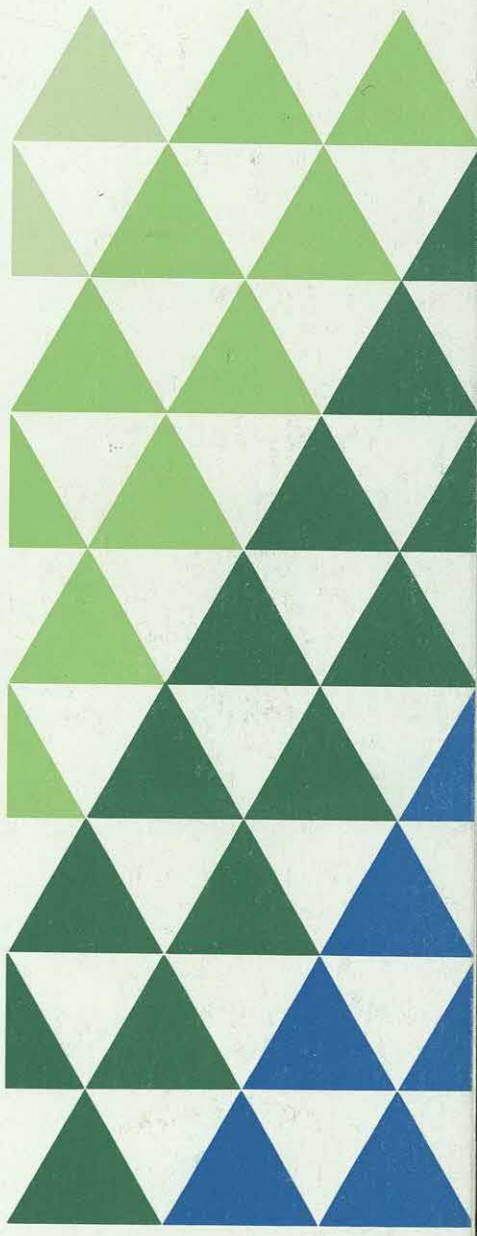
BUTLER COX FOUNDATION

Introducing Object Orientation



Management Summary

August 1992

CSC | Index

Introducing Object Orientation

Management Summary, August 1992

Published by CSC Index
12 Bloomsbury Square
London WC1A 2LL
England

Availability of reports

Sponsors of the Foundation receive copies of each report and management summary upon publication; additional copies and copies of earlier publications may be obtained by sponsors from CSC Index.

Photoset and printed in Great Britain by Flexiprint Ltd., Lancing, Sussex.

Management Summary

Introducing Object Orientation

Foundation Report 88, Introducing Object Orientation, was published in August 1992. Object orientation is a new and very different approach to the design and construction of information systems, which is now being applied to substantial commercial and technical applications. The report makes the case for introducing object orientation now. This document is designed to draw the attention of the systems director to the critical components of a strategy for introducing object orientation into a systems department, and eventually, to extend it into the whole field of corporate systems.

Our research has led us to the clear conclusion that Foundation sponsors should be starting now to introduce object orientation as their main approach to application development. For most sponsors, object orientation will be new, and very different from their existing methods. Its impact on the systems department will be profound, but its benefits for the business are potentially immense. (The benefits that have been achieved by one of the pioneering users of object orientation are described overleaf in Figure 1.) We believe that all systems directors should therefore formulate a strategy for introducing object orientation, and lead the drive to get it established.

In this management summary, we have deliberately omitted any detailed description of the characteristics or benefits of object orientation. These are clearly described in the main report. Instead, we concentrate on the main management steps in implementing the strategy. These are to communicate a clear vision of the importance and benefits of object orientation to all those involved with application development, to build a first successful application taking the new approach, and to extend the use of object orientation until it becomes the regular, mainline approach throughout the development department.

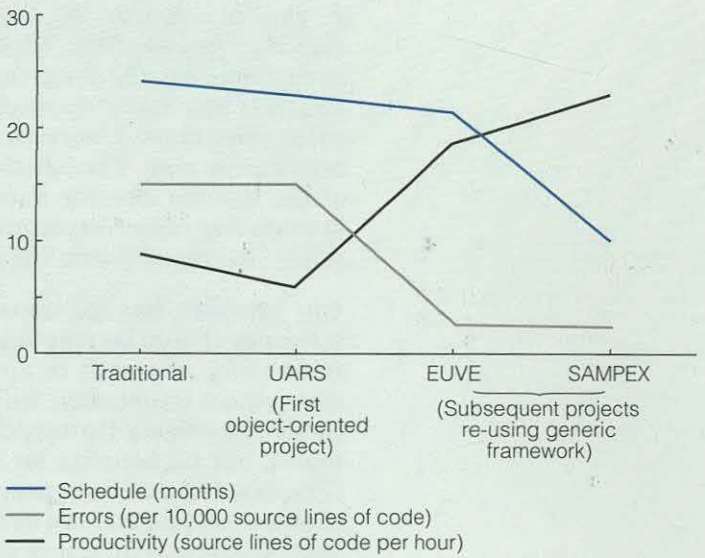
Systems directors must communicate a clear vision of the future

Object orientation is not just a new programming language; it is a different way of developing systems. Application developers are likely to regard it with some suspicion and to have difficulty putting it to use. The systems director must therefore first convince himself that object orientation represents the future of information systems, and that the initial difficulties will be fully repaid by the subsequent benefits. He must then communicate his vision of this future, and its impact on applications, technical architecture, the development process, and the organisation and management of the systems department, to all those involved with developing and supporting applications, both systems staff and users. We believe that

Figure 1 Use of object-oriented development (and Ada) enabled significant benefits to be achieved at NASA's Goddard Space Flight Center

At NASA's Goddard Space Flight Center, staff from Computer Sciences Corporation's System Sciences Division develop software for NASA. CSC had extensive experience of developing simulators (such as the dynamics simulator for the Gamma Ray Observatory). Despite the introduction of Ada, however, productivity and quality were somewhat disappointing. In 1987, CSC decided to take a new approach to its next simulation, the telemetry simulator for the Upper Atmosphere Research Satellite (UARS). Instead of simply creating the simulator, CSC planned to create classes that would be re-usable in this and future simulators. To do this, it used object orientation to design the classes. Then, because Ada is not an object-oriented language, it implemented each class as a kind of Ada module, known as a 'generic package'. One generic package, Sensor, is used 17 times. Through further review, CSC reduced the amount of new code to be developed another 30 per cent. The UARS simulator comprises 69,000 lines of code; without the generic packages, it would have been more than 100,000 lines of code. Significant time and effort savings were identified.

The generic framework was used in two subsequent projects of broadly similar size. In the Extreme Ultraviolet Explorer (EUVE) simulator, re-use was estimated at 80 per cent at design review, but 89 per cent re-use was actually achieved. Time and effort savings were estimated at 67 per cent and the error rate was reduced ten-fold. The Solar, Anomalous and Magnetospheric Particle Explorer (SAMPEX) telemetry simulator also showed substantial benefits.
The impact of the use of object-oriented development is illustrated in the diagram below.



this vision must be created and strongly promoted by the systems director himself, and not just by an advanced technology group or the development manager.

Object orientation can be applied to all kinds of commercial information systems, but it is particularly suited to the new classes of applications that are starting to emerge, especially those that make extensive use of graphics and images. These applications include decision-support systems, multimedia applications, and all kinds of applications to support groupwork, from enhanced electronic mail to complex workflow-management systems. Object orientation should be viewed both as an enabler and as a driver for these applications.

Its implications for technical architecture are also wide-ranging. Object orientation is appropriate to standalone PC applications, and to traditional host-terminal systems, and can deliver benefits in both cases. It is, however, likely to deliver the greatest benefits in a client-server architecture, and the majority of the early success stories reflect this. Any vision of the future of information systems must include the building of systems from interchangeable components: client-server and object orientation are just different manifestations of this.

The third aspect of the vision is the fundamental change in the way that systems are developed. One important aspect of this is a shift towards iterative development, where rapidly developed

prototypes evolve towards the finished system (and where maintenance is simply a continuation of this process). Another will be the increased emphasis on assembling systems from existing objects, including objects purchased from third parties in the form of class libraries.

This, in turn, suggests the fourth aspect of the vision: a change in the management of the information systems function. Object-oriented development will require different kinds of people. Those charged primarily with developing new classes of objects will need to be high-calibre analytical and abstract thinkers. Those charged primarily with assembling applications from existing classes will need different skills from those of traditional developers, some of them more people-oriented. Both will need new kinds of management and motivation.

The vision will communicate the message that both the benefits and the changes implied by full-scale object orientation are very large. It will also help to keep people focused on these major goals, even though the early phases of implementation may be small-scale.

Object-oriented technology should be deployed first on workstations

At present, the best place to start deploying object-oriented technology is on workstation applications. This is where the need for new approaches is greatest, and where the technology of object orientation is strongest. Workstation-based applications will deliver early benefits with relatively little risk. This starting point will provide a good base from which to develop an understanding of the technology, and eventually, to extend its use to other kinds of applications. The experience of the financial-services company, described overleaf in Figure 2, demonstrates how success with the first workstation-based application encourages further object-oriented development. To ensure that the first project is a success, it is essential to choose the right people for the first team, to select high-performance development technology, to adopt an iterative life cycle and to use object modelling to support development.

Create the first team

Selecting the right team of people for the first project is critical. The ability of people to adapt to object orientation varies considerably because learning object orientation is not about learning a new syntax, but about learning to 'think objects'. Those who are best at it are generally those with analytical power, creativity and exposure to entity-analysis methods. Staff with long experience of structured methods, especially in low-level languages, are unlikely to find it easy to transfer their skills to an object-oriented environment.

The first project should be undertaken by a small team. The team will need to be trained, initially in the concepts of object orientation, and immediately thereafter with hands-on experience of an object-oriented language. The team should include one proven object-orientation expert. He will teach the team, give advice on the right way to go, and review designs and programs. He will probably be a consultant, whose fees will be high, but this is an essential investment.

Figure 2 A financial-services company has used object-oriented technology to build new applications for workstation-based systems

The UK subsidiary of a US-based financial-services group had built up several systems on separate IBM AS/400s. They functioned well individually, but the information held on them could not be accessed in an integrated fashion. The company decided to implement client-server systems to provide brokers with better access to information, and it chose object-oriented technology to obtain faster delivery of systems. Object-oriented software, developed by the US-based parent company, provided a basic toolkit to handle some of the interfaces.

A client-server approach was adopted using PCs running Windows 3.0, and several Sun servers running either Sybase or Open Server/SQL to access the AS/400 databases.

The first application was a customer-support system for brokers. To develop this, the company's approach was "take the best people, put them in the middle of the business and give them aggressive timescales". This meant physically locating the development team with the brokers, initially to obtain maximum involvement from business users in prototyping. This was so successful that the team stayed there during subsequent phases.

The project started in 1991 and took place over a period of eight months. The first phase involved the business users in prototyping, and Visual Basic was used to design the screen layouts.

Following this phase, the system was developed using C++ as the programming language for all workstation applications. The system comprised 50 classes (30 in the applications, 20 in the supporting software) and took roughly three months to code and build. The UK company had no prior experience in C++, so programmers experienced in that language were brought in to work on the team and transfer their knowledge.

The initial reaction to the systems has been so good that almost all the business units in the company now want workstation-based systems. (This is due to the high visibility of the development team and the attractiveness of the graphical-user-interface-based solution rather than object-oriented technology *per se*.) The development productivity improvements have been promising; it is thought that the company's traditional approaches would have doubled the timescale for the project. Further improvements in productivity are expected now that the team is familiar with the technology.

The company is continuing to develop its portfolio of workstation applications using C++, but does not expect its use of object-oriented technology to extend to applications on other platforms in the short term, since support for the language is not currently thought to be available.

Adopt high-performance development technology

Development departments that are implementing object orientation on powerful workstations also need to choose an integrated toolkit and to buy class libraries to ensure that they reap the potential benefits.

Today, there are several object-oriented programming languages for workstations and PCs, but the most significant is Smalltalk and we recommend it as the starting point. Smalltalk is a pure object-oriented system. Its power arises not only from the language but from the provision of a highly interactive development environment, which makes programming and debugging very fast, and from the existence of a library of classes, which may be incorporated into new applications.

The re-use of existing classes allows an experienced developer, who knows the class library, to make a fast start. Systems departments should initially buy class libraries for basic functions, such as graphical user interfaces or list management, because the benefits of object orientation will be obtained fastest if development starts from existing class libraries. A small user survey, reported by Borland at ObjectWorld in July 1992, showed that developers using

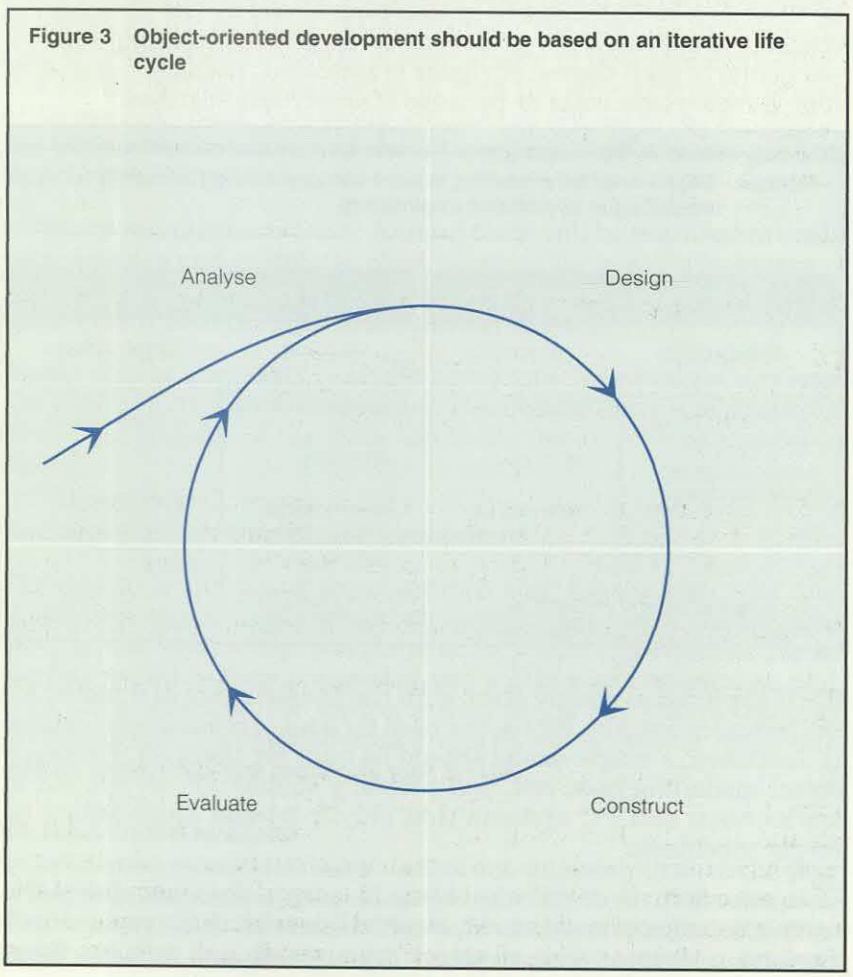
class libraries saw re-use benefits in two months, while those not doing so waited eight months. With new technical class libraries appearing every week, and business class libraries beginning to emerge, it will be wise for systems departments to buy wherever it is possible to do so, rather than try to develop them in-house.

Adopt an iterative life cycle

Object-oriented development projects can follow the traditional waterfall life cycle, but the nature of object-oriented development makes this unnecessary. In most object-oriented projects carried out to date, development is done by a small team of skilled developers equipped with powerful tools. The method is akin to prototyping; analysis and documentation are typically downplayed. Productivity is often high, but the lack of easy milestones makes the process hard to manage and the lack of documentation creates risks and may create maintenance problems.

These problems may be addressed by adopting an iterative project life cycle in which each iteration includes analysis, design, construction and evaluation. This is illustrated in Figure 3. With an iterative life cycle, each stage may be seen as a refinement of the previous stage, and little of the work done is discarded. If an object-oriented language (and preferably an object-oriented database management system) is used for construction, this holds true even for the finished software; a one-to-one relationship will exist between

Figure 3 **Object-oriented development should be based on an iterative life cycle**



classes in the domain model and classes in the compiled program. Because the original conceptual model is not distorted or hidden by development work, changes to the model can be rapidly expressed in software. Furthermore, maintenance can be managed in the same way as development (or *vice versa*) – a useful simplification of the project-management task.

Introduce modelling to support development

Object-oriented technology is supported by development systems, such as Smalltalk, that facilitate very rapid construction of working prototypes, which engage the user's interest and enable requirements to be established rapidly. Rapid application development is not always a complete solution, however. It is relatively weak for complex logic and structure that is not visible to the user, and it focuses user attention on current problems and solutions rather than on longer-term concerns. These problems may be addressed by the use of modelling.

Modelling is a critical component of object-oriented development and it provides benefits even if object-oriented technology is not used to construct the system. Most systems departments have used entity modelling for many years, as part of modern structured methods. Object modelling improves on this by depicting behaviour as well as attributes and relationships (see Figure 4), so making it possible to create clear and intelligible models that reflect the world more closely. We therefore recommend that object modelling be used as a complement to prototyping to clarify the structure of complex areas, as a pre-prototype on large projects, and where a particularly high degree of rigour is intended, perhaps because of high downstream costs or because of safety implications.

Figure 4 Object-oriented modelling is more comprehensive than entity modelling in Information Engineering

Information Engineering		Object orientation	
Construction	Modelling	Modelling	Construction
Implemented as database design	Entities	Classes	Implemented as classes
	Attributes	Attributes	
	Relationships	Relationships	Implemented as methods encapsulated in the classes
		Behaviours	
Implemented as programs	Dataflow diagrams		

Each application should start with the construction of a model for the business domain. Where the need for a set of related applications is identified, a single model should be built for them all. However, object modelling need not, and probably should not, be applied to workstation and PC systems that only front-end applications on larger machines.

This approach is relatively cheap to adopt, does not delay the completion of applications, develops in line with the organisation's growing understanding of object orientation, and will, in time,

yield most of the benefits of a more comprehensive commitment to modelling.

New approaches to application development will be required

A first successful application on workstations will serve to introduce object orientation into the systems department and to build up experience with the technology. It is, however, only a start. Once the early applications have been implemented successfully, systems managers should extend the use of object orientation. This will mean increasing the number of development teams and support for them, introducing more disciplined procedures, creating a central library, and introducing more object-oriented technology as it becomes available.

Expand the teams and provide organisational support

New teams should continue to be created, building on the strengths and experience of the first. The expert in each new team will be someone from the initial team, and new team members should undergo the same training process as the first. Once object orientation is proven and established, we would expect to find the following specialised roles or groups:

A modelling centre: On the successful completion of the first modelling exercise (or, perhaps, of the first two), the systems department should create a small centre of competence in business modelling. The centre will be the custodian of both the emerging models and of the methods used. This group will become critical to the successful use of object orientation and should comprise staff who are good at abstract thinking.

A library-management team: A small team will be required to evaluate, acquire and customise class libraries for generic functions – for example, database, communications and graphical user interfaces – from both internal and external sources.

It may also be necessary to create new posts (for example, a re-use coordinator), and to alter existing job descriptions – perhaps by extending the role of the data administrator to include re-usable objects.

Introduce more disciplined procedures

The experience of many organisations that have begun with the creative development of workstation applications is that, after several months, the team recognises the need for further disciplines. These can be created as they are needed and will be more acceptable, and thus more effective, for having grown out of the team's own experience. The two most obvious areas in which greater discipline will be beneficial are methods and re-use.

Object-oriented methods

Development teams that have started out with very little in the way of methods have often recognised the need for them as the scope of their systems has increased. The field of object-oriented development methods is evolving rapidly; the main contenders are listed

in Figure 5. Each has its origins in work in a particular language, often Ada or C, or in a particular application area, such as embedded realtime software or data processing. In the data processing area, the leading methods seem to be those created by Sally Shlaer and Stephen Mellor, by James Rumbaugh and co-workers at General Electric, and by Peter Coad and Edward Yourdon.

Figure 5 There are many object-oriented methods

Since the names of the methods are not distinctive, the methods are usually known by the names of their authors.

Author	Method
James Rumbaugh and co-workers at GE	Object Modeling Technique
Peter Coad and Edward Yourdon	Object-Oriented Analysis and Design
Edward Colbert at Absolute Software Co Inc	Object-Oriented Software Development
Sally Shlaer and Stephen Mellor at Project Technology Inc	Object-Oriented Systems Analysis
Ian Graham at BIS	Semantic Object Modelling Approach
Grady Booch at Rational Technology	Object-Oriented Design
Derek Coleman and Fiona Hayes at Hewlett-Packard Laboratories	Fusion
R J A Buhr	Visual Techniques in System Design
ParcPlace Systems staff	Object Behavior Analysis
Staff at the European Space Agency	Hierarchical Object-Oriented Design (HOOD)
Ivar Jacobsen of Objective Systems	ObjectOry

Despite their differences, object-oriented methods appear to be converging on a common, rather general set of modelling concepts, and now have generally similar forms, as shown in Figure 6. The early stages of the methods are, in fact, similar to those of modern structured data processing development methods.

Choosing which object-oriented method to use is a difficult task, since none is complete. For instance, none offers more than very general advice about the re-use of existing object-oriented models and of the models implicit in available class libraries. Nor do they have much to say about designing to foster re-use. These omissions doubtless reflect the fact that most object-oriented methods have been in use for only limited periods. Despite the immaturity of the field, most organisations should expect to build their models in accordance with the analysis phase of a commercially available method for which there is support in the form of books, training courses and expert advice. They should bear in mind, however, that in such a new field, no-one has a monopoly on wisdom; it will therefore be worth learning from what other methods have to offer, too.

Software re-use

As object-oriented projects proliferate, it will probably be necessary to adopt more formal procedures to ensure re-use. The fact that a system has been developed using an object-oriented method and object-oriented tools does not guarantee that the design will facil-

Figure 6 Today's object-oriented development methods have similar forms

All current object-oriented methods contain domain analysis, design, construction, testing, tuning and maintenance.

Domain analysis

The analyst constructs a model of the domain. The degree of detail varies according to the method.

*Scope definition**: The scope of the domain to be modelled must first be defined. This is largely a matter of experience and may be adjusted after class identification if the initial scope is too large. If a large scope (more than, say, 30 classes) is essential, the model should be built as several sub-models that are then reconciled.

*Class identification**: Classes are defined for each different kind of thing, person, process or event in the domain. Classes are documented in an object model.

*Attributes**: Attributes define the information that must be held for each instance of the class.

*Relationships**: Class/sub-class relationships within the domain are identified. Relationships between instances – for example, Customer OWNS Account, Sub-assembly COMPRISES Parts – are identified and added to the object model.

Object life cycle: The various possible states of an object are described and the triggers that move an object between these states are identified. For example, a Withdrawal may make an Account go from In Credit to Overdrawn. This is usually described in a state diagram.

Design

The designer defines a sub-set of the model that meets the requirements. Methods express changes of state defined in the state diagram.

Construction

The model sub-set is then implemented, preferably using an object-oriented language. If some features of the model cannot be directly expressed in the language – for example, multiple inheritance in Smalltalk, and relationships in most object-oriented systems – the implementation will diverge from the model.

Testing

Testing starts with individual classes and proceeds through groups of related classes to complete applications.

Tuning

Object-oriented technology provides a powerful means of tuning the system for improved performance. Changes to a class high in the hierarchy apply to all its sub-classes.

Maintenance

System maintenance consists of further cycles of analysis, design, construction, testing and tuning.

* Also occur in entity modelling

itate re-use of the software in other systems, or that developers will automatically seek to re-use the software in other systems. There are, undoubtedly, barriers to re-use, and overcoming them will require the creation of a development environment in which it is easy to find and re-use software components. In some cases, initial success has given way, in the longer term, to failure, due to a lack of continuing management attention.

Technical aids to re-use are limited, so the main emphasis must be on human factors such as management attitudes and development policies, until re-use is fully integrated into the culture of the systems department. Developers should be rewarded for creating components that can be re-used. (Some organisations pay a

'royalty' on re-used components.) At least during the transition period, it may be sensible to reward re-use itself, although the rewards need not be monetary.

Create a central library

There is much to be gained by creating a central library of software components to form the basis of further application development. First, it will usually take much less effort to find a software component in the library and to incorporate it in an application than to design, build and test a new one. Second, components are, in effect, tested every time they are used in a new application, so the quality of components in the library is continually improved. Third, if several applications use the same components for common functions, they will be more consistent in appearance and behaviour than if each contains its own code for these functions.

Obviously, only classes that are of high quality and wide applicability should be added to a corporate class library, and a quality check is essential first. Such a check would be stricter than those applied to classes that will not be re-used. It should include documentation, which should be available online, compliance with standards, and portability.

Introduce more object-oriented technology as it becomes available

During the last three years, the pace of development of object orientation has increased, and in many areas, it has emerged from its various niches to challenge established tools and techniques. Products in some areas are still, however, immature – most class libraries are purely technical in scope, object-oriented database management systems have serious deficiencies, and some of the tools do not support end-user development. Developments in these areas must be monitored actively to decide when and where to extend the use of object-oriented technology as proven products emerge.

The current immaturity of some aspects of the technology may tempt systems directors to delay the introduction of object orientation. However, the technology is maturing rapidly and it will take at least five years for most organisations to introduce object orientation fully (although benefits should be obtained in the first year). There is therefore a risk that many organisations will still be maintaining a watching brief long after the technology has matured. In the sense that it supports many of the longer-term trends in information technology, object orientation may be said to be the technology of the future; for many leading users, for whom it is already delivering great value, it is unarguably the technology of the present.

The Foundation

The Foundation is a service for senior managers responsible for information management in major enterprises. It provides insight and guidance to help them to manage information systems and technology more effectively for the benefit of their organisations.

The Foundation carries out a programme of syndicated research that focuses on the business implications of information systems, and on the management of the information systems function, rather than on the technology itself. It distributes a range of publications to its sponsors that includes research reports, management summaries, directors' briefings and position papers. It also arranges events at which sponsors can meet and exchange views, such as conferences, management briefings, research reviews and study tours.

Foundation sponsors

The Foundation is the world's leading programme of its type. The majority of sponsors are large organisations seeking to exploit to the full the most recent developments in information technology. The sponsorship is international, with more than 450 organisations from over 20 countries, drawn from all sectors of commerce, industry and government. This gives the Foundation a unique capability to identify and communicate 'best practice' between industry sectors, between countries, and between information technology suppliers and users.

Benefits of sponsorship

The scale and diversity of sponsors establishes the Foundation as the largest and most influential programme for systems managers anywhere in the world. Sponsors have commented on the following benefits:

- The publications are terse, thought-provoking, informative and easy to read. They deliver a lot of messages in a minimum of precious reading time.
- The events combine access to the world's leading thinkers and practitioners with the opportunity to meet and exchange views with professional counterparts from different industries and countries.
- The Foundation represents a network of systems practitioners, with the power to connect individuals with common concerns.

Combined with the manager's own creativity and business knowledge, participation in the Foundation contributes to managerial success.

Recent research reports

66 Marketing the Systems Department
67 Computer-Aided Software Engineering (CASE)
68 Mobile Communications
69 Software Strategy
70 Electronic Document Management
71 Staffing the Systems Function
72 Managing Multivendor Environments
73 Emerging Technologies: Annual Review for Managers
74 The Future of System Development Tools
75 Getting Value from Information Technology
76 Systems Security
77 Electronic Marketplaces
78 New Telecommunications Services
79 The Role of Information Technology in Transforming the Business
80 Workstation Networks: A Technology Review for Managers
81 Managing the Devolution of Systems Responsibilities
82 The Future of Electronic Mail
83 Managing Technical Architecture
84 Downsizing Computer Systems
85 Visual Information Technology
86 Strategic Alignment
87 Implementing Open Systems
88 Introducing Object Orientation

Recent position papers and directors' briefings

The Changing Information Industry: An Investment Banker's View
A Progress Report on New Technologies
Hypertext
1992: An Avoidable Crisis
Managing Information Systems in a Decentralised Business
Pan-European Communications: Threats and Opportunities
Information Centres in the 1990s
Open Systems
Computer Support for Cooperative Work
Outsourcing Information Systems Services
IT in a Cold Climate
New Directions in Client-Server Systems
Object Orientation
Innovation: The Challenge in Exploiting Technology
Information Appliances: Computers for the 1990s

Forthcoming research reports

Pervasive Computing
Quality Management in the Systems Function

CSC Index

The Foundation is one of the services provided by CSC Index. CSC Index is an international consulting group specialising in information technology, organisational development and business re-engineering. Its services include management consulting, applied research and education.

CSC Index
A company of Computer Sciences Corporation

United Kingdom
12 Bloomsbury Square
London WC1A 2LL
☎ 44 (0)71 831 0101
Fax 44 (0)71 831 6250

8 Clifford Street
Mayfair
London W1X 1RB
☎ 44 (0)71 434 2512
Fax 44 (0)71 439 7357

Cranfield IT Institute Limited
Fairways, Pitfield
Kiln Farm
Milton Keynes MK11 3LG
☎ 44 (0)908 569333
Fax 44 (0)908 569807

Australia, New Zealand and South-east Asia
Mr J Cooper
Butler Cox Foundation
Level 10, 70 Pitt Street
Sydney, NSW 2000
Australia
☎ 61 (0)2 223 6922
Fax 61 (0)2 223 6997

Belgium and the Netherlands
Prins Hendriklaan 52
1075 BE Amsterdam
The Netherlands
☎ 31 (0)20 6 75 51 11
Fax 31 (0)20 6 75 53 31

France
110/114 rue Jules Guesde
92300 Levallois-Perret
☎ 331.47.56.90.83
Fax 331.47.56.91.31

Germany, Austria and Switzerland
Butler Cox GmbH
Bavariaring 24
8000 München 2, Germany
☎ 49 (0)89 5 32 93 90
Fax 49 (0)89 5 32 93 999

Ireland
SD Consulting
8 Clanwilliam Square, Dublin 2
☎ 00 (0)1 764701
Fax 00 (0)1 767945

The Nordic region
Mr R Bydler
Box 160, Borganäsvägen 44
S-781 22 Borlänge, Sweden
☎ 46 (0)243 886 60
Fax 46 (0)243 124 50

Spain and Portugal
T Network SA
Núñez Morgado 3-6°b
28036 Madrid, Spain
☎ 34 (9)1 733 9910
Fax 34 (9)1 314 3198

United States of America

Headquarters in Boston, and offices in Chicago and San Francisco