South African
Computer
Journal
Number 14
July 1995

Suid-Afrikaanse
Rekenaar-
tydskrif
Nommer 14
Julie 1995

Computer Science and Information Systems

Rekenaarwetenskap en Inligtingstelsels

The South African Computer Journal
An official publication of the Computer Society of South Africa and the South African Institute of Computer Scientists

Die Suid-Afrikaanse Rekenaartydskrif
'n Amptelike publikasie van die Rekenaarverenging van Suid-Afrika en die Suid-Afrikaanse Instituut vir Rekenaarwetenskaplikes

Editor
Professor Derrick G Kourie
Department of Computer Science
University of Pretoria
Hatfield 0083
Email: dkourie@dos-lan.cs.up.ac.za

Subeditor: Information Systems
Prof Lucas Introna
Department of Informatics
University of Pretoria
Hatfield 0083
Email: lintrona@econ.up.ac.za

Production Editor
Dr Riël Smit
Mosaic Software (Pty) Ltd
P.O.Box 23906
Claremont 7735
Email: gds@mosaic.co.za

World-Wide Web: http://www.mosaic.co.za/sacj/

Subscriptions

	Annual	Single copy
Southern Africa:	R50,00	R25,00
Elsewhere:	$30,00	$15,00

An additional $15 per year is charged for airmail outside Southern Africa

to be sent to:
Computer Society of South Africa
Box 1714 Halfway House 1685

Phone: +27 (11) 315-1319 Fax: +27 (11) 315-2276

Guest Contribution

About the author

Dr. Matthew Jones from the University of Cambridge (Management Studies Group) is in fact trained in the Agricultural and Environmental Sciences with a PhD from the University of Reading (1984). He currently holds the position of University Lecturer in Information Management at Cambridge.

He was the secretary and treasurer of the British Institute of Energy Economics Energy Modelling Study Group 1984-1992, member of the Cambridge Health Authority Information Strategy Group 1989-1991. He is also currently member of the British Human Computer Interaction Group, member of the DTI Human Interface Club Special Interest Group on the Organisational Aspects of Information Technology and treasurer and board member of HICOM Human Computer Interaction electronic conferencing system.

He is a highly regarded scholar and has published a substantial number of academic papers in scholarly journals. Some of his more recent publications on business process reengineering (the topic of his contribution) are: (a) Jones, M.R. Don't emancipate, exaggerate: rhetoric, reality and reengineering. In: Baskerville, R.L.; Smithson, S.; Ngwenyama, O. & DeGross, J.I. (Eds) Transforming Organizations with Information Technology. North Holland, Amsterdam, pp357-378, (1994). (b) Jones, M.R. Empowerment and Enslavement: Business Process Reengineering and the Transformation of Work. Paper presented at the "New Visions of the Post-Industrial Society" conference. University of Brighton, (9-10 July 1994) (c) Jones, M.R. The Contradictions of Reengineering. Paper presented at the "Management Challenges in IS" Conference, Cranfield University, 12-13 July 1994.

Business Process Reengineering: Management's New Paradigm or Emperor's New Clothes?

Matthew R Jones
University of Cambridge, Management Studies Group, Department of Engineering, Mill Lane, Cambridge CB2 1RX, UK
e-mail: mrj1@uk.ac.cam.eng

Abstract

Business Process Reengineering (BPR) is currently attracting much attention as an approach to organisational change, but has received relatively little critical scrutiny. This paper seeks to examine the origins, antecedents and content of BPR as they are described in the main writings on the topic, in order to address three questions: Is there one form of BPR or many? Is it new? What factors may account for its popularity? Some of the criticisms that have been made of BPR are also discussed.

Keywords: *Business Process Reengineering*
Computing Review Categories: *J.1*

1 Introduction

Business Process Reengineering (BPR) has been described as the "hottest management concept since the quality movement" [8] and as "one of the key management concepts of the 1990s" [23]. Klein [41], for example reports that 88% of a survey of senior executives claimed that they were engaged in reengineering projects, while Cane [10] reports that up to 70% of large UK companies have undertaken, or are about to undertake BPR initiatives. It has certainly also received its fair share of attention in the management literature in recent years. An on-line search using the terms "Business Process Reengineering", "Reengineering", "BPR" and "Business Process Redesign" in early 1995, for example, identified 3,592 journal, magazine and newspaper articles on the topic since 1990.

Yet, despite the huge amount that has been written about it, the concept of BPR remains elusive. Is it simply

the latest buzzword, "that people bandy around ... even if they are just putting in a new computer system" as some popular reports suggest [65], or is it the most important contribution to thinking about organisations since Adam Smith's *Wealth of Nations* as Hammer & Champy [29] claim? Is it primarily an extension of traditional time and motion studies as Parker [54] states, or of Total Quality Management as Shillingford [57] describes, or is it an entirely "new paradigm" [26]? And what is the relationship between Information Technology (IT) and BPR? IS BPR simply a way of "rebadging" existing IT products to increase sales [65], or is it one of the resources to be reengineered as Morris & Brandon [51] imply, or should IT be seen as enabling new processes as Coulson-Thomas [11] suggests, or is it the primary enabler as Davenport [13] proposes?

The confusion surrounding these questions suggests the need for a critical review of the concept of BPR; to examine its origins, antecedents and content. From this it is hoped to develop a clearer picture of the nature of BPR. Is there only one type of BPR or are there several alternatives? What is the substantive content of BPR and how much of this is new? What factors may account for the interest shown in it? This paper therefore seeks to complement the small number of critical analyses of BPR, such as Grint [25], Willmott [67] and Grey & Mitev [24] that have recently begun to emerge. It differs from them however in focusing primarily on the "theory" of BPR, such as it is, rather than seeking to locate the concept within broader social and organisational changes.

For practical reasons it should be clear that this paper cannot aim to provide a full literature review. Moreover a large proportion of the articles consist of reports (frequently quite brief) on the BPR phenomenon for different audiences, or "case studies" of BPR success (often written by the consultants involved), rather than discussions of the nature of the concept. This paper therefore concentrates on those publications widely cited as influencing the development of BPR, or offering a more extended discussion or substantial report on it.

2 Origins of BPR

A number of writers have sought to suggest that the origins of BPR go back many years. Holtham [34], for example suggests that the concept of 'breakthrough' management coined by Juran [39] is "virtually the same" as reengineering, while Bartram [3] traces the "theoretical background of reengineering" to a 1961 Harvard Business School paper on work simplification. Klein [42] even identifies reengineering projects in the Nineteenth Century and cites Henry Ford as an early exponent.

Whatever the merits of these arguments, the evidence of the on-line search of bibliographic databases indicates that the vast majority of the literature on BPR has appeared in the last five years, with 96% being published since 1992. This may be attributed to the influence of two articles published in the latter half of 1990 in the *Harvard Business Review* and *Sloan Management Review*. Both discussed the need for companies to reorganise along process lines, exploiting the opportunities provided by developments in Information Technology.

The *Harvard Business Review* article, "Reengineering Work: Don't Automate, Obliterate" by Michael Hammer [28], a former computer science professor at MIT, is the more widely cited and is credited by many with starting the BPR bandwagon rolling. It introduced the term "reengineering" and set the tone for much later writing on the topic. The rather more academic article in the Sloan Management Review by Thomas Davenport & James Short entitled "The New Industrial Engineering: Information Technology and Business Process Redesign" [14], was associated with the MIT "Management in the 90s" programme [56]. This introduced the acronym BPR and also, as the title indicates, explicitly linked it with IT. The ideas in this paper were later extended by Short & Venkatraman [58] to address processes operating across the boundaries of the traditional business firm, an activity they called Business Network Redesign.

Although Hammer [28] does not actually use the term "Business Process Reengineering", and indeed has recently criticised its widespread use in association with his ideas [30], when he described reengineering as "the radical redesign of business processes" he was clearly talking about a similar activity to that of Davenport & Short. The two papers also referred to some of the same examples as illustrations of their concepts. Thus, even if reengineering and business process redesign are not completely synonymous they would appear to be sufficiently closely related (in practice, if not necessarily in principle) that the use of BPR as an acronym to describe them both would seem justified. On similar grounds, therefore, BPR will, for the purposes of this paper, also be used to describe the various process-oriented approaches to significant business re-organisation that have emerged since 1990, even though their authors may originally have employed alternative names such as Business Process Improvement, Business Reconfiguration, Business Reengineering, Core Process Redesign, Corporate Reengineering, or Process Innovation.

3 Antecedents of BPR

In his original *Harvard Business Review* article [28], Hammer did not seek to associate BPR with any theory of organisations, indeed he presented it specifically as overthrowing arrangements of work organisation that he traced back to the Industrial Revolution. Instead, his argument was based pragmatically on the experience of process redesign in two companies. Although the argument and examples were later elaborated in Hammer & Champy [29], this also does not attempt to develop a theoretical basis for BPR. Rather it was suggested that it is a "conceptually new business model" with no precursors.

Davenport & Short [14], in contrast, described BPR

as "the new industrial engineering (IE)". Earlier IE, they explain, was based on F.W. Taylor's extremely successful "mechanizing vision". This was effective in the past, they argue, because the business environment was "largely stable". In today's turbulent business environment, however, a new approach is needed. It is not clear, though, whether Davenport & Short see BPR as rejecting, transcending, or pragmatically adapting Taylorism, although it would appear that they certainly wish it to be seen as lying within the tradition of the "scientific school of management". This view would seem to be supported by Parker [54] who describes BPR as involving time and motion techniques, by Morris & Brandon [51] who associate it with "IE, time and motion, Operational Research and Systems analysis" and by Glover (quoted in [65]) who suggests that it is "upmarket O & M". Wheatley is quoted in Brown [6] as describing BPR as the "supernova" of this "mechanical view of organisations".

Although Hammer and Davenport & Short are generally acknowledged as the progenitors of BPR, it does not mean that their view of the topic can necessarily be seen as definitive. BPR is a commercial product as much as a theory of work organisation. For the customer of BPR, therefore, it is the approach they buy which is likely to define it for them, even if this does not conform to Hammer or Davenport & Short's model. Thus, for example, despite Hammer & Champy 's emphatic statement that Reengineering is not "the same as quality improvement, total quality management (TQM), or any other manifestation of the contemporary quality movement" [29], Shillingford [57] describes BPR as "often evolving out of a TQM initiative" and suggests that its rationale is that "concepts like Japan's 'best manufacturing practice', which uses Just-in-time techniques, can be applied to the office as well as the factory". This view is not a simple misunderstanding, but is based on the approach to BPR developed at Rank Xerox, who were selling it, with accompanying software tools, as a consultancy service (at a starting price of $100,000) [22]. The association of BPR with TQM is also made by Johannson *et al* [36] and *inter alia* by Davenport [13], while Earl [19] and Mumford [52] link it with lean production [68].

Other writers have identified BPR with a number of different organisational and technical developments. For example Coulson-Thomas [11] links it to the emergence of the "network organisation", Stewart [59] to the "horizontal organisation" [53] and Kaplan & Murdock [40] to Dichter's vision of the organisation of the 90s [17]. More bluntly, Byrne [8] suggests that BPR can sometimes be seen as "little more than a euphemism for laying people off", and it is certainly recognised in the popular BPR literature that part of the interest in it comes from companies seeking to reduce staff numbers in the face of economic recession [65]. In some reports BPR is also linked with the implementation of specific types of technology, usually image processing and workflow systems for office automation [57].

Given that the theoretical antecedents of BPR may be expected to be reflected in the approach adopted, i.e. those seeing it as an extension of TQM may be expected to draw on techniques and social structures historically associated with Quality programmes, then this diversity of views on the source of BPR thinking could suggest that there are several distinct versions of it. Another possibility is suggested by Davenport [13] who argues that Process Innovation, as he calls it, does not have a single source, but has developed from the combination of five different approaches to business improvement. He identifies these precursors as: the quality movement, industrial engineering (and its offshoot the soft systems movement *sic*) , the socio-technical school, the technology transfer literature and the IT for competitive advantage literature. Even if it is accepted that these approaches can practically be combined, however, it is possible to question whether such a mixed pedigree would seem any more likely to yield a consistent approach to BPR than the various different precursors on their own.

As has already been noted, however, BPR is not just a theoretical construct, but a guide to practical organisational intervention. Whether BPR's theoretical antecedents are consistent, or even what they are thought to be, may not matter in practice, therefore, if what is actually advocated is the same for all the different 'versions', or at least sufficiently similar that a common approach is likely to be adopted. The next section will therefore consider the content of BPR as it is presented by a number of different authors to see if they share a common view of what BPR involves.

4 Content of BPR

There have been many attempts in the BPR literature to define the concept, not least by Hammer & Champy [29] who devote a whole chapter of their book to an elaboration of their statement that it is "the fundamental rethinking and radical redesign of business processes to achieve dramatic improvements in critical, contemporary measures of performance, such as cost, quality, service and speed". Unfortunately, few of the definitions show more than a limited consensus on its key features, particularly when the terms employed are examined in more detail. While some of the differences revolve around distinctions that would probably have appealed to mediaeval theologians, others mark more significant divisions within the field.

Process orientation

Perhaps the only real common ground between all BPR approaches is their view of organisations as being made up of *processes*. Hammer & Champy [29] define a process as "the collection of activities that takes one or more kinds of input and creates an output that is of value to the customer". This is contrasted with the traditional perspective which sees organisations as made up of separate functions (such as marketing, finance, sales and so on), typically located in their own specialist departments.

While agreed that BPR involves a focus on processes, there is disagreement in the literature about their scope.

There would appear to be four different "levels" at which these may operate. At the lowest level there are individual work tasks, such as customer inquiry handling, which combine various different, if simple, activities. At the second are work processes which may operate within a single functional department. At the third level the processes operate across departments within an organisation, and at the highest level they are inter-organisational. Each of these are, in principle, compatible with Hammer & Champy's definition, particularly if we take the currently popular view of organisations as markets, in which customers may be internal as well as external.

Harrington's concept of a business process which he defines as "all service processes and processes that support production processes" [31] would seem to operate at the first two levels, that is, within established departmental boundaries. Similarly, Morris & Brandon [51] define a process as "larger than a task ...[but] smaller than an area of business such as operations, human resources or shipping", while Hall *et al* [27] state that "a process can be as narrowly defined as a single activity in a single function". Davenport & Short [14] also talk of redesigning "interpersonal processes" which "involve tasks within or across small workgroups typically within a function or department". Hammer & Champy [29], however, suggest that processes to be reengineered typically operate across traditional functional/departmental boundaries, a view shared by Johannson *et al* [36] and Davenport [13] amongst others. BPR, Hammer & Champy argue, involves "breaking down the functional silos" and Davenport [13] states that "adopting process innovations inevitably entails cross-functional and cross-organizational change". Although these writers also allude to the possibility that processes may cross organisational boundaries, this is the particular focus of Venkatraman [64] who talks of "business reconfiguration" and Short & Venkatraman [58] who discuss "business network redesign".

Scale of change

Partly related to the scope of processes is the question of the scale of change, both in terms of extent of reorganisation and the performance improvements sought. Hammer & Champy [29] argue for a resolutely radical approach, describing re-engineering as a "new beginning", a "blank sheet of paper". As Hammer [28] puts it "the point is not to learn what happens to form 73B in its peregrinations through the company, but to understand the purpose of having form 73B in the first place". Thus BPR, Hammer & Champy argue, means "discarding existing processes and replacing them with entirely new ones" so as to achieve a "quantum leap" [in performance]. This view is shared by Johannson *et al* [36] who argue that organisations should be aiming not just for marketplace parity, but for "market dominance". This requires them to "break the china", "to challenge the very purposes, principles and assumptions on which their business is based". Davenport & Short [14] also talk of "fundamental reshaping" and Davenport [13] of "effecting creative and radical change to realize order-of-magnitude improvements". Similarly, Heygate [32] argues that managers have to be "immoderate in their ambitions to improve processes", while Hall *et al* [27] argue that "effective transformation ... requires a clean slate approach to process redesign".

Other writers, in contrast, appear to envisage a more modest, evolutionary activity. Harrington [31], for example, describes BPR as involving the "improvement of inefficient processes" and this is echoed in Fintech [23]. Glover, quoted in Warren [65], also argues that "radically changing the whole ethos of the business may not be necessary: there may be just one division which needs attention". Morris & Brandon [51] do not specifically discuss the scale of change, but the general absence of the revolutionary rhetoric characteristic of Hammer & Champy [29] suggests that their approach is also more incremental.

This point has recently shown signs of becoming a point of serious schism in the BPR literature. Thus as, Yates [70] reports it has lead to a parting of the ways between Michael Hammer and James Champy. Hammer, for his part, likens reengineering to an organisational "neutron bomb" and argues that "trying to do a little bit of reengineering is like trying to be a little bit pregnant". Champy, in contrast, argues that "there are good things that we don't want to destroy ... there is a sense of pragmatism in older managers that I don't think companies can afford to lose". Davenport has also criticised "the myth of the clean slate", arguing that "designing with a dirty slate will often yield a more implementable process" [15].

The means of achieving BPR

Another difference between BPR approaches relates to their ideas on how it should be achieved. Thus, while most writers specify a methodology for conducting BPR, Hammer [28] offers only a set of principles which the exercise should seek to pursue. Even in Hammer & Champy [29], there is no specific method, rather there is a discussion of outcomes, of the roles involved, case studies of the "experience of process redesign" in several companies and a discussion of the most common errors that lead to reengineering failure. Avoid these it is argued "and you almost can't help but get it right".

It would seem clear therefore that Hammer wishes to avoid a formulaic approach to BPR and sees it as a way of thinking and achieving radical organisational change rather than as a rigorous procedure to be followed. This may be contrasted with the recommendations of Morris & Brandon [51] and Booz-Allen & Hamilton [5] who argue that the use of a systematic methodology is essential for BPR success (even if, on closer examination, this often turns out to be a set of vague guidelines of limited practical utility). This view appears to be particularly prevalent in the information systems and industrial engineering literature on BPR (see for example, [26, 42, 66]). Klein [43] suggests that these positions represent two different schools of thought on the approach to BPR which he calls the "intuitives" and "methodologists" and identifies the particular consultancy companies associated with each.

The role of information technology

A potentially important area of disagreement between the different BPR approaches, as was noted in the introduction, is the contribution of IT. Two different roles for IT need to be distinguished here: firstly its use as a tool for the support of the BPR activity, and secondly its position as an axial principle around which the redesign of business processes should be planned. Davenport [13] calls these two roles "IT as implementer" and "IT as enabler" though there are some problems with this terminology as is discussed below. With respect to the first of these there would seem general agreement that BPR may be usefully supported by IT-based tools and some approaches explicitly recommend their use. For example, Morris & Brandon [51] give considerable attention to the use of simulation to test new process designs and identify it as a specific stage in their method, and Davenport & Short [14] discuss the use of CASE tools in drawing process models. Others discuss BPR activities which involve IT support. For example Sutherland [61] discusses the use of "reengineering laboratories" which make use of a variety of IT systems, while Heygate [32] suggests that "the role of IT in building individual's' skills may well prove to be the most valuable application of information technology so far discovered".

On the question of whether reengineered processes should be based on the use of IT, however, there would appear to be a significant divergence of opinion. A number of the key writers, such as Davenport & Short [14] for example, see IT as playing a central role in BPR, making the consideration of IT "levers" one of the five stages in their BPR method and describing as an exemplar a case of "IT-driven process redesign" at Xerox. Similarly, Guha *et al* [26] talk of "IT-induced reengineering". The concept has also been picked up by a number of IT/IS consultants who present it as a new approach to applying IT in organisations. For example Heygate & Brebach [33] explain how their "decision engineering, [IT-supported] organisational flattening, technology scan and IT cost estimation" techniques can support an "IT-driven business strategy", and Aikins [1] discusses how BPR can make use of knowledge-based systems. In some cases BPR is even associated with specific types of IT, usually workflow software and document image processing [57]. These writers would therefore seem to share Davenport & Short's view that IT and BPR "have a recursive relationship ... each is the key to thinking about the other" [14]. From this perspective IT needs to be considered in the early stages of redesign so that "awareness of IT capabilities can – and should – influence process design" [14].

Other writers, however, argue that "BPR and IT are certainly not synonymous" [61] and that "in theory, BPR does not have to involve IT at all" [23], rather it is "the common-sense practice of redesigning business processes before bringing in technology" [21]. Harrington [31], for example, does not mention the use of IT and confines discussion of the use of computers to a section on "automation". Morris & Brandon [51] also state that BPR "is not an IT topic", while Rowland [55] argues that "those who say that IT is a requirement of BPR have missed the point". This does not mean that these writers deny that IT may contribute to BPR, but rather that they believe that process design should precede consideration of IT options. For example Shillingford [57] states that "[o]ne attraction of BPR is that it makes IT subordinate to business objectives ... in most cases 80% of the value of BPR comes out of examining the way things are organised ... [o]nly 20% comes from IT".

An intermediate position appears to be that IT can "enable" new ways of carrying out processes and that redesign should therefore "be undertaken with a full knowledge of how technology can help" [5]. Hammer & Champy [29], for example, devote a chapter to illustrating that shared databases, expert systems, telecommunications networks, decision support tools and so on, "break the rules that limit how we conduct our work" and that IT is therefore a "part of any reengineering effort" whose importance it is "difficult to overstate". This argument, however, smacks rather of technological solutions in search of suitable applications rather than a convincing case that the use of IT is an essential feature of any reengineering activity as they suggest. It may be contrasted, for example, with the statement by Holtham [34] that "very few of the successful case studies of BPR have been initiated out of the new opportunities offered by IT .... the development of technologies such as workflow, groupware or document image processing have no *a priori* connection to the need to reengineer business processes". The concept of IT as an "enabler" may therefore be rather closer to the IT-led position taken by Davenport & Short [14] than might at first appear to be the case.

It is also possible, however, to exaggerate the differences between authors on the role of IT. Thus, as has been noted, those who see IT as secondary may also discuss how it can "make it possible for processes to be carried out in a new way" [23], while the proponents of IT-driven approaches will emphasise that BPR requires a "carefully-considered combination of both technical and human enablers" [13]. The vagueness of terms such as "enabler" means that it is almost impossible to tell whether there is a real difference of interpretation in either theory or practice between the writers, but it would seem reasonable from the context in which these terms are used that there is at least a difference of emphasis.

5 Discussion

Having considered the content of BPR, attention may now be turned to the three sets of questions identified in the introduction to this paper for which it was hoped to find answers. Is there only one type of BPR or are there several alternatives? What is the substantive content of BPR and how much of this is new? What factors may account for the interest shown in it? Each of these will now be considered in turn.

One BPR or many?

The analysis of the literature would seem to suggest that BPR is not a unitary concept (and it would have been perhaps more surprising to find that it was). However, if each approach is not to be classified as a separate version depending on the particular context in which it is applied, then the distinctive characteristics of BPR, or of certain recognised versions of it, need to be identified in order to develop an appropriate categorisation. This raises the problem of which approaches should be recognised as legitimate. While precedence might suggest that the approaches of Hammer [28] and Davenport & Short [14] should be seen as definitive this would seem unsustainable on practical grounds if nothing else. Thus, despite the efforts of CSC Index, the consultancy with which Hammer developed reengineering, to register "business reengineering" as a trademark, this cannot prevent other companies from selling approaches, which might be quite different from that envisaged by Hammer, under a slightly different name. Nor will it stop others from describing projects as reengineering, even if they bear little resemblance to Hammer's description. This is evident from the literature (see for example [2, 18]) where analyses of "BPR success" refer to widely varying initiatives, many of which did not even describe themselves as reengineering.

A second problem with categorising approaches concerns how closely they need to correspond to an archetype to be classified as belonging to a particular version (or even to be excluded from the concept altogether) and how this might be assessed. For example two approaches may share a strong emphasis on the role of IT (assuming there is some reliable measure of strength of emphasis), but differ on the importance of a definite methodology (which again may be very difficult to define). It would therefore seem difficult to provide a robust basis on which to classify different forms of BPR.

An alternative perspective is put forward by Davenport [13] who suggests that, rather than discrete "schools" of BPR, there is a continuum of approaches between two extremes of "process improvement", an incremental, continuous, bottom-up approach, and "process innovation" which is radical, IT-led, one-off and top-down. Talwar [62] suggests a similar spectrum, from "process improvement" though "process reengineering", "business reengineering" and "transformation" to "ongoing renewal". While this takes account of the observed differences between approaches, it assumes that they vary consistently along all dimensions. In practice, this would not appear to be the case. For example, Johannson *et al* [36] might appear to be a good example of Process Innovation with their emphasis on radical organisational change. However they clearly view their approach as originating with TQM, and see IT as only one among a number of transformatory factors.

It would seem therefore that it is likely to be impossible to agree a precise definition of BPR that does not exclude some of the approaches that have been associated with the concept, or to identify a limited number of distinct versions. It may thus be best to regard BPR simply as an umbrella term for a set of process-oriented approaches to significant organisational change.

What's new?

The limitations of such a broad, inclusive definition of BPR are obvious, however, when considering the possible novelty of the concept. For example, most of the literature acknowledges that process thinking has been at the heart of the "quality revolution" and Morris & Brandon [51] also state that the techniques on which they claim BPR is based (IE, time and motion, Operational Research and Systems analysis) "have all been concerned with processes for several decades".

If the core idea on which BPR is founded is shared with many other techniques, then what is the basis for claiming that it offers something significantly different from the others? What is to say that it is not simply a well-marketed repackaging of them? One of the key themes of a substantial proportion of the BPR literature which might therefore be a novel feature, is the contribution of IT. Certainly, the BPR literature typically lays considerably more emphasis on IT than a number of the concepts, such as TQM and "Excellence", that immediately preceded it.

The idea of IT as a "driver" of organisational change, however, goes back at least to Leavitt & Whisler [47]. Moreover, even if it is claimed that recent technological developments have made existing uses of IT more effective and provided opportunities for new and more significant interventions, the precise role that IT should play is a source of considerable disagreement amongst writers on the subject. It would thus seem an unsatisfactory basis for defining the unique character of BPR, particularly as many of its proponents are seeking to move away from the sort of technocentric viewpoint that such a definition would imply.

The remaining distinctive feature of BPR would therefore seem to be the scale of the change involved. As has been noted, however, there are some authors who do not seem to share the enthusiasm of Hammer & Champy [29] for "fundamental rethinking and radical redesign". Even if it were possible to exclude these faint-hearts from associating their approaches with BPR by some mechanism, though, this would still not resolve the problem of how the scale of change is to be measured and how the boundary between BPR and the other process-oriented techniques could be defined. For example, does BPR only describe approaches which seek change across organisational boundaries? If so, how many boundaries need to be crossed? All? More than one? Should change be measured by intent or by outcome? If the former, then BPR risks being discredited by approaches which promise more than they can deliver. If the latter, then a way is needed of separating out the contribution of the BPR activity to the outcome, from those due to other changes which may have been going on at the same time. For a variety of reasons therefore, it would appear that scale of change also does not provide a reliable means of defining BPR.

The final way in which BPR might be argued to be novel is its particular combination of process thinking, cre-

ative use of IT and radical organisational change. While there may be some practical value to this viewpoint in terms of enabling BPR writers to differentiate their approach from other techniques by emphasising some particular aspect of this combination as it suits their case, the analysis of the literature would seem to suggest that the degree of variation on each of these dimensions is so great that 'novelty' of BPR would vary from approach to approach.

This would seem to suggest, therefore, that despite BPR being widely hailed as a distinctive new contribution to management thinking (the endorsement of Hammer & Champy's book by no less an authority than Peter Drucker would suggest that this this view is not confined to just a few cranks), it seems very difficult to identify clearly what it is that makes it so significant. In large part, this problem stems from Hammer's definition of BPR in terms of practice. In the absence of any explicit, new theoretical principle on which the concept may be said to be based, Hammer requires BPR to be judged by the actions it gives rise to – if it achieves new forms of organisational behaviour which significantly improve organisational performance in some agreed way, then it is a significant new technique. As has been noted, however, this does not provide the basis for an unambiguous definition of the term. Moreover, as Grint [25] points out, all of the changes that Hammer & Champy [29] identify as characterising the "new world of work" in the reengineered organisation have been proposed by earlier writers. Thus even the changes in organisational behaviour associated with BPR may not be new either.

In the traditional interpretation of Kuhn's theory of scientific revolutions [44], therefore, it would seem difficult to argue that BPR constitutes a new paradigm, since it does not offer a new perspective on organisations which differs radically from TQM and other improvement programmes that preceded it. In Lakatos' terms [45], both appear to be operating within the same Scientific Research Programme. It could hardly be claimed, moreover, that TQM, for example, has not been extremely influential in changing management thinking and that BPR was thus in some way a conceptual Copernicus building on the unrecognised contribution of TQM's Tycho Brahe. Yet, if it is accepted that the immense interest shown in BPR is based on something more than self-delusion, then we need to provide some explanation of what this particular Emperor is wearing, even if it does not seem possible to define this in absolute terms.

To do so, however, would seem to require a reassessment of the nature of imperial regalia, to argue that the Emperor's old clothes were equally insubstantial. TQM, Taylorism or any of the other techniques which we may wish to consider as underlying earlier forms of organisational practice are/were, just as much as BPR, based on a set of socially-sustained beliefs which may or may not have had 'real' substance, but which depended on continuing faith in their efficacy. This is not simply a sleight of hand, but is based on the analysis of science and technology developed by Callon [9], Latour [46] and others which suggests that even in the so-called "hard" sciences, knowledge is not "discovered" in an external "reality", but is constructed through the recruitment of a network of allies who underwrite a particular viewpoint. Paradigm shifts therefore result from the ability of concepts to recruit the most influential allies. In this, rhetorical power may be rather more important than the "truth" of their views. The final question (why is BPR so popular?) may therefore help us to understand the allies that Hammer and Davenport & Short have, wittingly or otherwise, recruited and hence how BPR has come to have such an impact on current management thinking.

Why is BPR so popular?

The answer given by Hammer & Champy is that corporate America (*sic*) is facing a crisis, which they characterise in terms of three Cs – customers (who are taking charge), competition (which is intensifying) and change (which is becoming constant). Solutions are needed to enable companies to succeed in this new world of business. BPR, they argue, is the only technique which is capable of providing one.

Whether or not it is accepted that this crisis is real and enduring, for example Wood [69] questions similar claims made in relation to the flexibility debate, the concept that the business environment has recently become significantly more unstable and that substantial organisational change is unavoidable, appears to be widespread, particularly in the US, at present. BPR therefore appears to capture the spirit of the times in proposing that a totally new approach is needed. Such a viewpoint may also be seen as serving to legitimate the adoption of radical measures. If the company cannot hope to survive without complete re-organisation and the shedding of "armies of unproductive workers" [28] then these become a necessary price to pay. Thus, as Dixon *et al* [18] comment, "a crisis – real or perceived – may be necessary to create the conditions required to attempt a reengineering effort".

As Hammer & Champy observe [29], though, they are not the first to have diagnosed a crisis or to have suggested the way out of it. Many other techniques, from "management by objectives ... [to] ... one-minute managing" have been put forward in the past. None of them, however, "has reversed the continuing deterioration of America's corporate competitive performance". This is taken by Hammer & Champy as evidence that they were passing fads which "have only distracted managers from the real task at hand". The failure of these techniques is therefore used to bolster the case for BPR.

Paradoxically, these earlier techniques also help to sustain BPR. Thus, for example, the notion that new solutions have to be continuously invented provides a reason for looking to reengineering. The precedents for process-based techniques, even if they have not been particularly successful, also provide legitimation for the reengineering approach. In particular, the continuing influence of TQM, despite Hammer & Champy's dismissal of it, means that process-thinking may be seen as a distinct element in the network.

Hammer & Champy's particular focus on the prob-

lems of US companies may be seen as the recruitment of US patriotism as another ally. Thus they write that reengineering is "not another idea imported from Japan", but "capitalizes on the same characteristiscs that have traditionally made Americans such great innovators". Rather than "try to change the behaviour of American workers and managers" it "takes advantage of American talents and unleashes American ingenuity".

The success stories are another important element of the BPR network, serving both to affirm the potential of the technique, but also to associate it with the named corporations. Thus the 'hard facts' of the performance improvements cited by Hammer & Champy provide quantitative support for their claims which are thereby reinforced against sceptical questioning. Moreover the 'fact' that Ford, Bell Atlantic or Hallmark cards have successfully adopted reengineering provides powerful evidence of its value. Other case-study companies may be less well-known, but their inclusion shows that it is effective for all organisations not just those in the premier league.

Success stories also relate not just to the effectiveness of the technique, but to those who sell it. For example the near sixfold increase in the revenues of CSC Index in the last five years, is said to have "propelled [it] ... to the forefront of the [US management] consulting business" [8]. IDC [35] estimated a world market for BPR of $230 million in 1993 which they suggested was growing at 46% per annum.

These sorts of claims and figures draw in a variety of human actors to the reengineering network. Clients would seem likely to be attracted by the chance of order of magnitude performance improvements. As more leading companies adopt BPR, the need for others to join the bandwagon would seem likely to increase, both to show themselves to be up with the latest developments and to ensure that potential benefits are not lost.

For executives of the many companies which, in Hammer & Champy's words are "bloated, clumsy, rigid, sluggish, non-competitive, uncreative, inefficient, disdainful of their customer needs and losing money", the message that BPR provides the way out of this situation would seem a very seductive one. Hammer & Champy's insistence on the need for charismatic leadership of a firmly top-down process may also be expected to be attractive to those whose position this reinforces or who aspire to such roles. Similarly, those working in IS and/or Industrial Engineering may be expected to be attracted by the prominence given to their role in many versions of BPR.

For management consultancies looking for new opportunities, the size and relatively under- developed state of the BPR market (giving low entry barriers, few established players and significant scope for product differentiation) would seem to have considerable potential. The investment being made by the major consultancies in developing BPR products illustrates their recognition of this potential [63], but also serves as a signal of reengineering's importance to competitors and clients.

For IT consultancies BPR provides both an opportunity and a threat. If a suitable "front-end" product can be developed then there may be opportunities to capture market share in a new and rapidly-expanding area. Those who fail to offer BPR services, however, may find that part of their market has been sequestered both by traditional competitors who have taken the initiative and by new entrants from the management consultancy field. By getting in early on the BPR act, IT consultancies may also be able to strengthen the links between IT and BPR and thereby increase the size of their market.

The effectiveness of BPR as a management concept may also be due not just to the claims it makes, but to the way it makes them. As Jones [37] discusses, the violent language adopted by Michael Hammer, with its talk of "don't automate, obliterate", neutron bombs and grinding and frying out organisational fat gives the concept a strongly macho flavour. The apocalyptic rhetoric of organisational redemption through destruction of old habits and the adoption of pure, new processes also has biblical overtones. This is echoed in the comments of Stewart [59] who describes Hammer as "reengineering's John the Baptist, a tub-thumping preacher" and by Hammer himself who talked of reengineering as a "theology because it requires a belief that there is a new way of doing things ... it requires faith" (quoted in [7]).

Finally, it may be argued that process thinking itself simply has a considerable intuitive and practical appeal. Whether positive associations with the Quality movement are made explicitly or not, for individuals accustomed to a traditional functional perspective on organisations it can be seen to constitute a good example of an "interesting proposition" in terms of the classification scheme of Davis [16] . Thus it suggests that what had traditionally seemed to be a phenomenon that functioned effectively as a means to an end, ie the division of labour along functional lines to encourage the development of specialist expertise would ensure maximum organisational performance, was in reality a phenomenon that functions ineffectively.

6 Fad, Failure or Fatally Flawed – The Future of BPR?

This diverse network of allies recruited by BPR may help to account for its current position as "the hottest trend in management" [60]. Not surprisingly perhaps, it has also attracted many detractors. Thus Talwar [62] describes BPR "as the most fashionable and potentially the most detested management concept of the 90s". Many of these critics seek to show that BPR is simply a passing fad whose fall will be almost as meteoric as its rise. Holtham [34], for example, provides a timetable for the BPR lifecycle which predicts its demise in 1996-98. Those who dismiss BPR in this way generally adopt two lines of attack: arguing either that it is nothing more than a rebadging of existing concepts, or that its success owes more to skilful packaging than to any substantially new contribution to management thinking.

As has been discussed, the case for doubting BPR's novelty is fairly strong. The particular reasons offered by different authors for arguing that "we have seen it all before", however, often provide more insight on their own background than on the theoretical antecedents of BPR. Thus IS people see BPR as a reworking of systems analysis, industrial engineers as an extension of O&M and quality experts as a rebadging of TQM concepts. There can also sometimes appear to be an element of defensiveness in such arguments that suggests that the authors see this upstart concept as a threat to their expertise (and lucrative consultancy opportunities?).

Criticisms of the hype surrounding BPR also have some validity. As many authors note, the concept has clearly become one of the key management buzzwords of the 1990s and there is plenty of evidence of a bandwagon effect in operation. Stewart [60] even reports one telephone company executive as stating that "if you want to get something funded around here – anything, even a new chair for your office – call it reengineering on your request for expenditure". Such attitudes, however, do not necessarily prove that BPR is not a significant development. Thus, while it is possible to criticise the particular rhetorical style adopted by Michael Hammer, BPR is certainly not unique amongst management concepts in being promoted in such robust and evangelical terms. Indeed as Eccles & Nohria [20] have argued, effective use of rhetoric is an important aspect of successful change management. Moreover, as Hammer & Champy [29] are keen to argue, the proof of the reengineering pudding is in the eating.

This may help to explain why Hammer & Stanton [30] so vigorously contest what they see as the mythology of reengineering failure, arguing that the widely-cited figure of 70% was made up by Michael Hammer, based on experience with early reengineering projects. Now that the concept is much better understood, it is suggested, there is "little excuse" for companies to fail. If they do, it is because they fail to apply it correctly. This view is implicitly supported by a thriving sub-genre of the BPR literature which describes how to ensure reengineering success (see for example [4, 27]). Unfortunately, the empirical support for these recommendations, as with much of the writing on BPR, is generally weak and their prescriptions often amount to little more than 'be a successful company and do it right'.

Moreover, even if they do not fail completely, BPR initiatives may deliver less than is promised by their proponents. Thus Moad [49] reports that the benefits obtained from BPR projects intended to increase revenue were rated as achieving only 2 on a scale from 1 to 10, and even the best scored just 5 out of 10. James Champy is also reported as conceding that 30–40% of BPR initiatives "disappoint" [48]. A further interesting slant on this question is provided by Craig & Yetton [12] who point out that the rewards from an incremental improvement programme over a number of years may exceed those of a one-off radical change programme (and with less risk).

The claim made by Hammer & Stanton [30] that "the fault [for reengineering failure] lies not in reengineering, but in ourselves" may also be challenged on more substantial grounds. Thus, as Jones [38] discusses the BPR concept may be seen to have many internal contradictions. One of these is neatly illustrated by King [41] who notes that "the most problematic aspect of reengineering is that people are often asked to be creative so that their jobs (or those of their colleagues) can be eliminated or drastically changed". Similarly, Wheatley (quoted in [6]) argues that reengineering will inevitably fail as its top-down, mechanistic approach stifles learning and creativity.

Does this mean therefore that BPR is past its sell-by date and that forward-looking organisations should seek elsewhere for their inspiration? Eccles & Nohria [20] argue that the perceived failure of established management concepts is a necessary precursor of the emergence of new "solutions". The "open season" for BPR critics noted by Hammer & Stanton [30] may therefore be an ominous sign for the concept's longevity. The on-line search also indicates that there has been a significant decline in the number of BPR-related articles since mid-1994. However, despite a number of papers discussing what is "beyond" reengineering [15] or "after" it [50], the absence, as of early 1995, of any obvious successor that is capturing the management imagination, suggests that reports of its imminent demise may be premature. Moreover, unless organisations are to be caught up in a continuous cycle of change for change's sake, and are to learn nothing from past experience, then it is important that the valuable elements of BPR are not lost, even if the label itself becomes out-dated. It is hoped that the analysis provided in this paper, may help in identifying these elements.

References

1. J Aikins. 'Business process reengineering: where do knowledge-based systems fit'. *IEEE Expert*, **8**(3), (1993).
2. A Ascari, M Rock, and S Dutta. 'Reengineering and organizational change: lessons from a comparative analysis of company experiences'. *European Management Journal*, **13**(1), (1995).
3. P Bartram. *Business Reengineering:The Use of Process Redesign and IT in Transforming Corporate Performance*. Business Intelligence, London, 1992.
4. B J Bashein, M L Markus, and P Riley. 'Preconditions for BPR success: and how to prevent failure'. *Information Systems Management,*, **11**(2), (1994).
5. Booz-Allen and Hamilton. Business process redesign: an owner's guide, 1993.
6. T Brown. 'De-engineering the corporation'. *Industry Week*, (18 April 1994).
7. J A Byrne. 'Management's new gurus'. *Business Week*, (31 August 1992).
8. J A Byrne. 'Reengineering: beyond the buzzword'. *Business Week*, (24 May 1993).
9. M Callon, J Law, and A Rip, eds. *Mapping the dynam-*

ics of science and technology: sociology of science in the real world. Macmillan, Basingstoke, 1986.
10. A Cane. 'No guarantee of success - business process reengineering'. *Financial Times*, (26 September 1994).
11. C J Coulson-Thomas. 'Corporate transformation and business process reengineering'. *Executive Development*, **6**(1), (1993).
12. J F Craig and P W Yetton. 'The dual strategic and change role of IT: A critique of business process reengineering'. Working Paper 94-002, Australian Graduate School of Management, Kensington, NSW, (1994).
13. T H Davenport. *Process Innovation: Reengineering Work Through Information Technology.* Harvard Business School Press, Boston, Mass, 1993.
14. T H Davenport and J E Short. 'The new industrial engineering: information technology and business process redesign'. *Sloan Management Review*, (Summer 1990).
15. W H Davidson. 'Beyond re-engineering: the three phases of business transformation'. *IBM Systems Journal*, **32**:65–79, (1993).
16. M Davis. 'That's interesting! towards a phenomenology of sociology and a sociology of phenomenology'. *Philosophy of the Social Sciences*, **1**, (1971).
17. S F Dichter. 'The organization of the 90s'. *McKinsey Quarterly*, **1**, (1991).
18. J R Dixon, P Arnold, J Heineke, J S Kim, and P Mulligan. 'Business process reengineering: improving in new strategic directions'. *California Management Review*, **36**(4), (1994).
19. M Earl. 'The new and old of business process redesign'. *Journal of Strategic Information Systems*, **3**(1), (1994).
20. R G Eccles and N Nohria. *Beyond the Hype: Rediscovering the Essence of Management.* Harvard Business School Press, Boston, 1992.
21. Fintech. 'Business process redesign: IT suppliers take lesons from users'. *FinTech Electronic Office*, (9 September 1992).
22. Fintech. 'Rank Xerox to sell business process redesign consultancy and tools'. *FinTech Electronic Office*, (6 May 1992).
23. Fintech. 'Spotlight: Business process redesign brings big benefits'. *FinTech Electronic Office*, (2 December 1992).
24. C Grey and M N. 'Re-engineering organizations: a critical appraisal'. *Personnel Review*, **24**(1), (1995).
25. K Grint. 'Reengineering history: social resonances and business process reengineering'. *Organization*, **1**(1), (1995).
26. S Guha, W J Kettinger, and J T C Teng. 'Business process reengineering: building a comprehensive methodology'. *Information Systems Management*, (Summer 1993).
27. G Hall, J Rosenthal, and J Wade. 'How to make reengineering really work'. *Harvard Business Review*, **71**, (Nov-Dec 1993).
28. M Hammer. 'Reengineering work: don't automate, obliterate'. *Harvard Business Review*, **68**, (Jul-Aug 1990).
29. M Hammer and J Champy. *Reengineering the Corporation: a Manifesto for Business Revolution.* Nicholas Brealey, London, 1993.
30. M Hammer and S Stanton. 'No need for excuses'. *Financial Times*, (5 October 1994).
31. H J Harrington. *Business Process Improvement.* McGraw Hill, London, 1991.
32. R Heygate. 'Immoderate redesign'. *McKinsey Quarterly*, **1**, (1993).
33. R Heygate and G Brebach. 'Corporate reengineering'. *McKinsey Quarterly*, **2**, (1991).
34. C Holtham. 'Business process re-engineering: contrasting what it is with what it is not'. In *Business Process Re-engineering: Myth and Reality.* Kogan Page, London, (1994).
35. IDC. Business process redesign: the confluence of management and information technology consulting, 1992.
36. H J Johannson, P McHugh, A J Pendlebury, and W A Wheeler. *Business Process Reengineering: Breakpoint Strategies for Market Dominance.* John Wiley, Chichester, 1993.
37. M R Jones. 'Don't emancipate, exaggerate: rhetoric,reality and reengineering'. In R Baskerville, S Smithson, O Ngwenyama, and J I DeGross, eds., *Transforming Organizations with Information Technology*, Amsterdam, (1994). North Holland.
38. M R Jones. 'The contradictions of reengineering'. In G Burke and J Peppard, eds., *Examining Business Process Reengineering.* Kogan Page, London, (1995).
39. J M Juran. *Breakthrough Management.* Macmillan, London, 1964.
40. R B Kaplan and L Murdock. 'Core process redesign'. *McKinsey Quarterly*, **2**, (1991).
41. W R King. 'Process reengineering: the strategic dimension'. *Information Systems Management*, **11**(2), (1994).
42. M M Klein. 'IEs fill facilitator role in benchmarking operations to improve performance'. *Industrial Engineering*, (September 1993).
43. M M Klein. 'Reengineering methodologies and tools'. *Information Systems Management*, **11**(2), (1993).
44. T S Kuhn. *The Structure of Scientific Revolutions.* Chicago University Press, Chicago, 1961.
45. I Lakatos. 'Falsification and the methodology of scientific research programmes'. In I Lakatos and A Musgrave, eds., *Criticism and the Growth of Knowledge.* Cambridge University Press, Cambridge, (1970).
46. B Latour. *Science in Action.* Open University Press, Milton Keynes, 1987.
47. H J Leavitt and T L Whisler. 'Management in the 1980's'. *Harvard Business Review*, **36**, (Nov-Dec 1958).
48. C Lorenz. 'Putting reengineering in perspective'. *Fi-*

nancial Times, (21 October 1994).
49. J Moad. 'Does reengineering really work?'. *Datamation*, (1 August 1993).
50. J Moad. 'After reengineering: taking care of business'. *Datamation*, (15 October 1994).
51. D Morris and J Brandon. *Reengineering Your Business*. McGraw Hill, London, 1993.
52. E Mumford. 'New treatments or old remedies: is business process redesign really socio-technical design?'. *Journal of Strategic Information Systems*, **3**(4), (1994).
53. F Ostroff and D Smith. 'The horizontal organization'. *McKinsey Quarterly*, **1**, (1992).
54. J Parker. 'An ABC guide to business process reengineering'. *Industrial Engineering*, (May 1993).
55. P Rowland. 'The process for improving business'. *The Independent*, (22 June 1994).
56. M Scott-Morton, ed. *The Corporation of the 1990s - IT and Organizational Transformation*. Oxford University Press, Oxford, 1991.
57. J Shillingford. 'Offices follow the factory's example'. *Financial Times*, (19 March 1992).
58. J E Short and N Venkatraman. 'Beyond business process redesign: redefining baxter's business network'. *Sloan Management Review*, (Fall 1992).
59. T A Stewart. 'The search for the organisation of tomorrow'. *Fortune*, **125**(10), (1992).
60. T A Stewart. 'Reengineering: the hot new management tool'. *Fortune*, **128**(4), (1993).
61. D Sutherland. 'The reengineering lab'. *Insights Quarterly*, **4**(3), (1992).
62. R Talwar. 'Re-engineering – a wonder drug for the 90s'. In C Coulson-Thomas, ed., *Business Process Re-engineering: Myth and Reality*. Kogan Page, London, (1994).
63. J Thackray. 'Fads, fixes and fictions'. *Management Today*, (June 1993).
64. N Venkatraman. 'IT-induced business reconfiguration'. In M Scott-Morton, ed., *The Corporation of the 1990s - IT and Organizational Transformation*. Oxford University Press, (1991).
65. E Warren. 'Streamlining from the outside in'. *Informatics*, (May 1993).
66. D G Wastell, P White, and P Kawalek. 'A methodology for business process redesign: experience and issues'. *Journal of Strategic Information Systems*, **3**(1), (1994).
67. H Willmott. 'Business process reengineering and human resource management'. *Personnel Review*, **23**(3), (1994).
68. J P Womack, D T Jones, and D Roos. *The Machine That Changed the World*. Rawson Associates, New York, 1990.
69. S Wood. 'The transformation of work?'. In S Wood, ed., *The transformation of work?* Unwin Hyman, London, (1989).
70. R E Yates. 'Two separate tracks for re-engineering's engineers'. *Chicago Tribune*, (July 24 1994).

Received: June 1995

Editorial

What is a good contribution?

The work of refereeing is a thankless task that infringes on the limited time of those who are best in the field. Nevertheless, it is important since the quality of refereeing does significantly impact on the quality of the final product. In order to be fair to those limited and committed referees involved, I am of the opinion that we need to become more discerning in what we put before them. There is, of course, the important role of a journal (especially so with SACJ) to assist young researchers to develop their skills by getting feedback from competent reviewers in the field. However, one must not "abuse" this privilege by sending anything "just in case"—it is the academic responsibility of the authors to ensure that the paper submitted is of high standard.

What, then, is a good contribution? A good contribution has essentially three dimensions [1]: relevance, rigor and impact—of which relevance is the most important. Although these dimensions were formulated in the context of a discussion on Information Systems research, they may also have relevance for Computer Scientists. To assist potential authors in assessing their contributions (in the making), here are some questions, steps or criteria that should clearly and explicitly be addressed before submitting a paper to SACJ.

Relevance

- Who is the target audience of influence (IS mangers, system designers, educators, etc.)?
- What is the concern in the target audience that the paper addresses?
- Why is this a concern in the target audience?

Rigor

- What is the wider intellectual context of the study? Place it in this wider context.
- What other disciplines (or fields within the discipline) have studied or are concerned with the problem in the paper?
- What is the best research methodology for this problem domain?
- Why is the proposed research methodology selected?
- What question/problems may a referee raise and how are they being addressed?

Impact

- What contribution is the paper making? Is this clear and explicit in the conclusions?
- Why would a practitioner want to read the paper?
- What would a top scholar in the field say about the paper?
- Would a supervisor recommend the paper to his student(s)?
- Why can the journal not afford to publish the paper?

Clearly all of these questions may not always be equally relevant, and there may be others that could be relevant in assessing the contribution of a paper. Nevertheless, potential authors should rigorously critique their contributions and not merely "dump" material for publication. We owe it to ourselves and to the community at large.

News

Readers and contributors will be interested to hear of various plans for SACJ. These have evolved during a series of meetings held between editorial staff and available members of the editorial board and are briefly outlined below.

- Much time has been spent discussing whether or not SACJ should have a particular and/or exclusive focus. Should it seek a niche market, such as only publishing articles dealing with IT and development? Our deliberations have lead us to reject such an exclusive focus. Instead, SACJ will build on its traditional role of being primarily, but not exclusively, a platform for IT researchers in South Africa. In addition, the editorial staff intends widening horizons by actively recruiting contributions, primarily from other Southern African countries, but also from countries elsewhere in Africa. While SACJ will thus continue publishing a miscellany of articles that reflect the local status of IT research, it hopes to be a vehicle that encourages, stimulates and eventually reflects research on the African continent as a whole.
- SACJ specifically encourages contributions that synthesise existing research results, such as survey articles, reviews and taxonomies. Where such articles are of significant scope, they are important for research. They will therefore be placed in the research section of the journal and may, in consequence, be submitted to the Department of National Education (DNE) for subsidy purposes.
- In order to sharply differentiate the research section from the non-research section of SACJ, page numbers in the Communications and Reports section will henceforth be preceded by an A. This is intended to signal that articles in the Communications and Reports section should not be submitted to the DNE for subsidy purposes.
- There have been strong pleas that the journal be brought out on a more regular basis. To date, issues have been held back until a fairly sizeable number of articles are available for publication. This has resulted in two to

three issues per year. In future we intend bringing out four issues, two of which will be regular issues, and the other two will be special issues dealing with some topical theme. This process has already started in that the next issue (in October) will be on IT and Development. A further special issue on Networking/Telecommunications is planned for mid-1996. Articles for the former special issue are currently being reviewed. Potential contributors to the latter special issue might want to start thinking about possible publications, but should delay submission of manuscripts until further details are provided in a formal call for papers.

- To date, SACJ has not obliged contributors to sign away copyright. Although the matter of copyright has become somewhat obscured in the age of electronic documents, the Internet, WWW, etc., SACJ will in future require that contributors sign a copyright form before their work is published. In dealing with issues relating to electronically available material, we shall follow guidelines proposed in recent issues of the Communications of the ACM.
- Some readers may have noticed that SACJ now appears on the World-Wide Web. This is due to the efforts of the production editor. The WWW home page address is provided on the front inside cover of the journal. Extensive information to potential subscribers and contributors has been provided. This includes information on submission procedures, document preparation, etc. Abstracts of published articles are also given.
- Presently SACJ is sent to Inspec where selected articles are indexed. Arrangements are currently being made for Science Citation Index to provide a similar service. This should significantly increase exposure of SACJ's contents to international researchers.
- SACJ encourages contributions from young and/or inexperienced researchers and will assist them in various ways to attain the quality required by the journal. For example, they may be provided with additional editorial assistance (perhaps to be paid for, if warranted by the scope of the assistance). Referees will be urged to be encouraging in their feedback and to formulate feedback clearly and specifically in ways that will facilitate the revision process. Potential contributors may also request to be put in touch with a mentor to advise in planning and writing up research. As a further encouragement to young researchers, consideration is being given to an annual citation for the best student contribution.

Derrick Kourie and Lucas Introna
Editors

1. Keen, Peter,G.W. 'Relevance and Rigor in Information Systems Research: Improving Quality, Confidence, Cohesion and Impact' in Nissen *et al*, eds., *Information Systems Research: Contemporary Approaches and Emergent Traditions*. IFIP 8.2, 1991.

SACJ is produced with kind support from
Mosaic Software (Pty) Ltd.

Parallel Distributed Objects' Structure and Their Synchronization in an APPC Environment

Dr. Valentin Kisimov* Ivone da Silva†
*Computer Science Department, University of the Witwatersrand
†ITS Projects and Consulting, Liberty Life

Abstract

Over a period of a few hours, every computer participating in a distributed system has many unused resources. If these resources can be combined, a large amount of computational power can be generated. This paper offers an object oriented approach for using this power and organizing an efficient work scheme, compatible with the efficiency of multiprocessor systems. A special structure for an object is described, with the help of which the work of a parallel concurrent object is ensured. Since a major problem in parallel systems is that of synchronization, synchronization methods and algorithms are offered to support the execution of parallel objects. The research and the proposed development are based on a common and popular application programming interface – APPC (Advanced Program to Program Communication). Experimental results show the efficiency of the approach.

Keywords: *Distributed systems, synchronization mechanisms in distributed systems, distributed objects, concurrent programming, distributed databases*
Computing Review Categories: *C.2.2, C.2.4, D.1.5., D.2.2, D.4, H.4.3*

1 Introduction

Modern-style, complex systems cannot sensibly be built without design tools, code generators and testing tools and without supporting object-oriented tools. While software engineering tools are based on a set of existing methodologies, object- oriented tools are now being introduced which have low level features, such as class libraries. At the same time the computing community is going to connect more computers together to form networks, thereby organizing distributed systems. During a 24 hour period, these systems accumulate a considerable amount of unusable computational power (especially in the case of a LAN having multitasking workstations). The distributed systems and particularly the distributed object applications can be a good basis for using this power by locating different numbers of objects in the computers and dynamically migrating these objects, depending on the current loading of each computer in the distributed system. In this way, every application will use the combined computational power of the distributed system more effectively. This organization of work for concurrent objects needs supporting predicates, the most important of which is synchronization.

In distributed systems, synchronization can be defined as an operation in which two or more processes (objects) exchange information (messages) to coordinate their concurrent activities. This definition is based on the message exchange principles and not on shared memory. The use of synchronization methods and algorithms, developed for centralized systems in a distributed system is not a trivial process, because of resource distribution, lack of global state information and communication delay. The following are the main approaches used to construct distributed synchronization algorithms [11]:

- Lamport's event-ordering,
- token passing, and
- priority-based event-ordering.

All of these are oriented to serving the two synchronization methods – event-ordering (mainly partial event-ordering) and mutual exclusion synchronization. The main event-ordering algorithms for synchronization – distributed logical time [10] and Lamport's distributed timestamp (and even its last evolutions [3, 18]) show the need for active message exchange in these types of algorithms. For distributed systems these algorithms are intensive network consumers and for small or middle speed networks (up to 10 Mbps) their execution is slow. For these reasons, event-ordering algorithms will not be considered in this paper. Mutual exclusion synchronization needs less communication activities so this paper presents a modification of this algorithm according to the chosen distributed environment.

The method of synchronization for concurrent execution of processes in distributed systems has not been sufficiently investigated nor evolved. Another synchronization method for distributed systems, which is not well developed is synchronization with stored distributed data, which is a modification of centralized synchronization algorithms for ensuring efficient distributed synchronization. Independent of the wide spread communication between different computers types, logically every distributed system is oriented to bind homogenous computers. From the network point of view, if a computer executes the functions of the 7-th layer of ISO/OSI Protocol Reference Model, this computer is accepted as an open system. From the distributed application point of view, the realization of the open system is not enough, because every distributed application interrelates to a concrete operating system, programming language's compiler, database management system and net-

work application programming interfaces. That is why it is felt that for the distributed applications, another term – *common system* – has to be defined, in which the design requirements for independence of hardware, operating systems, network protocols, databases and transaction processing monitors are satisfied. Naturally, an open system is a step towards common systems, but away from the realization of full common systems.

For the current development of distributed applications there is a demand for a common standardized programming language, common application programming interface for the 7-th layer of ISO/OSI Protocol Reference Model (application layer) and common application programming interface for databases. As a standardized programming language C (with all vendors' differences) can be used and as an application programming interface for databases – SQL language can be used. Presently the APPC Interface is imposed on the world as one of the biggest application programming interfaces for the 7-th layer ISO/OSI Protocol Reference Model. Created initially by IBM, APPC is supported by a several different computer architectures – HP, DEC, Sun, covering a large range of computers from PCs, through middle- range computers to mainframes. At a time when there are few good distributed operating systems, APPC is a platform on which sufficiently efficient distributed applications can be developed.

The purpose of this paper is to introduce the structure of an object which gives the possibility for distributed parallel execution in an APPC environment and offers synchronization algorithms for this type of work. All the proposed synchronization primitives are organized as user-level threads, which have, according to Anderson *et al* [4], better performance than kernel threads. The article gives experimental results and conclusions for the use of the developed algorithms.

There are four approaches for implementing network application programming interfaces:

- the application programming interface is part of a general-purpose language;
- the program code for the application programming interface is automatically derived from a protocol specification, expressed using a formal description technique;
- the application programming interface is realized as a callable based language extension, and
- the application programming interface is a macro-language extension.

The first three approaches are explained in [1] and the fourth one is presented in [20] and it is used in the synchronization proposed herein. The built-in application programming interface in a general-purpose language is the most efficient way for using the application programming interface, but it is linked to the current operating system. Formal description techniques for protocol specification such as Estelle, LOTOS or SDL can be used for automatic generation of code, but in their current state these technologies have not lived up to promise. The third approach lies between these two, having low level optimization and relative independence from the operating system. The fourth approach, used here, is an evolution of the third approach in the direction of increased functionality. The macro- language extension offers the possibility to create complex primitives with interrelationships between them. For the present distributed object synchronization, the main primitives have to comprise: access to network layers, access to Distributed Database Management System and a few logical blocks, girding the whole primitive functions. These macro extensions assume that the user program is written in C and will generate a C code extension. The macro extension is a well known technique and only the logic of each synchronization primitive will be emphasized here – how it works, but not how it will be included in a user program.

A specific aspect of this approach is that application types of means (primitives from application programming interface) are used for realization of system primitives (synchronization primitives). Initial research on the architecture of distributed objects has been done from the project, called "Distributed Parallel Object Architecture", undertaken in the Computer Science Department, Wits University [14]. The hardware-software platform chosen for the experimentation comprised PCs 386 DX/40 with OS/2 2.x, linked in a Ethernet LAN, managed by the IBM Communication Manager (a product in OS/2 line) and using OS/2 Database Manager. The execution of higher level protocol software in distributed systems is important from a performance point of view, because it can become the bottleneck of the system [17]. For this reason, if distributed algorithms with high level network application programming interfaces are effective for a low performance level distributed systems, they will be valid with more power for higher performance distributed systems. That is why all the experiments described herein were conducted in a low performance level distributed system – PCs (386 DX/40), linked in an unsophisticated LAN – Ethernet.

2 Object Structure in an APPC Environment

Concurrency is an element of the object model [5] and in recent years researchers have been engaged in the realization of concurrency in distributed object systems [2]. Most of these approaches are oriented towards using homogeneous computers and not fare or a common environment, supported by different computer architectures, such as APPC.

In principle the object program consists of a hierarchy of objects and object program code as illustrated in Figure 1. The hierarchy of objects is a collection of objects, each one representing an instance of some class in which a hierarchical structure with inheritance exists between classes. The object program is a set of messages, each one of which is intended for one object, e.g. the program is a single set of messages. A special architecture was developed to organize the environment for the existence and parallel execution of distributed objects – distributed parallel object architecture (DPOA). This architecture includes the specially designed structure of an object and an environment

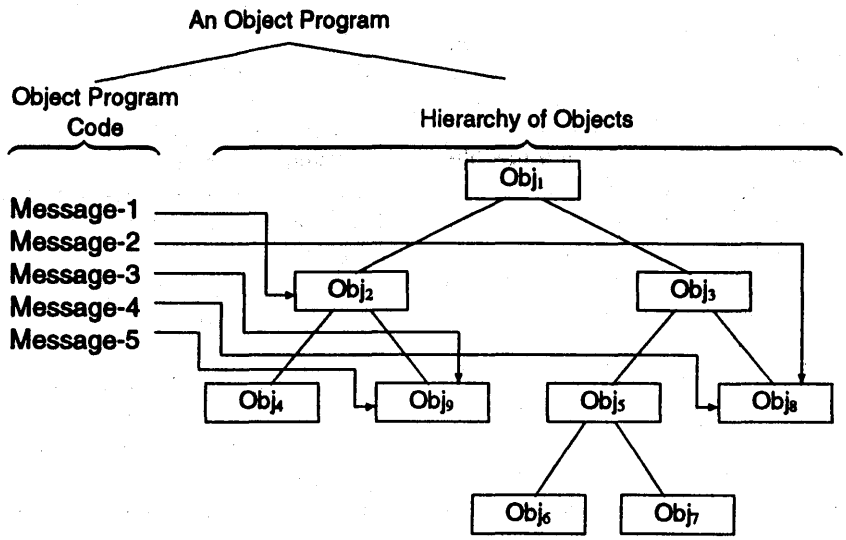

Figure 1. An object program as a hierarchy of objects and code

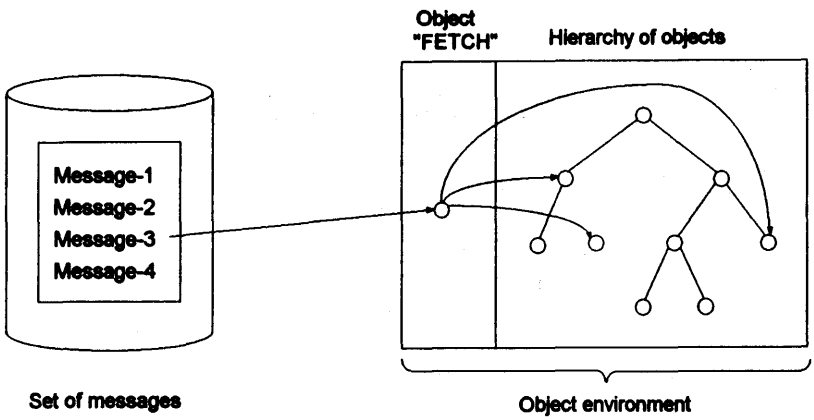

Figure 2. The operation of the FETCH object

in which the object program is running. In DPOA, the execution of an object program is organized by using a special object, called FETCH, which reads the messages from the set of messages and sends every message to the target object (Figure 2).

In general, the object environment is a virtual space for the existence of object, for message passing between them and for running the objects' methods. Every multitasking programming environment with an APPC application programming interface can be accepted as a part of a whole distributed object environment, because one or a few objects (tasks) can run simultaneously in each computer and the message exchange between objects is transparent to object location. This is true, because every object is assigned a separate LU name (Logical Unit – a term from SNA [16], identifying a distributed program piece as an independent entity). APPC has a feature to deliver a message to an object (LU name) regardless of the location of the receiving object – be it in the same computer with the sender object or in another computer in the distributed system. That is why all computers in the distributed system with multitasking and APPC form one common distributed object environment. In DPOA the chosen distributed database (SQL oriented) is a place for storing the set of messages. The object FETCH can read the set of messages independently of the computer s location and irrespective of which computer's disk is used to store them.

From the classical object approach, a message consists of the object receiver's name, the name of the method, which has to be executed and finally argument-data for the method. The following designation for a message will be used:

$Object_i$: $Method_k$(Arg-$list_k$)

In DPOA an evolved structure of the messages is chosen, giving the possibility to specify not only one method, but several methods for execution in an object:

$Object_i$: $Method_1$(Arg-$list_1$); . . . , $Method_n$(Arg-$list_n$)

These methods can be executed in sequence or in parallel. This article does not emphasize the parallelism inside the object, because that type of parallel execution can be synchronized with centralized methods and algorithms.

Figure 3 shows a global structure of an object in DPOA. It consists of 4 parts – the message receiver (MR), the parser of received messages (PRM), the object repository (OR) and the object bodies (OB). The message receiver accepts the incoming messages, checks them, transforms them from form (2) to (1) and stores them in the object repository. After that the message receiver activates the object body to execute the messages, collected in the object repository. All stored messages form the set of messages. The object repository uses a distributed relational database management system for storing the information in tables. Each object has access to its tables. But for service purposes, these tables can be accessed by other objects, according to predefined access rights. The service mechanisms in DPOA can create service messages for each object (the program messages are received from the message receiver and stored into the object repository, whereas the service messages are stored directly into the object repository).

One of the purposes of the service messages is to help the synchronization between objects. The PRM has only service functions - to analyze the stored messages and activate (or not) the object repository, according to the messages and the state of the object. The object has flags, stored in the object repository. These flags define the state of the object. The most important of these flags are:

- Active Body/Deactive Body (AB/DAB);
- Empty set of messages/Non Empty set of messages (ESM/NESM);
- WAIT/NOTWAIT.

The object's body can be logically linked with the classical concepts for an object – it has methods and data, but reads messages from the object repository instead to receive them. This is equivalent for the object's body that stores message in the object repository. The logic of the work of these object's blocks can be presented by the following outline:

```
Message Receiver (MR)
    Analyzes and stores the messages;
    Sets NEMS;
    if DAB
        Sets AB;
        Activates the Object's Body;
        End;
    else /*AB*/
        End;

Parser of Received Messages (PRM)
    if ESM
        End;
    else /*NESM*/
        Sets AB;
        Activates the Objects' Body;
        End;

Objects' Body (OB)
/* Activated from Message Receiver or Parser of
    Received Messages*/
    if WAIT
        Sets DAB;
        Deactivates the Object's Body
        End;
    else /*NOTWAIT*/
        if ESM
            Sets DAB;
            Deactivates the Object's Body;
            End;
        else /*NOTWAIT+NESM*/
            Reads a message;
            Treats the message;
            Go to the beginning of the Object
              Body's execution;
            End;
```

The object's components – message receiver, PRM and object body – are separate APPC programs, however the object's body is always called from one of the other two.

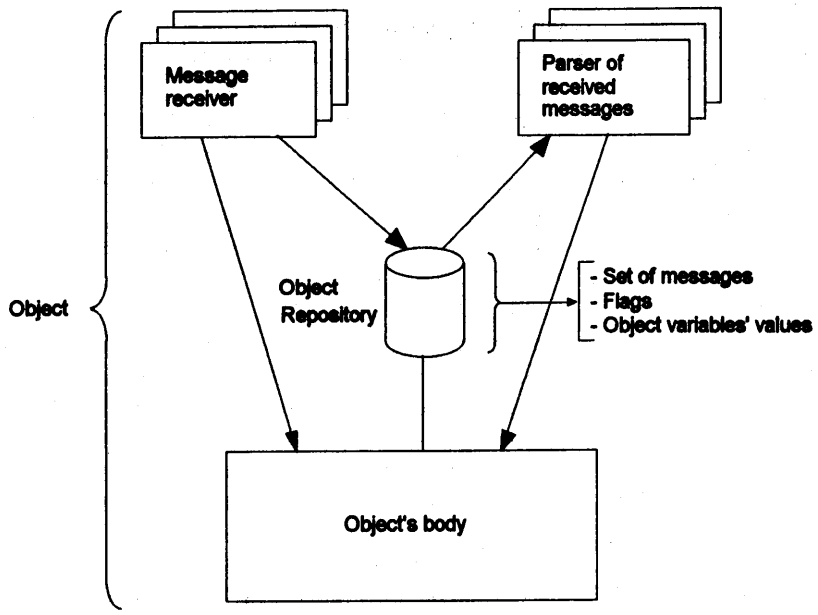

Figure 3. Global structure of DPOA object

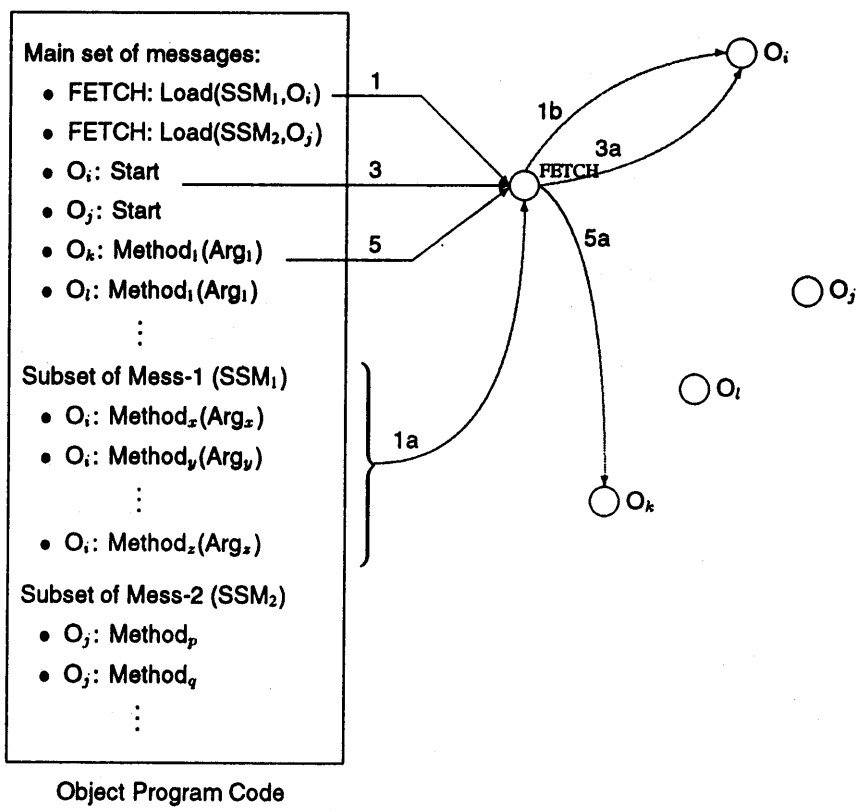

Figure 4. FETCH operation

The method of activation of the object's body is as follows (the APPC verbs used in the figure are explained in [16]):

```
/* Activation of the OB */
TP_STARTED
ALLOCATED                  /* Starting the OB */
FLUSH ------------------> RECEIVE_ALLOCATE
                           :
                           TP_ENDED
                           /* Deactivated body */
```

The object FETCH has another function – to take a subset of messages (SSM) from the object program code and to load it into the object repository of another object. After that it can send a message to that object with a method START, which will insure the execution of this subset. A similar case is shown in Figure 4, in which the object program code has 3 parts – a main set of messages and two subsets. With the first message from the main set, the object FETCH loads the subset SSM_1 into the object repository of the object O_i and with the third message of the main set, the object O_i is started to execute its own messages, loaded into the object repository. A similar case exists with the second and fourth main messages, but oriented to the object O_j.

The object program code having got no subsets of messages is called a single message set. When the program code has subsets of messages, the code is called a multiple message set. In the case of multiple message set parallel, object execution exists – in a fixed period of time one object executes a message from the main set and at least one other object executes its subset of messages. For a multiple message set several objects, execute their own subsets of messages, which may be equal or different (one subset for a few objects or n subsets of messages for n objects). That's why DPOA provides the following types of object execution:

a) Pure sequencial execution, in which the object FETCH reads message by message from the main set of messages and every message contains one specified method;
b) Complex sequencial execution, in which the object FETCH reads message by message, every message can contains a few specified methods, but these methods are executed in sequence from the target object, without concurrency inside the object;
c) Partial parallel execution, in which the object FETCH reads message by message from the main set of messages, every message can contain a few specified methods, but the target object executes in parallel all or a part of these methods;
d) Parallel execution in a single message set, in which the object FETCH reads message by message, one or a few of these messages require object FETCH to load an SSM into one or a few target objects and after that those target objects will execute their equal subsets (object programs) in parallel;
e) Parallel execution in a multiple message set, in which the object FETCH reads message by message, a few of these messages require object FETCH to load different SSMs into target objects and after that those target objects will execute their different subsets (object programs) in parallel;
f) Comprehensive parallel execution in single message set the same as d), but with the features of c) as well;
g) Comprehensive parallel execution in a multiple message set the same as e), but with the features of c) as well.

It must be mentioned that a multitasking technique is required for c), f) and g), which doesn't exist in an APPC and can be achieved only with the particular operating system's features. Every multitasking realization can be different, depending on the type of the operating system. In DPOA our wish was to have a common approach, without reflecting the features of the different operating systems. For this reason the types of object execution c), f) and g) will be not considered further herein.

3 Mutual Exclusion Synchronization with Token Passing

Based on Tanenbaum [19], and giving an account of the distributed system principles, the following definition for *atomic action* is acceptable – an indivisible action from one object (from one computer), done as a single action. Couching this definition in programming terms, yields the term *critical section*, which according to Burns and Davis [6] is a piece of code that must appear as an atomic action. If two objects contain critical sections whose overlapped executions could interfere with one another, then we need to ensure that those sections are executed under *mutual exclusion*. So, mutual exclusion ensures that the critical section from an object cannot be interrupted and it appears as an atomic section from the other objects' point of view. The support of the mutual exclusion mechanism cannot be done from the internal object structure. It requires outside synchronization mechanisms.

In [15] a method is proposed for mutual exclusion on the basis of the token approach. The idea is to circulate a unique message, called a token, having a specific format. Here this idea is evolved with the following premises:

- in an object system, the Critical Section can be a method, a part of a method or a sequence of execution of methods, belonging to one object and the designer of the object is able to define the critical section in every object;
- according to the object-approach, every action needs to be encapsulated in an object and for this the action – synchronization has to be shaped in a separate, service object;
- the logical ring is created every time when an object or a few objects need to execute critical sections (see Figure 5) and the arbitrary sequence of these objects organizes the logical ring;
- a special service object has to collect all requirements for critical sections and has to support the logical ring;
- if, from the application point of view, a lot of critical

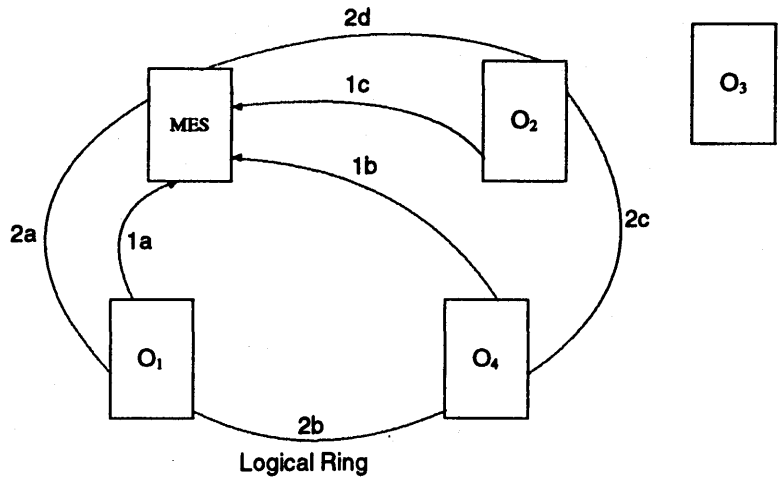

Figure 5. Mutual exclusion synchronization

sections are defined, these sections can be classified into separate classes, so that a separate logical ring can be established for each class.

In DPOA a service object, MES, is defined for the whole mutual exclusion synchronization. It accepts the requirements for mutual exclusion (arrows marked with 1 in Figure 5) creates the logical ring (arrows marked with 2 in Figure 5) and starts the empty token. The objects executing the application will be called *application objects*. Their purpose is to achieve synchronization for mutual exclusion for the execution of their critical sections. When an application object receives an empty token, it puts its own object id (O_{id}) in the token and send the token in the ring. After the circulation of the token, when the token arrives with O_{id} equal to its own id, the object executes its critical section. A more detailed algorithm for the work of the application object is given in Figure 6. The time for sending

```
Every object Oj receiving the token:
    if the token is empty
        Fills the token with its own Oid;
        Sends the token to the next in the Logical ring;
        End;
    else /*the token is not empty*/
        if the Oid in the token is not = to its Oid
            Sends the unchanged token to the next in
             the Logical ring;
            End;
        if the Oid in the token is = to its own Oid
            Executes its Critical Section;
            Creates an empty token;
            Sends the token to the next in the Logical
             Ring;
            End.
```

Figure 6. Mutual exclusion synchronisation

the message of requirements for mutual exclusion is equal to the time of sending a message from one object, because the other objects either sent their requirements before this time (when the object MES was engaged with other work), or they sent their requirements in parallel with this object. N messages are required to establish the ring, where N is the number of application objects which need mutual exclusion in a fixed period of time. With this, the number of messages for the token ring creation, m_{TE} is:

$$m_{TE} = N + 1$$

The time t_{O_i} that the i-th object spends waiting until it starts its critical section is described by:

$$\begin{aligned} t_{O_1} &= t_{TE} + t_{TP} = t_{SM}.m_{TE} + t_{TP} \\ &= t_{SM}.(N+1) + t_{TP} \qquad (1) \\ t_{O_i, i>1} &= t_{O_1} + \sum_{i=2}^{N}((N+2).t_{TP} + t_{CS_{i-1}}) \qquad (2) \end{aligned}$$

where:

- t_{TE} is the time to establish the token ring;
- t_{TP} is the time for passing the token from object to object in the ring;
- t_{SM} is the time for sending a message;
- t_{CS_j} is the execution time of the critical section in the j-th object.

The mathematical expressions 1 and 2 are valid with the assumption that all objects have the same t_{SM} and t_{TP}. If we have to include the differences between these times for each object, the expressions expressions 1 and 2 will be very complex. From the APPC point of view, t_{SM} and t_{TP} are different as illustrated in Figure 7.

The experimental results from the synchronization and execution of critical sections for N objects, when $N = 3, 5$ and 10 are given in Table 1. The experiment is based on the following assumptions:

- all objects are in different computers;
- each computer runs only one object – the object that will execute its critical section;
- each critical section requires 2 seconds execution time.

From the preliminary results it can be seen that for N=10, the synchronization process can be organized not as one ring with ten objects, but as five sequence cycles, in each

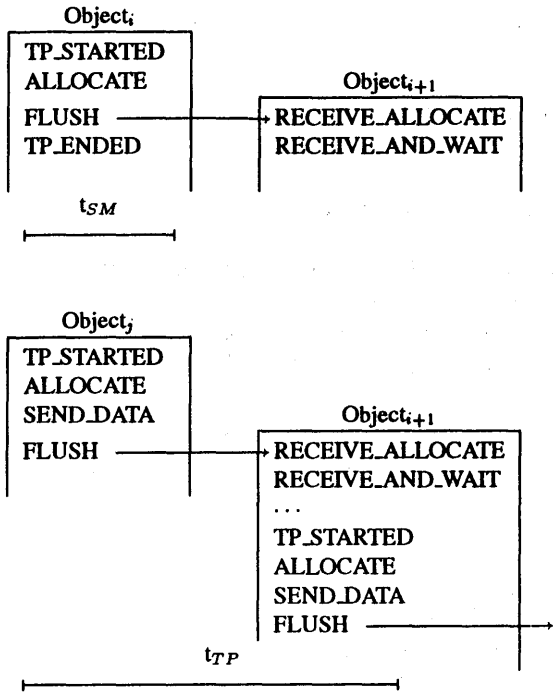

Figure 7. The difference between t_{SM} and t_{TP}

Table 1. Experimental results from the synchronization and execution of critical sections for N objects

	N				
Time	1	2	3	5	10
t_{O_i} (N=10)	12.0s	24.8s	37.6s	63.2s	127.2s
t_{O_i} (N=5)	6.5s	14.8s	23.1s	39.7s	
t_{O_i} (N=3)	5.5s	12.9s	20.5s		

one of them there will be a logical ring between only two objects, i.e. in the first cycle the ring will obtain O_1 and O_2, in the second ring – O_3 and O_4 etc. In this case the total time for ten object synchronizations will halve. Further tests have been done on finding the optimal number of objects in a cycle (one logical ring) and yielded a number between one and two. That is why the algorithm in the service object MES was reconstructed to establish a logical ring taking two objects from the total number of simultaneous required objects for synchronization. After that the next ring will take the next two objects etc. until all objects have been serviced.

4 Synchronization with Shared Distributed Data

Many of synchronization principles and algorithms exist for multiprocessor centralized systems [19, 6]. All of them are well developed, but designed for the use of centralized common data-variables that can be shared by all processors in the system. In distributed systems, distributed shared data can be used for the purpose of synchronization in the form of a distributed shared dataspace or distributed database. Typical examples of a distributed shared dataspace are LINDA, formally described in [7] and the service space, presented in [21]. The main principles of the distributed shared dataspace are based on an asynchronous user servicing, which is not convenient for synchronization of parallel, concurrent working processes or objects. The distributed database can be used with success as distributed shared data, because it combines: the synchronous user servicing; the management of multiple simultaneous user access and the use of heterogeneous media for storing the distributed data. Until recently, a distributed database would not be used for synchronization purposes because of the impression that it works slowly. In this research a distributed database was used as distributed shared data and to show the acceptability of this approach the distributed database was applied on a slow computer – a PC 386. In the DPOA the approach of using a distributed database for the organization of common data is accepted. The current state of the distributed database offers relatively good performance and availability over different platforms from workstations to mainframes. In addition, distributed databases have builtin features for multiuser access management and well developed locking mechanisms. That is why it was decided to use a table from a distributed database as shared data, called *shared distributed data*. In DPOA this shared distributed data is implemented as semaphores. Dijkstra's semaphores [8] and related operations are applied with modifications, according to the APPC possibilities and the structure of the distributed presented above. The purpose of synchronization with semaphores is to ensure mutual exclusion of the execution of critical sections.

It is perceived that a semaphore is detached for every critical section type. Each semaphore consists of flags,

whose number is equal to the number of the objects participating in the appropriate critical section's type. For the realization of each semaphore, a row of a the database table was chosen, in which every flag is a separate short integer item. A queue is attached to each semaphore, in which temporary flag-ids are collected for all the objects wishing to start their critical sections, but which cannot because at the same time previously started critical sections are executed. Two service methods are defined for application objects for this type of synchronization – Test-and-Set-Flag (TAS) and Reset-Flag (RS). In the message set, a message with the method TAS marks the beginning of the critical section and a message with RS, the end of the critical section:

Message-Set$_k$	*Message-Set$_l$*
⋮	⋮
O_k: M_p	O_l: M_x
O_k: TAS	O_l: TAS
O_k: M_q	O_l: M_y
⋮	⋮
O_k: M_r	O_l: M_z
O_k: RS	O_l: RS

A special service object, FLAGS, is created for management of synchronization with stored distributed data. Figure 8 gives the algorithms for application methods TAS and RS and for the service methods SvTAS and SvTR, supporting the appropriate application methods. The realization of the method SvTAS and SvRS applies to the database for storing the flags and QUEUEs. This database need not be distributed, because only object FLAGS uses it.

The experiment showed that the time for successful execution of TAS is 950ms and for RS averages 1000ms. The unsuccessful execution of TAS can be indefinitely long and it depends on the number of objects, separated for one type of semaphore. The optimistic time for permission of critical section's execution is 1.9s (no other critical sections have been executed in that time).

Application methods:

TAS
generates a message:
FLAGS: SvTAS(Semaphore$_n$, Fl$_i$, O$_i$)

RS
generates a message
FLAGS: SvRS(Semaphore$_n$, Fl$_i$, O$_i$)

Service object FLAG's methods:

SvTAS:
Reads and locks the whole semaphore's flags;
if all flags are not = 0
Stores *(Semaphore$_n$, FL$_i$, O$_i$)* in the QUEUE;
Sets Object's O$_i$ flag WAIT := 1;
End;
else /* all flags are = 0 */
Sets FL$_i$:= 1;
Activates PRM$_i$;
End;

SvRS:
sets F$_i$:= 0;
if QUEUE is empty
Activates PRM$_i$;
End
else /* the QUEUE is not empty */
Reads F$_i$ and O$_i$ from the QUEUE;
Sets FL$_i$:= 1;
Activates PRM$_i$;
Activates PRM$_j$ from O$_j$;
End.

Figure 8. Methods for the FLAGS service object

5 Synchronization for Concurrent Execution

In the last few years Computer Supported Cooperative Work has started to become more important for distributed systems [12]. It utilizes networking, communications, concurrent processing and window environment. In this type of distribution, the concurrent work is a mechanism with a great influence. The synchronization is a crucial point for parallel, concurrent working processes. That is why this article emphasizes the synchronization of concurrently executing objects. The method of synchronization developed here, called synchronization for concurrent execution, serves the parallel concurrent object execution in an APPC environment. It is based on barrier synchronization [6]. Two primitives are defined for the method. The primitive DOPAR (derived from do in parallel) is defined to start the execution of a parallel object. The primitive RV (from *rendez-vous*) is defined to ensure that the object will wait until all parallel executed objects reach a common point –

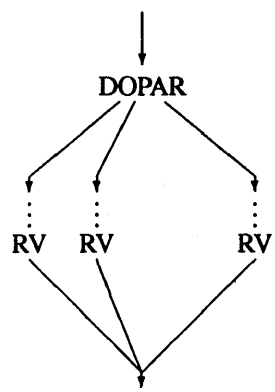

Figure 9. The primitives DOPAR and RV

DOPAR/RV: CMSCO((O_1,...,O_n),(Pm/m_j),(Pd),(WORV/RV))
DOPAR/RV: CMSCO((O_1,...,O_n),(Pm/m_j),(Pd_1,...,Pd_n),(WORV/RV))
DOPAR/RV: MMSCO((O_1,...,O_n),(Pm_1,...,Pm_n),(Pd),(WORV/RV))
DOPAR/RV: MMSCO((O_1,...,O_n),(Pm_1,...,Pm_n),(Pd_1,...,Pd_n),(WORV/RV))

where:

Pm_i	is a pointer to a message subset of the i-th object;
m_j	is a the name of the method that the i-th object has to run;
Pd_i	is a pointer to the data area for the i-th object;
O_i	is an object-id for an object which will run in parallel;
WORV	means that the current message subset will be executed in parallel, but without a rendezvous mechanism;
RV	means that the current message subset will be executed with subsequent using of rendezvous mechanism.

Figure 10. Syntax of messages sent to DOPAR/RV

see Figure 9.

As mentioned above, in the presented object architecture for parallel distributed execution, there are two types of object program code – single message set and multiple message set. It was stressed as well, that the multiple message met offers a possibility for parallel object execution of the main message set and one or a few message subsets. That means, one object (FETCH) executes the main message set and one or more other objects (called concurrent objects) execute one or more message subsets in parallel. The concurrent objects can use one message subset for all of their execution and in this case we will have a common message subset for the concurrent objects (CMSCO). If the concurrent working objects use different message subsets for their works, we will have multiple message subsets for concurrent objects (MMSCO). Normally, each object can work with some external data. This data can be common for all concurrent objects – single data (SD) or different data can exist for these concurrent objects – multiple data (MD). Hence, we can define the following 4 categories of concurrent execution of distributed objects:

- CMSCO+SD;
- CMSCO+MD;
- MMSCO+SD;
- MMSCO+MD.

Parallel object execution has to distinguish these four categories. For this, a service object was developed (called DOPAR/RV) with two DOPAR type methods: CMSCO and MMSCO. The syntax of the message, submitted to the service object DOPAR/RV, determines the type of data that will be used - single data or multiple data. Normally, the object FETCH sends this message to DOPAR/RV, because the concurrent parallel object work is organised from the main message set. But every application object can send this message as well, organizing sublayers of parallel object execution. Figure 10 gives the syntax forms of messages, sent to DOPAR/RV, defining these four categories of work.

The using of RV defines a common point after the concurrent work, whereas WORV ensures that the parallel working objects will not be further synchronized for common activities.

If an object wants to start a parallel object execution it has to execute its own DOPAR method, which sends a message with a syntax matching one of those showed above. The generation of this type of message plus the working of the method CMSCO/MMSCO in DOPAR/RV, realize the primitive DOPAR.

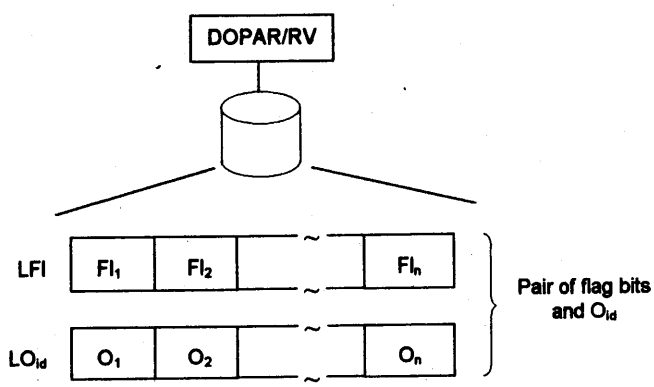

Figure 11. The pair of lists required for the functioning of DOPAR/RV

Each concurrent object started with rendezvous possibilities (with attribute RV in the message, sent to DOPAR/RV) has to have in its own message subset, a message activating its RV (rendezvous) method. The execution of the concurrent object's method RV, sends a message to the server object DOPAR/RV, to organize the synchronization:

DOPAR/RV: RENDEZVOUS(O_i)

where:

RENDEZVOUS is the method's name in the service object DOPAR/RV, which organized the rendezvous functioning;
O_i is the id of the concurrent object which was sent the message.

The functioning of DOPAR/RV requires the existance of a pair of lists for every started group of objects for parallel execution (see Figure 11). The first list, LO_{id}, is for object ids from the group. The second list, LFl, is for flags – each object from the group has one flag in this list. This pair of lists is used for the rendezvous mechanism. At the starting point of the parallel execution, all flags in LFl are zero. Every execution of RV sets its appropriate flag to one. When all flags are ones, then we have the common point for all concurrent objects and they can continue to work further.

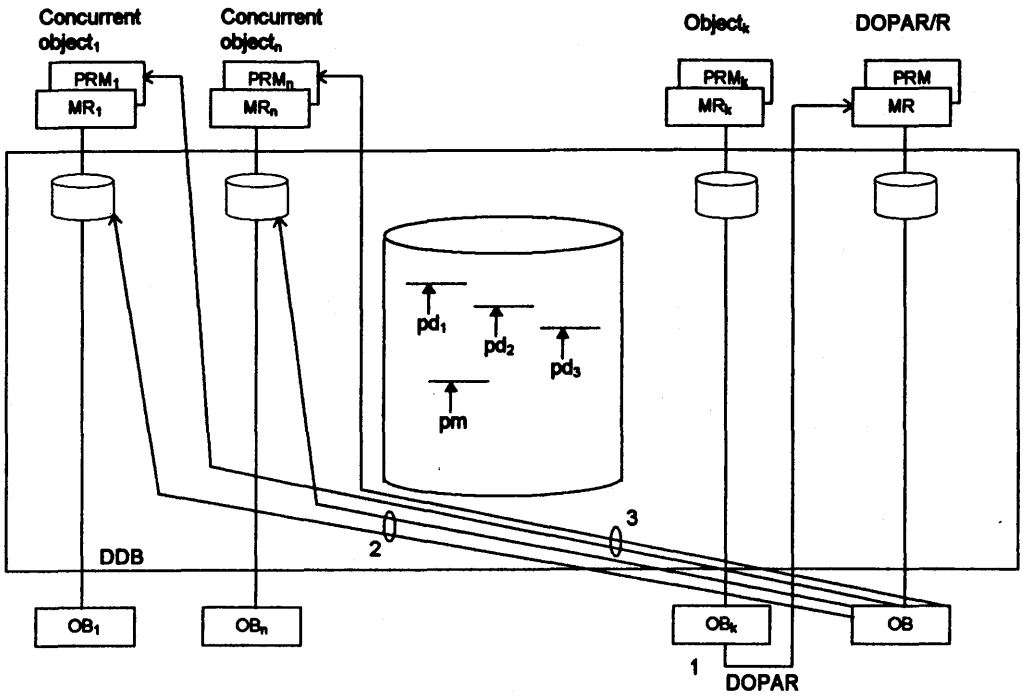

Figure 12. Parallel object execution for CMSCO+MD

Table 2. Results

	C					
Time	2	5	10	15	20	50
Time for object DOPAR/RV	30 ms	30 ms	30 ms	30 ms	30 ms	30 ms
Time for DOPAR/RV:xMSCO+MD	1.3s	3.0s	5.6s	8.2s	10.9s	27s
Time for Concurrent Object: RV (successful)	1.2s	2.7s	5.4s	8.0s	10.7s	26.6s
Total time for service procedures	3.5s	6.7s	12s	17.2s	22s	55s
Min time for efficiency of the synchronisation method	3.5s	1.7s	1.3s	1.2s	1.19s	1.13s

In Figure 12 we see the case of CMSCO+MD. A place exists in the distributed database where data is stored for the concurrent objects (pointed by Pd_i) and for the message subset (pointed by Pm). In this figure the object K starts the parallel execution on n concurrent objects. The algorithm of the work of the method MMSCO for multiple data is the following:

DOPAR/RV - method MMSCO for MD

```
choose a free pair of lists (LO_id+LFl);
put all objects' ids in the LO_id;
for all O_1–O_n
    write messages into object repository ;
        O_1:Pm_1,Pd_1
        ⋮
        O_n:Pm_n,Pd_n
activate all PRM_1–PRM_n.
```

The algorithms for CMSCO+SD, CMSCO+MD and MMSCO+SD can be considered as variants of the more common case, MMSCO+MD, given above.

The realization of the primitive RV is by the method RV in a concurrent object and the method RENDEZVOUS in the service object DOPAR/RV. The algorithms for these two methods are:

Concurrent object - method RV

```
send a message:
    DOPAR/RV: RENDEZVOUS(O_i)
set the object's flag WAIT;
End.
```

DOPAR/RV - method RENDEZVOUS

```
finds the appropriate pair LO_id+LFl;
sets the appropriate flag in LFl in 1;
tests all flags in LFl;
if all flags NOT= 1
    End;
else /*all flags are 1*/
    set all flags to 0;
    for all objects in LO_id
        change the object's flag WAIT
        to NOTWAIT
    for all objects in LO_id
        activate PRM ;
    clear LO_id;
    End.
```

6 Results

The experiment of the method of synchronization for concurrent execution has been done with ten computers, in which each one works with one concurrent object. The analysis of the algorithms and the results showed that scalability is valid for this synchronization method. That is why in Table 2 results are given not only for ten computers/objects, but for a projected 50 (for a case of a LAN from an average range). In this table "C" is the number of computers and the number of concurrent working objects as well. The parameter "Min time for efficiency of the method" shows that for object's number > 10, this time is stabilized. That means, if we have 10 parallel executed objects and every object has a time of parallel execution bigger than 1.3 sec, this parallel execution is more effective than the sequencial execution of the same objects in one computer (the amount of the time is strictly oriented to the testing environment). We would like to stress that for PC 386 DX/40, for which the mentioned results are received, to have a process with an execution time bigger than 1.3 sec is not a problem. That means the method of synchronization for concurrent execution is efficient for general purpose applications and not just for a special application area.

Table 3. Efficiency of synchronisation method

Time of execution	C					
	2	5	10	15	20	50
2s	0.7	1.1	1.4	1.5	1.6	1.7
5s	1.17	2.13	3.0	3.4	3.6	4.2
10s	1.5	3.0	4.5	5.5	6.1	7.7
20s	1.7	3.7	6.2	8.0	9.4	13.4

In Table 3 the efficiency of the method of synchronization for concurrent execution is given, depending on the time taken by the work of the concurrent objects. It is assumed that all concurrent objects in one experiment have the equal time of parallel work and this time is called "Time for execution". From this table it can be seen that with 10 or more parallel working objects efficiency improves. It can be said as well that the developed synchronization method ensures a good use of all available free resources.

In [9] and [13] an analysis of the efficiency of a few parallel computers is made. If we compare that efficiency with the efficiency of the proposed parallel work of objects in DPOA, we will receive results, given in Figure 13. In this figure are included data from three parallel computers – Alliant FX/2000 20, CRAY Y-MP C90, Supernum S1C1 and from two cases of Synchronization for Concurrent Execution (SCE) parallel object work – with Time of execution 5 sec and 20 sec. As can be expected, the multiprocessor systems have better efficiency than the parallel organized work in distributed objects. But it has to be stressed that the proposed method does not need special, expensive, dedicated parallel work computers. It uses the existing computers with their existing networks, ensuring only a new type of engagement of the resources. And at the same time the efficiency of the work of the method synchronization for concurrent execution is in the same range as that of parallel computers.

7 Conclusions

The structure of objects for distribution and for parallel execution and for the synchronization methods and algorithms for it in an APPC environment gives the opportunity for efficient and universal use of the existing computer resources

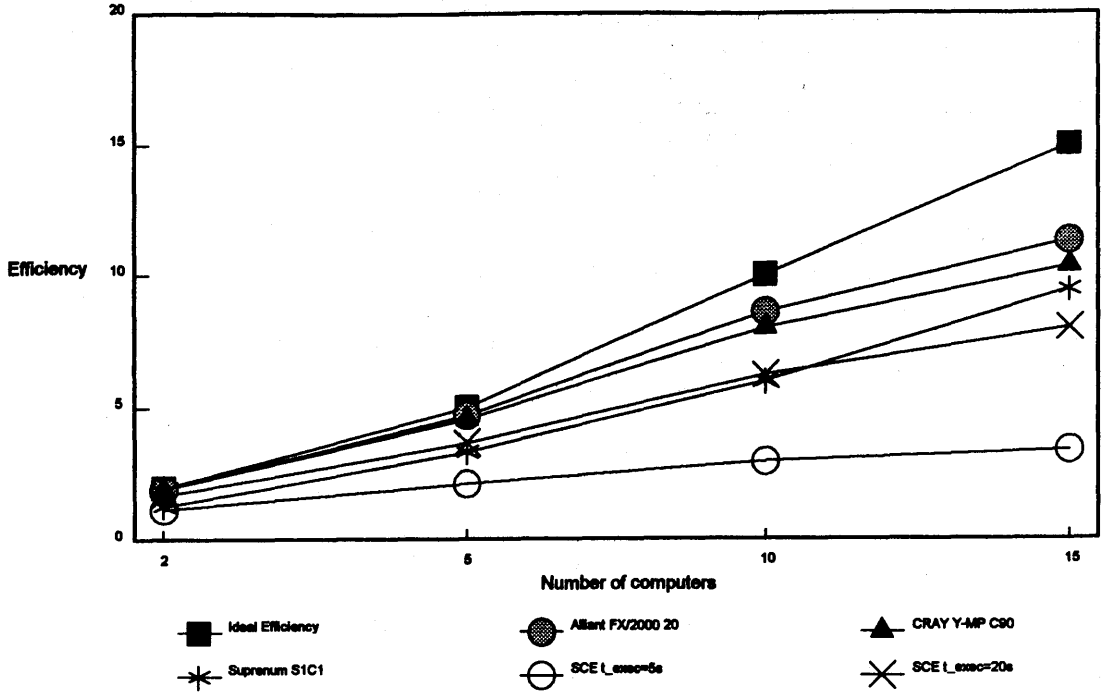

Figure 13. Comparison between the efficiency of parallel computers and the method SCE for DS

in distributed system. The experimental results show that the presented principles and methods do not need a special class of applications or hardware solutions, in order to be efficient. The boundaries from which this efficiency is valid is within the requirements of the general applications, which makes the proposed approach applicable for universal purposes. So, the results received from the parallel execution of distributed objects in an APPC environment can be compared with the results received from the concurrent work of a multiprocessor computer.

References

1. M Abbott and L Peterson. 'A language based approach to protocol implementation'. *IEEE/ACM Transaction on Networking*, **1**(1):4–19, (1993).
2. G Agha. *Actors:A Model of Concurrent Computation in Distributed Systems.* MIT Press, Cambridge, MA, 1988.
3. P Ammann and S Jajodia. 'Distributed timestamp generation in planar latice network'. *ACM Transaction on Computer Systems*, **11**(3), (1993).
4. T Anderson, B Bershad, E Lazovska, and H Levy. 'Scheduler activations: Effective kernel support for the user-level management of parallelism'. *ACM Transaction on Computer Systems*, **10**(1), (1992).
5. G Booch. *Object Oriented Design with Applications.* Addison Wesley, 1991.
6. A Burnsa and G Davis. *Concurrent Programming.* Addison Wesley, 1993.
7. P Ciancazini, K Jensen, and D Jankelvich. 'The semantics of a parallel language based on a shared dataspace'. Technical Report TR 26/92, University of Piza, (1992).
8. E W Dijkstra. 'Co-operating sequential processes'. In *Programming Languages.* Academic Press, (1961).
9. J Dongara. 'Performance of various computers using standard linear equations software'. *Computer Architecture News*, **20**(3):22–44, (1992).
10. C Fidge. 'Logical time in distributed computing systems'. *IEEE Computer*, (August 1991).
11. A Goscinski. *Distributed Operating Systems – The Logical Design.* Addison Wesley, 1991.
12. J Grudin. 'Computer supported cooperative work'. *Communications of the ACM*, **34**(12), (1991).
13. V Kisimov. 'Intelligent workstation engine's architecture'. Technical Report 1992-13, Computer Science Department, University of the Witwatersrand, Johannesburg, (1992).
14. V Kisimov. 'Distributed parallel object architecture'. Technical Report 1993-04, Computer Science Department, University of the Witwatersrand, Johannesburg, (1993).
15. G LeLann. 'Distributed systems – towards a formal approach'. In *Proceedings of IFIP Congress*, Toronto, (1977). North-Holland Publishing Company.
16. J Martin. *SNA:IBM's Networking Solution.* Prentice Hall, 1987.
17. C Murray and R Franks. 'Alternative software architectures for parallel protocol execution with synchronous IPC'. *IEEE/ACM Transaction on Networking*, **1**(2):178–186, (1993).
18. G Neiger and S Toueg. 'Simulating synchronized

clocks and common knowledge in distributed systems'. *Journal of the ACM*, 40(2):334–367, (1993).
19. A Tanenbaum. *Modern Operating Systems*. Prentice-Hall International, 1992.
20. C Thekkath, T Nguyen, E Moy, and E Lazowska. 'Implementing network protocols at user level'. *Computer Communication Review*, 23(4):64–73, (1993).
21. R Tolksdorf. 'LAURA: A coordination language for open distributed systems'. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, Pittsburg, Pennsylvania, (May 1993).

List of abbreviations

APPC	Advanced Program to Program Communication
DPOA	Distributed parallel object architecture
LU	Logical Unit
MR	message receiver
PRM	parser of receiver messages
OR	object repository
OB	object's body
AB/DAB	Active Body / Deactive Body
ESM/NESM	Empty set of messages / Not Empty set of messages
MES	service object for mutual exclusion synchronization
TAS	"Test-and-Set Flag" application method
RS	"Reset Flag" application method
SvTAS	"Test-and-Set Flag" service method
SvRS	"Reset Flag" service method
CMSCO	Common message subset for concurrent objects
MMSCO	Multiple message subsets for concurrent objects
SD	Single data
MD	Multiple data
DOPAR	Do in parallel method
RV	Rendez-Vous method

Received: 8/93, Accepted: 4/94, Final copy: 5/95.

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research notes. However, non-refereed review and exploratory articles of interest to the journal's readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal, papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for *review* should be prepared according to the following guidelines.

- Use wide margins and $1\frac{1}{2}$ or double spacing.
- The first page should include:
  - title (as brief as possible);
  - author's initials and surname;
  - author's affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be numbered and titled.
- References should be listed at the end of the text in alphabetic order of the (first) author's surname, and should be cited in the text in square brackets [1–3]. References should take the form shown at the end of these notes.

Manuscripts accepted for publication should comply with the above guidelines (except for the spacing requirements), and may be provided in one of the following formats (listed in order of preference):

1. As (a) LaTeX file(s), either on a diskette, or via e-mail/ftp – a LaTeX style file is available from the production editor;
2. As an ASCII file accompanied by a hard-copy showing formatting intentions:
   - Tables and figures should be original line drawings/printouts, (not photocopies) on separate sheets of paper, clearly numbered on the back and ready for cutting and pasting. Figure titles should appear in the text where the figures are to be placed.
   - Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.

   Contact the production editor for markup instructions.
3. In exceptional cases camera-ready format may be accepted – a detailed page specification is available from the production editor;

Authors of accepted papers will be required to sign a copyright transfer form.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect typesetting, reproduction and other costs. Currently, the minimum rate is R30-00 per final page for LaTeX or camera-ready contributions that require no further attention. The maximum is R120-00 per page (charges include VAT).

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers in category 2 above may be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. Camera-ready submissions will only be accepted if they are in strict accordance with the detailed guidelines.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to less than about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.

References

1. E Ashcroft and Z Manna. 'The translation of 'goto' programs to 'while' programs'. In *Proceedings of IFIP Congress 71*, pp. 250–255, Amsterdam, (1972). North-Holland.
2. C Bohm and G Jacopini. 'Flow diagrams, turing machines and languages with only two formation rules'. *Communications of the ACM*, **9**:366–371, (1966).
3. S Ginsburg. *Mathematical theory of context free languages*. McGraw Hill, New York, 1966.

South African Computer Journal

Number 14, July 1995
ISSN 1015-7999

Suid-Afrikaanse Rekenaar-tydskrif

Nommer 14, Julie 1995
ISSN 1015-7999

Contents