics is rather shaky. For instance, the semantic functions are not purely applicative: the unique name generators and the treatment of recursive procedures depend on hidden state. Also, the different versions of the semantics are unlikely to be equivalent; they may all be refinements of a further version of the semantics, but the book neither gives that version nor suggests how to give it.

Other books offer introductions to denotational semantics which are both broader and deeper. However, the results of work on compiler generation are rarely described fully. People wanting introductions to both denotational semantics and compiler generation could well start with this book – but they should not stop with it.

R.E. Milne
STC Technology Ltd.,
Harlow,
Essex

Notes on Authors

G.P. Abraham

Graham Abraham joined ICL in 1967 after graduating from Manchester University with B.Sc. honours in Physics and Electronic Engineering. Since joining the Company in Manchester he has worked exclusively on mainframe developments, being involved in four projects to date.

In July 1981 he began the exploratory technical discussions with Fujitsu, which led to the first mainframe technology collaboration agreement between the two companies being signed later that year.

In June 1986, he took responsibility for the SX processor and tools group.

G. Allt

George joined ICL in 1968 with an HND in Computer Science from Oldham College of Technology. He initially worked for the Computer Engineering Service Organisation before moving to development in 1971. He has been involved in both 1900 and 2900 machines primarily in microcode development. On SX George was the design manager for the Picode development.

He is currently working as a senior system designer on future mainframe system development.

E. Babb

Ed Babb obtained qualifications in Electrical Engineering and Computer Science at Imperial College. He then researched adaptive pattern recognition systems at Cambridge University on secondment from Hawker Siddely Dynamics. From about 1971, working in ICL, he researched speech recognition and information retrieval. His work on CAFS covered storage structures, architectural and query languages. He is now studying the application of mathematical logic to business and is currently Manager of the Logic Language Project in the Systems Strategy Centre at Bracknell.

Ian Cole

Ian Cole has an Honours degree in Biology and Human Factors from Aston University, which included an industrial year at British Steel Corporation.

He went on to gain a PhD at Loughborough University, with research into the role of human memory in the storage and retrieval of information. In 1981, he joined the National Computing Centre as a consultant in office automation before being recruited by the Engineering Support Centre of ITT Europe in Harlow. Most of his time with ITT was spent working on the design of a large distributed office system. Ian moved to STC Technology Ltd. in 1984 in order to build up a capability for user-centred system development. Three years as the Technical Strategy manager of the User-centred Systems department has led to current responsibilities as manager of the Professional Community Support group and project manager of the Esprit Multiworks project.

Peter Crowther

Joined the Multimedia group as a Research Assistant in July 1989 after completing his degree. Prior to this he worked at Racal Imaging Systems Limited on a multimedia document processing system. His main areas of expertise are in database and user interface design and implementation.

Dr. Costas Daskalakis

Joined the Department of Computer Science in December 1985. When the MMIS project started he transferred to Electrical Engineering, where he is currently a lecturer in Signal Processing. His interests are in the area of signal/image processing algorithms and architectures and in particular image segmentation, texture identification and semantic representation of images. Dr. Daskalakis holds a number of UK and international patents and is the author and co-author of two books and over 20 papers in the areas mentioned above.

A. Drahota

A. Drahota joined ICL in 1977 after gaining a Graduate Diploma in Operational Research and Computer Science from the Polytechnic of the South Bank London. Specialising in systems performance and sizing, he has worked on a number of large systems projects, predominantly with UK Universities (Oxford, Edinburgh, Queen Mary College London, Nottingham etc). During the past 7 years he has held several technical management and project management positions, working with different Government Departments and with Universities. Currently, he is a member of the Technical Directorate in the Central Government and Defence Business unit, and also acts as the ICL technical manager for two ESPRIT II projects – RICHE and BANK'92.

J. R. Eaton

John joined ICL in 1965 after graduating from Sheffield University with a BEng in electronic engineering. Since joining ICL he has worked exclusively on mainframe developments being involved in both hardware and micro-code development. During his career he has worked on five mainframe developments and has been responsible for the design of two – Series 39 L80 and SX.

He is currently a senior design manager and is leading the investigation into future mainframe system strategy and development.

R. E. Epworth

Ricard Epworth is a Technical Strategy Manager in the Communications Systems Division of STC Technology Ltd. (STL) in Harlow. He received the B.Sc. degree in Electrical Engineering from the Univerisity of Manchester in 1966, then joined STL, where he worked on a variety of projects including radar signature recognition, intruder detection systems, an instrument landing system, And various optical links.

Since 1973 he has worked almost exclusively on various aspects of optical-fibre communication and sensor systems. He developed the concept of Modal Noise to explain the problems caused by the use of lasers in multimode fibre systems, and has worked on the development of a wide range of coherent optical fibre systems. More recently he has become fascinated by the HCI aspects of visual perception, and has spent part of his time investigating eye-movement controlled technology.

Christer Fernström

Christer Fernström is the Technical Director of the Eureka Software Factory project (ESF), a position he held since the start of the main phase of the project in 1987. Prior to this, he was part of the team which laid the technical foundations for ESF.

He works for Cap Gemini Innovation, CGI, which is the research section of the Cap Gemini group. Between 1985 and 1988 he was responsible for CGI's R&D efforts in software engineering and was project manager for the Esprit project PIMS, dealing with support for project management. He has been active in the area of software engineering since 1980. He holds a BSc in Electrical Engineering and a PhD in Computer Engineering, both from the University of Lund, Sweden.

Dr Richard Flavell

Richard Flavell is a Reader at The Management School, Imperial College. He holds a BSc in Chemical Engineering and a PhD in Management Science, both from the University of London. He has taught at Imperial for the past 15 years, specialising in finance and project appraisal. He has published on these topics in a variety of journals. He has also worked in a bank dealing room and is a recognised financial trader.

Much of his work involves the production of computer-based financial models. Several years ago, this stimualated the question of how to produce effective displays and especially using colour, for these models. Casting around, very little useful guidance was found and hence an interesting sideline was born. This work has been funded solely by industry over the past seven years and is becoming widely recognised as a leading source of advice.

Fiona Flower

Graduated from the University of Aberdeen in 1978 with MA Honours in Modern Languages (French and Spanish).

Joined ICL in September 1978 as a graduate recruit, and completed the graduate induction programme.

First assignment with ICL was with Local Government and Public Corporations (LGPC) in Bristol, as an account support executive. After a short secondment to Dataskil in Reading, she was tasked with providing all pre- and post-sales support for 7700 for LGPC customers in Wales and the West Country. In 1981 she transferred to major Accounts, and provided accounts support in office systems for Major customers such as House of Fraser and Dowty.

When Customer Service was formed, she joined the Small Systems support team in Bristol, providing support in DRS systems and applications to customers in the Wales and West Country area, and remained with Customer Service until May 1985, when she took 6 months' maternity leave.

Following maternity leave, she joined CPS as a technical author, working part time from home. In the 4 years since then, she has completed a wide variety of projects, producing documentation for a number of clients, both internal and external. Internal clients included Defence Region, Office Systems, CRS&S, Networks Region, Northern Telecom and CIS. Her current project is to produce two of three booklets for Andrew Hutt, to complement his Marketing to Design lectures:

"How to Identify a High Valued Solution" (describing a process for deriving product attributes from user needs).

"How to Deliver a Business Solution" (describing how to identify the manuals, man-machine interface and service to be delivered with a product).

D.C. Freeth

Dave Freeth graduated in Electrical Engineering from Kings College, London in 1963. He joined EMI at Hayes on the graduate apprenticeship scheme and was there introduced to computers through involvement in the development of Magnetic Thin Film Storage Systems. Moving to Elliott Automation in 1967 to work on the Elliott 4300 computer system, he soon found himself part of ICL based at Stevenage. There he became involved in studies into ICL's 'New Range' (later to become 2900 series) and was part of the 2960 development team. Re-located to West Gorton in 1975, he continued his involvement with mainframes participating in the development of 2956, 2966 and managed the teams responsible for the development of ICL's first multi-nodal system, the Super-dual 2988.

He then became involved with Series 39 and is currently Manager of the Firmware Systems Product Centre within the Future Products Development. He is a member of the IEE and a Chartered Engineer.

Carole Goble

Has been a lecturer in the Department of Computer Science at Manchester University since 1985. Prior to this she was a Research Associate in the Barclays Groups within the same department, specialising in medical databases. She has expertise in data modelling, databases, multimedia, hypertext and medical informatics. Her recent work has been in object-oriented and binary semantic data models, particularly applied to MMIS.

Olaf Hesse

Olaf Hesse is the Market Integration Manager of ESF. In this capacity he is responsible for ESF market relations and for relations to other software engineering initiatives. This also includes the application of ESF results in trial environments and their first use in practice. He is seconded to ESF Headquarters from STZ GmbH on behalf of the University of Dortmund.

While studying mathematics/computer science and economics at the Technical University and the Free University, both in Berlin, he worked part-time with IBM. Since graduating in 1979 he has introduced and applied software engineering technology as Head of the Data-processing Department of a bank and worked on the development of a software engineering environment as Vice-President of a consultancy company. As an independent consultant he has supported several application development projects and the foundation of a private bank.

K. Hughes

Kevin joined ICL in 1979 after graduating from Sheffield City Polytechnic with a BSc in Computing Science. He initially worked at Bracknell on emulation systems (CME and DME) for the 2900 range moving to West Gorton in 1982 to develop the CME* emulation system on Series 39 L35. Since then he worked as a system designer on distributed mainframes. He has a keen interest in system development routes and was a prime mover in the adoption of modern software techniques and tools for I/O controllers.

He is currently working as a system designer on future mainframe system.

Dr Andrew T. F. Hutt

Dr Hutt holds a BSc Mathematics from Durham University and PhD Computer Science from Southampton University and has worked for ICL since 1966.

At ICL between 1966 and 1973, he worked on the design of the Edinburgh Multi-Access system and of VME Operating System. In 1973, he was seconded to Southampton University where he developed an early Relational Database Management system. On his return to ICL he led a 25 man team which produced QueryMaster, Personal Data System and CAFS Relational Interface which are based on his research work. These products are used widely throughout the ICL customer base.

In 1984, Dr Hutt became ICL's HCI Strategy Manager responsible for coordinating all the HCI work in ICL. In this capacity he had 2 Alvey projects: User Skills and Task Match and the Alvey Lab. Project: both of which have produced valuable research results. He has also contributed to other collaborations namely, the Alvey Early Evaluation Project, and the Esprit HUFIT and Eurohelp projects. Dr Hutt has been responsible for the industrialisation of the results of this research work and has developed the "Marketing to Design" human factors based methodology which is routinely used throughout ICL.

In the period 1985 to 1989, Dr Hutt has also worked on the MAP MultiStar Project on distributed relational database systems which has procuded the only usable European system of this type.

Mark Ireton

Is a Research Associate in the Department of Electrical Engineering and has been involved with the MMIS project since its start in October 1988. Prior to this his research was in the area of low bit-rate speech compression, concentrating on simulation and real-time implementations.

Stephen Kay

Has been a lecturer in Information Systems within the Department of Computer Science at Manchester since January 1987. Before that he was a Research Assistant, then analyst/programmer in the College of Medicine at the University of Wales. Research interests are focussed on medical informatics encompassing modelling, database, knowledge base and user interface issues. He is also involved with international standards work related to data interchange of multimedia health care documents.

Michele Morris

Michele Morris began her career as a computer operator/trainee programmer for a wholesale foods distributor. During a ten year career break to look after her three sons, she obtained an Honours degree in Psychology from the

Open University. In 1987 Michele returned to full-time education, and gained an M.Sc. in Intelligent Systems from Plymouth Polytechnic. In October 1988 she joined STC Technology Ltd as a Senior Research Engineer where she has been involved in requirements capture and HCI design for several projects. She is a member of the British Psychological Society, The British Computer Society and the SSAISB.

Michael O'Docherty

Is a Research Associate in the Department of Electrical Engineering at Manchester University. He joined the Multimedia Group in October 1988 and since then has been involved with the design and implementation of techniques to allow the interaction of database management and medium-specific interpretation via data modelling and intelligent querying. Prior to this he has been involved in document recognition, especially the analysis of hand-written characters. His main interests are artificial intelligence, logic and object oriented programming and medium-independent data modelling.

C. Shaw

Chris Shaw, born in 1951, has a BSc in Electrical Engineering and an MSc in Digital Electronics from the University of Manchester (UMIST). He joined ICL in 1973 and has worked on the design and development of 2900 mainframes including 2960, 2950, 2966 and Series 39 Level 80. He is currently the development manager responsible for I/O and Store projects and for physical design for Corporate Servers Product Group based at ICL West Gorton.

Mick Smith

Dr Michael J. Smith completed his BSc in Computational Science at the University of Leeds in 1974, followed by research in Artificial Intelligence related to Adaptive Learner Modelling, and was awarded the degree of PhD in 1985. During the period 1978–1983 he was a Lecturer in Computing Science at Keele University, and returned to Leeds in 1983 as a research fellow in the Computer-Based Learning Unit where a major responsibility was to head the CEC Esprit Eurohelp team. In 1986 he joined ICL's Knowledge Engineering Business Centre, as it then was, to manage ICL's part of the Eurohelp project.

Colin Tattersall

Dr Colin Tattersall completed a BSc in Computational Science at Leeds University in 1986. He then joined the Computer Based Learning Unit on an ESRC studentship linked to ESPRIT project EUROHELP. Here he worked on knowledge representation and text generation for intelligent help systems, obtaining his PhD entitled 'Question Answering and Explanation in On-Line Help Systems: A knowledge-based approach' in June 1990.

Richard E. Thomas

Dick Thomas qualified as an electrical engineer and has very broad experience in the computer industry in both hardware and software, having worked on research, development, commissioning, support, marketing and customer applications in a vatiety of technical and managerial positions. These have been as diverse as a management information system for a large consumer goods company, real-time military control for NATO, and operating systems R&D.

Since 1985, he has concentrated on development processes and advanced environments for effective support of development processes. This work has included several joint industry studies and joint formulation of the software engineering strategy for ICL, leading to co-operative European work in establishing the ESF Project.

H. Vosper

Harry Vosper jointed ICL in 1962 as a student apprentice. He graduated from Queen's University Belfast with an honours degree in Electrical Engineering and Electronics.

He transferred to Development in 1970 working originally in Microsystems on thick film and integrated circuit design.

He became Hardware Quality Manager for the Estriel project in 1983 and subsequently for Future Products in Mainframe Systems.

He is now Process Development manager for Future Products Development.

Darren Van Laar

Darren Van Laar is a research assistant in Imperial College, London where he also lectures in Human-Computer Interaction (HCI). He is currently completing his PhD thesis on the use of colour with computers.

His first degree was in Psychology from Manchester Polytechnic in 1985 after which he studied computing and Human-computer ineraction for his MSc. at York University. Since that time he has been working at Imperial College on various projects concerned with HCI, colour, statistics, computing and psychology.

Vincent West

Vincent West graduated from Gonville and Caius College, Cambridge, with a B.A. in Mathematics in 1964 and joined the Central Electricity Generating Board as a programmer. He moved to ICL in 1966 to work on Fortran compilers, including the design of code optimisation methods. After a variety of software design work, including operating systems, he specialised in databases working on IDMS, relational databases and especially natural language interfaces. His current interests include logic programming and end-user interfaces to logic systems.

Peter Wiles

Peter Wiles obtained a B.Sc and M.Sc in Computer Science from the University of Manchester, and has worked for ICL since 1971. Until 1977 he was involved in the development of VME, eventually joining the 2900 OS strategy team OSTECH. In 1977 he joined the newly-formed EEC project in Luxembourg. While there he developed a number of Transaction Processing applications, including a set of MAC-type editors and job schedulers. In 1981 he returned to the UK and joined the Inland Revenue Project Team in Telford. He was responsible for a number of developments including the initial design studies for the CAFS-based National Tracing System which he described in the November 1985 Technical Journal and which has since proved of inestimable benefit to IR. From 1985 to 1987 he was responsible for co-ordinating IR's requirements for ICL product developments. In 1988 he moved to the Central Government & Defence Business Unit's Technical Directorate, with rsponsibility for defining the strategic requirements for Departmental Systems, and the GIN Programme has grown out of this work. He is currently requirements advisor to the GIN development teams, and is investigating the strategies and developments necessary to support and exploit the GIN architecture in 1991 and beyond.

Prof. Costas Xydeas

Has been active in digital signal processing research since 1974. In 1977 he joined the Department of Electronic and Electrical Engeneering at Loughborough University as a Research Fellow and was involved in low bit-rate speech coding research. In 1979 he was appointed Lecturer and in 1984 Senior Lecturer in Telecommunications at Loughborough, where he directed an MSc. course in Digital Communications and a research group in Speech and Image Coding. He currently holds a chair in Electrical Engineering at the University of Manchester where he is leading the Speech/Image Signal Processing and Multimedia groups.

ICL TECHNICAL JOURNAL

Guidance for Authors

1. CONTENT

The *ICL Technical Journal* has a large international circulation. It publishes papers of high standard having some relevance to ICL's business, aimed at the general technical community and in particular at ICL's users and customers. It is intended for readers who have an interest in the information technology field in general but who may not be informed on the aspect covered by a particular paper. To be acceptable, papers on more specialised aspects of design or applications must include some suitable introductory material or reference.

The Journal will usually not reprint papers already published, but this does not necessarily exclude papers presented at conferences. It is not necessary for the material to be entirely new or original. Papers will not reveal matter relating to unannounced products of any of the STC Group companies.

Letters to the Editor and reviews may also be published.

2. AUTHORS

Within the framework defined by §1 the Editor will be happy to consider a paper by any author or group of authors, whether or not an employee of a company in the STC Group. All papers are judged on their merit, irrespective of origin.

3. LENGTH

There is no fixed upper or lower limit, but a useful working range is 4000–6000 words; it may be difficult to accommodate a long paper in a particular issue. Authors should always keep brevity in mind but should not sacrifice necessary fullness of explanation to this.

4. ABSTRACTS

All papers should have an Abstract of not more than 200 words, suitable for the various abstracting journals to use without alteration. The Editor will arrange for each Abstract to be translated into French and German, for publication together with the English original.

5. PRESENTATION

5.1 Printed (typed) copy

Two copies of the manuscript, typed 1½/2 spaced on one side only of A4 paper, with right and left margins of at least 2.5 cms, and the pages numbered in sequence, should be sent to the Editor. Particular care should be taken to ensure that mathematical symbols and expressions, and any special characters such as Greek letters, are clear. Any detailed mathematical treatment should be put in an Appendix so that only essential results need be referred to in the text.

5.2 Diagrams

Line diagrams will if necessary be redrawn and professionally lettered for publication, so it is essential that they are clear. Axes of graphs should be labelled with the relevant variables and, where this is desirable, marked off with their values. All diagrams should have a caption and be numbered for reference in the text, and the text marked to show where each should be placed – e.g. "Figure 5 here". Authors should check that all diagrams are actually referred to in the text and that all diagrams referred to are supplied. Since diagrams are always separated from their text in the production process these should be presented each on a separate sheet and, most important, each sheet must carry the author's name and the title of the paper. The diagram captions and numbers should be listed on a separate sheet which also should give the author's name and the title of the paper.

5.3 Tables

As with diagrams, these should all be given captions and reference numbers; adequate row and column headings should be given, also the relevant units for all the quantities tabulated. Short tables can be given in the text but long tables are better submitted on separate sheets and these, as for diagrams, must carry the author's name and the title of the paper.

5.4 Photographs

Black-and-white photographs can be reproduced provided they are of good enough quality; they should be included only very sparingly. Colour reproduction involves an extra and expensive process and will be agreed to only exceptionally.

5.5 References
Authors are asked to use the Author/Date system, in which the author(s) and the date of the publication are given in the text, and all the references are listed in alphabetical order of author at the end.

e.g. in the text: "... further details are given in [Henderson 1986]"

with the corresponding entry in the reference list:

HENDERSON, P. Functional Programming, Formal Specification and Rapid Prototyping. IEEE Trans. on Software Engineering 1986, SE-12, 2, 241–250.

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, and only text references; references to works not quoted in the text should be listed under a heading such as "Bibliography" or "Further reading".

5.6 Style
A note is available from the Editor summarising the main points of style – punctuation, spelling, use of initials and acronyms etc. – preferred for Journal papers.

6. REFEREES
The Editor may refer papers to independent referees for comment. If the referee recommends revisions to the draft the author will be asked to make those revisions. Referees are anonymous. Minor editorial corrections, as for example to conform to the Journal's general style for spelling or notation, will be made by the Editor.

7. PROOFS, OFFPRINTS
Printed proofs are sent to authors for correction before publication. Authors receive 25 offprints of their papers, free of charge, and further copies can be purchased; an order form for copies is sent with the proofs.

8. COPYRIGHT
Copyright in papers published in the *ICL Technical Journal* rests with ICL unless specifically agreed otherwise before publication. Publications may be reproduced with the Editor's permission, which will normally be granted, and with due acknowledgement.