Communications of the Association for Information Systems

Volume 2 Article 24

December 1999

# Legacy Information Systems and Business Process Change: A Patterns Perspective

Ashley D. Lloyd
*University of Edinburgh*, A.D.Lloyd@ed.ac.uk

Rick Dewar
*University of Edinburgh*

Rob Pooley
*Heriot-Watt University*

Follow this and additional works at: https://aisel.aisnet.org/cais

Recommended Citation
Lloyd, Ashley D.; Dewar, Rick; and Pooley, Rob (1999) "Legacy Information Systems and Business Process Change: A Patterns Perspective," *Communications of the Association for Information Systems*: Vol. 2 , Article 24.
DOI: 10.17705/1CAIS.00224
Available at: https://aisel.aisnet.org/cais/vol2/iss1/24

This material is brought to you by the AIS Journals at AIS Electronic Library (AISeL). It has been accepted for inclusion in Communications of the Association for Information Systems by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

Communications of the Association for Information Systems

**Volume 2, Article 24**
**December 1999**

# *LEGACY INFORMATION SYSTEMS AND BUSINESS PROCESS CHANGE:*

## A PATTERNS PERSPECTIVE

Ashley D. Lloyd
Management School
University of Edinburgh

Rick Dewar
Division of Informatics
University of Edinburgh

Rob Pooley
Computing and Electrical Engineering
Heriot-Watt University

A.D.Lloyd@ed.ac.uk

**LEGACY SYSTEMS**

# *LEGACY INFORMATION SYSTEMS AND BUSINESS PROCESS CHANGE :*

# A PATTERNS PERSPECTIVE

Ashley D. Lloyd
Management School
University of Edinburgh
Rick Dewar
Division of Informatics
University of Edinburgh

Rob Pooley
Computing and Electrical Engineering
Heriot-Watt University

A.D.Lloyd@ed.ac.uk

## ABSTRACT

Integration of technical systems with business processes, and coherent strategies for the development of both, have long been recognised as critical to the competitiveness of companies. However, as separate systems become integrated, dependencies are established that complicate future reengineering exercises. These internal dependencies increase the risk associated with change, promote incremental approaches to systems reengineering and hasten the emergence of legacy systems. Once established, these legacy systems not only represent an impediment to advancing the technology strategy, they may also lock in redundant business processes with a consequent erosion of competitiveness.

Reengineering these legacy systems to improve competitiveness therefore requires both technical expertise in systems engineering and an understanding of what the business process is intended to achieve. Recent technical and business change drivers such as the 'Year 2000 Problem' (Y2K)

and Economic and Monetary Union (EMU), however, exposed concerns that many organisations may lack even the required technical expertise.

Clearly a demand for improving the capture and dissemination of systems reengineering expertise exists. One recent and promising approach to allowing transfer of expertise in well-defined contexts uses patterns. This paper explores patterns as a means of codifying and disseminating systems reengineering expertise. Through widening the definition of a legacy system to include the business process, we propose that patterns may provide a communication link between business and technology strategists that would help align their objectives and improve the sustainability of any resulting competitive advantage.

**Keywords**: patterns, reengineering, business processes, business strategy, information technology, competitive advantage.

## I. BUSINESS PROCESS AND TECHNOLOGY STRATEGY

A business process is a sequence of activities that achieve a business outcome. These outcomes are typically quantified at a customer interface, because this point is the only one at which value to a customer [Hammer and Champy, 1993] and hence a direct improvement in the competitiveness of the business can be measured. Hence, the ends of a business process are typically delimited by internal or external customer interfaces (Figure 1), with a socio-technical business system in the middle that crosses a number of functional boundaries within the firm. As competition increases, companies that wish to remain in a particular market typically seek to improve their competitiveness either by reducing the cost of production or by increasing the value perceived by the customer.

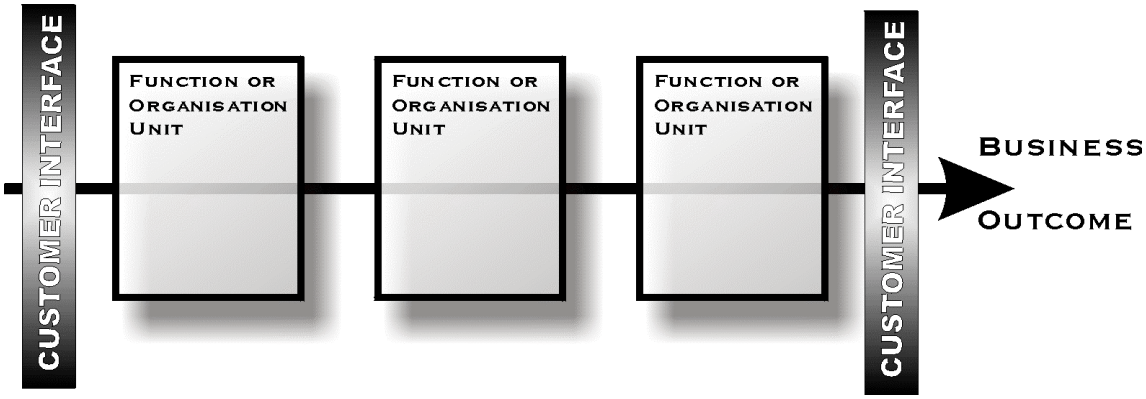


Figure 1. A Generic Business Process.

Technological change provides a bewildering number of ways to help achieve these goals. For example, Davenport & Short, [1990] identified 9 different types of competitive benefit to a business process that can be achieved through investments in Information Technology (IT) alone.

The permutations of type of competitive benefit, and the fact that they could appear at any or all points of the business process, complicates rational decision-making about changes in technology. Assessing the benefits for each new investment is impractical in a company of any size, and hence guidance is usually provided by an overall technology strategy. However, as prior studies by Ein-dor et al. [1984], Benjamin and Scott Morton [1988], Cronin et al. [1988] and Cecil & Goldstein [1990] show, formulating a good strategy for technology is not enough. To make a significant impact on competitiveness, a technology strategy must support the business strategy and the technology itself must be integrated with the other operations and processes of the firm. This point has long been recognised in a manufacturing context [Skinner, 1969].

Even with a business process perspective and a coherent technology strategy, however, investment cases guided by a rational desire to eliminate performance bottlenecks can still end up being the responsibility of a single

functional unit. Each unit will tend to focus quite narrowly on its own issues, and hence evolution of the organisation as a whole proceeds with:

1. ‘business process’-scoped initiatives using technology to achieve process benefits on the ‘cost’ side of the balance sheet, and

2. functionally-focused initiatives to add new product/service characteristics at a defined customer interface to protect and develop the ‘revenue’ side.

The former increases the degree of integration of existing systems while the latter may create new ‘islands of technology’ [Swenson and Cassidy, 1993] that will have to be integrated in the future.

This integration-introduction-integration cycle increases the coupling between individual systems that are operated by people for whom many of the couplings/dependencies are hidden within the system. This cycle forms an organisational ‘intra-structure’ that is typically understood by few people within the organisation (a situation often exacerbated by rounds of downsizing and outsourcing) and becomes a constraint to system redesign that promotes incremental approaches to systems reengineering. Successive revisions do little to change core functionality but do tend to obscure the original structure, increase resistance to change, and eventually produce what may be recognised as a ‘legacy’ system.

## II. LEGACY: SYSTEMS AND BUSINESS PROCESSES

Legacy systems can be defined [Brodie and Stonebraker, 1995] as those that significantly resist modification and evolution to meet business requirements, with a consequentially negative impact on competitiveness. This working definition is chosen carefully from the many alternatives available, because it recognises that a system that is simply ‘old’ or inflexible is not necessarily a legacy system if there is no business requirement for change. Such systems may simply ‘expire’ at the point at which the market being served ceases to be economic or appropriate. Expiration was notably the case of the electronic programmable computer built in 1943 by Turing and

his colleagues, which ‘expired’ in 1945 when the demand for its code-breaking capabilities reduced significantly [Sale, 1999].

Having identified a legacy system, the now famous decision matrix (e.g., Jacobson and Lindstrom, 1991) shown in Figure 2 demonstrates the choices available to an organisation. Candidates for reengineering are those which are too valuable to the business to be discarded, but are too hard to change/enhance without restructuring. The task of categorising systems in this way is already acknowledged to require great skill [SRAH, 1997; Brown et al., 1996 and Ransom et al., 1998], but in this paper we extend the definition of ‘value to the business’ to include explicit consideration of anticipated changes to the business process itself. At a trivial level this extension might simplify matters, as if the business need disappears then a potential candidate for reengineering may become a candidate for disposal. However, in general this extension adds to the complexity of the analysis as it forces explicit consideration of the impact that changes in business process may have on the choice of technology, and the impact that improvements in technology may have on the design of the business process.

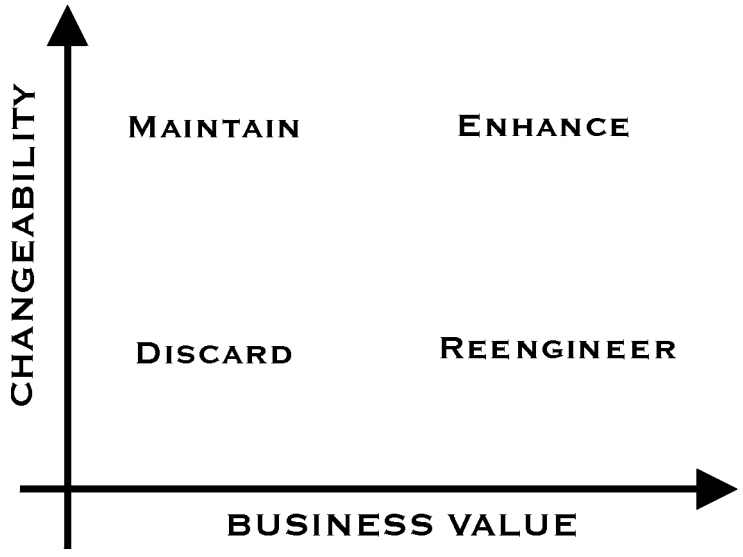


Figure 2. Decision Matrix for ‘Legacy’ Systems.

An example of the complex choices facing a company with a legacy system was seen by one of the authors in a Life Assurance company [Lloyd et al., 1997]. Figure 3 captures one interaction between a business process and

a legacy computer system. The process involved is the preparation of a quotation for dispatch to an Independent Financial Advisor (IFA). It shows how an inflexible mainframe resulted in the creation of manual systems to support new financial products. Reengineering was needed because the system was too valuable and too inflexible. Immediate cost benefits would attend the automation of the manual processes. However, the final approach would depend on how the characteristics of the business process could be made more competitive through use of new technologies. These 'new' technologies included replacement of the letter quotation by a direct FAX and potentially Internet capabilities. Contemplating the Internet however, complicates the decision as it potentially challenges the need for the original business process!

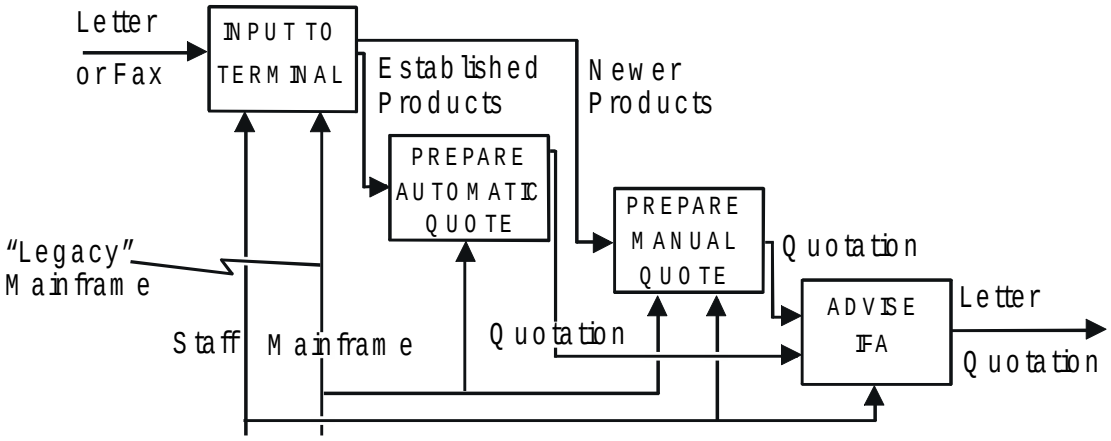


Figure 3. Industry Standard (IDEF0) Representation of Part of the Annuity Cycle of a Major Assurance Company

The benefit of considering both process and technology concurrently is not a new observation. Hammer and Champy, who are widely credited with introducing the term BPR, state that 'a company that cannot change the way it thinks about information technology cannot re-engineer' [Hammer and Champy, 1993]. However, as Heany noted over 30 years ago, IT should not be used to try to bring a failing system under control - automating a mess produces, at best, a faster mess [Heany, 1968]. Information technology is therefore only part of the over-all solution, and whilst IT is a central enabler of

organisational change [Coombs and Hull, 1995; Peppard, 1996] it is ultimately the business process that constrains the organisation's performance as a whole.

Though a linear model of 'process then technology' development is clearly problematic, a conclusion that could be drawn from the above is that IT systems should conform to the established/re-designed business process through the purchase/development of bespoke systems. This approach was accepted for large corporate IT projects during the 1980s. However, the rise of pre-packaged solutions to common business processes such as accounting and invoicing, produced with economies of scale and benefiting from compliance with complex legislation, changed the equation. Smaller companies now found an economic incentive to 'fit' their business process to the standard solution.

During the 1990s, pre-packaged solutions were increasingly accepted by large companies. These companies also saw standard solutions provided by market leaders as a means of benchmarking best practice. The pre-packaged market grew quickly in the mid 1990s, with the number one reason for buying the market leading product, SAP R/3, given perhaps not surprisingly as business process re-engineering [Newswire, 1999]. Though 'BPR' as a justification for investment was overtaken by 'ERP', process re-engineering is still identified as a critical step in achieving returns on the investment [Manchester, 1999].

Accepted wisdom might now therefore be re-expressed as 'modify' rather than 'make'. Kempis & Ringbeck [1998], who made detailed studies of 70 manufacturers across Europe, the United States and Asia, support this aphorism. They classified the companies surveyed according to the efficiency (IT cost as a percentage of revenues, plus project management performance against schedule and budget) and effectiveness (the availability, functionality, and utilisation rates of IT applications for each core business process). Although over 60% of their sample used functionally integrated standard software, this figure varied significantly over the segments identified. Of those who made very efficient and/or effective use of IT, 77% used standard

software, compared to only 42% of those identified as both inefficient and ineffective.

However, standard software is not always a rational choice. Whilst it can help a company to make its cost base competitive, the values that differentiate its products from the competition may arise from unique aspects of its business process. In these cases, the need to modify a standard solution heavily rather than the business process may eliminate the economic advantage [Kempis & Ringbeck, 1998]. Ultimately, internal development is the surest way of protecting a unique competitive strength.

What is clear from the above analysis is that the design of large IT systems is extremely hard to separate from the design of business processes. The question then arises: can legacy computer systems 'lock-in' inefficient or even redundant 'legacy' business processes?

A review of the BPR literature shows that many occasions exist where people are employed to perform tasks that are supported by IT but for which there is no business case. For example, when Ford compared its operations to Mazda, it discovered that its Accounts Payable department employed 500 people to match 14 documents, whereas Mazda employed only 5 people to perform the same function. The subsequent re-design cut the number of document matches to 3 and reduced the headcount by 75% [Talwar, 1993]. Many of Ford's documents proved to be redundant, however because the IT system helped manage the process of checking those documents, IT also helped to 'lock-in' the need for them. IT was acting as a barrier to the re-design of the process.

The literature on the learning organisation also supports this argument. Lambert and Peppard [1993] observe that the patterns of behaviour in large organisations are typically 'hard-wired' into the system through organisational structure, incentive schemes, hiring and promotion practice, and notably information systems.

It seems clear therefore, that cases do exist where legacy systems helped to 'lock-in' inefficient business processes. Hence effective reengineering requires an appreciation of both the opportunities for changing a business process in the light of 'new' technology, and the potential for

simplification of the technology required through rationalisation of the business process - if the ‘mess’ referred to by Heany is not to be perpetuated.

The pervasiveness of legacy systems, as well as their potentially critical impact on competitiveness, is illustrated through corporate recognition of issues such as the Y2K problem. Though systems engineers recognised the existence of this problem as long ago as the 1950s [Caminer et al., 1996], this awareness only relatively recently achieved prominence at a policy level in both industry and government. The resultant attention that legacy systems received exposed concerns that many organisations, and especially those that ‘de-layered’ in the early 1990s, lack the required expertise in systems reengineering to respond effectively to this issue. Moreover, when the organisation’s business processes are constrained by the ‘intra-structure’ of a restrictive legacy, the opportunities to improve competitiveness proactively are not only reduced, the risks associated with change increase.

Taking the above factors into account and applying our chosen definition of legacy to both young and old systems today, we can assert that the legacy problem is:

| | |
|---|---|
| 1. widespread | it affects all kinds of organisations |
| 2. large in scope | legacy systems may ‘lock-in’ legacy business processes |
| 3. serious | failure to reengineer can hamper an organisation’s attempts to remain competitive; |
| 4.complex | both the decision to reengineer and identification of a competitive solution depend on assessing the interdependencies and benefits of sequential or concurrent changes in technology and process |
| 5.persistent | there seems no reason to be confident that today’s new systems are not also tomorrow’s legacy systems |

## III. PATTERNS AS AN APPROACH

The architect and philosopher Christopher Alexander introduced the concept of generally applicable patterns in the late 1970s [Alexander et al. 1977, Alexander 1979]. He recognised that certain attributes in building and

urban design frequently occurred throughout history and across cultures. In other words, he identified successful solutions to recurring problems in context, and found a way of communicating these solutions by standardising the format of each pattern and linking related patterns to form a ‘pattern language’.

In general, therefore, a pattern must contain a description of the *problem* and the *solution*. To be generally applicable, a pattern must also contain a description of the *context* in which the problem arises, and the potential *consequences* of applying the pattern. The users of the pattern can then map the solution to their own circumstances, assess the potential consequences, and implement it according to their own priorities. To become a pattern language, these patterns must have *names* that evoke the problem, solutions and consequences in a few words, and thereby allow patterns to be related to each other at a higher level of abstraction. It is this ‘localisation’ of the solution, and its use in concert with others, that results in the use of a core solution “a million times over, without ever doing it the same way twice” [Alexander, 1977].

We aim to understand how experienced practitioners undertake the reengineering of legacy systems, so that we can develop better techniques and material for communicating that expertise. Such expertise will necessarily draw on a number of fields, and hence it is important that any emerging ‘language’ uses constructs that will be meaningful to all the communities who are able to contribute expertise. Though patterns are a relatively recent development, they have already been applied to design [Gamma et al., 1995], the software development process [Cunningham, 1995; Coplien, 1995], organisational patterns [Coplien, 1997], software process improvement [Appleton, 1997], and directly to business process reengineering by Beedle [1997]. Although these classes of patterns are all different, the apparent suitability of patterns for codifying and communicating expertise in these complementary disciplines makes it natural to consider patterns as an approach also to the reengineering problem.

Patterns lead to a number of important difficulties. Two of the most vexing are

- the level of abstraction chosen, and
- the treatment of patterns that encapsulate the opportunities of new technology.

Both of these difficulties impact the effectiveness and risk of using this technique to communicate expertise. If the level of abstraction is too detailed, patterns become over-specialised and unwieldy. If they are too abstract then they start to assume potentially dangerous levels of domain knowledge by the user. Any pattern that arises from a new technology is, by definition, unproven and may be exceptionally risky, but without the possibility of adapting to technical change, any language will become obsolete. After all, how limiting might Alexander's patterns be for a designer with a new building material, or one who is expected to design a living space in outer space! These issues are being addressed in the wider pattern community and were discussed by the authors in Dewar et al. (1999).

In view of the need to define a formal structure that allows for a high level of abstraction without making patterns unwieldy, and which prevents new patterns from being mistaken for established patterns, we propose the inclusion of two further elements: *Status* and *Known Uses*. *Status*, describes the level of confidence which may be placed in the pattern, and *Known Uses* describes cases where the pattern was used successfully, giving specific names of products or companies.

The elements of the *systems reengineering pattern* are therefore:

- Name — a few words, describing as evocatively as possible, the overall nature of the pattern.
- Status — a few words, describing how well established the pattern is (see 'Known uses').
- Context — a situation giving rise to a problem
- Problem — the recurring problem arising in that context
- Solution — a proven resolution of the problem
- Consequences — notes on the merits and demerits of the resolution described, with references to other possible solutions or relevant patterns where appropriate
- Known uses — of this pattern.

One important distinction between a design pattern and a systems reengineering pattern is whether a related business process exists. A reengineering pattern must apply to a system which already exists and for which there is an existing case for both the system itself and the related business process. An existing process need not be true for a design pattern. In a reengineering pattern therefore, the context must reflect both the technical and business imperatives. This distinction is important because it gives an opportunity to achieve the real communication between business and technical communities that is required for coherent technical and business strategies. Similarly, a legacy system can only be reengineered into an evolvable system capable of responding to, or anticipating, business process needs if this communication can also take place between domain specialists throughout the reengineering process. Real communication requires not just a common language, but also a common conceptual base if the classic problems with literal translations are to be avoided. We therefore propose that by embedding concepts of competitive advantage we enhance the potential for a common focus on what is specifically required by the organisation to improve the competitiveness of its business processes.

## COMPETITIVE ADVANTAGE

Though economists agree about the importance of investment in technology for the growth of economies [Boltho, 1996], attempts to uncover relationships between IT investment and competitiveness at the level of the firm produced rather inconclusive results. Studies carried out across Europe [OECD, 1988], supported by related studies in the US [Cronin, 1988, Strassman, 1990], showed no positive correlation between expenditure on IT and changes in productivity, and established what came to be known as the "productivity paradox" of investments in IT during the 1980s. This analysis was followed with notable IT project failures in the early 1990s, such as the London Stock Exchange's Taurus, leading some commentators to observe that far from leading to increases in productivity, IT investment led to 'wealth destruction' [Kirkpatrick, 1994; HMSO, 1994].

Brynjolfsson and Hitt later challenged this productivity paradox in their 1996 study of a new data set, on which they applied many of the same tests and yet were able to find significant evidence of productivity increases at a firm level. Though they challenged the productivity paradox, this did not necessarily invalidate the earlier studies as increased productivity might have arisen from industry-wide changes in business and IT strategy in the late 1980s, which they raised as a possibility but did not test. Willcocks and Lester (1997) reinforced the Brynjolfsson and Hitt results in their review by highlighting deficiencies in evaluation practice at a macroeconomic level, but also identifying deficiencies in approaches to assessing benefits at an organisational level. The former obscures the true picture, whilst the latter inhibits alignment of business and technology strategies and thereby helps to propagate the paradox.

The lack of an over-all consensus in these studies may be partly explained by their focus on an aggregate measure of firm performance, i.e. productivity, which owes as much to the choice of business strategy as to how well the investment is aligned with, and advances that strategy. Any direct link between IT investment and a competitive advantage is further complicated by to the strong dependence of competitiveness on exogenous variables that include, naturally, the competition. If they too are investing heavily in IT, then unless the market in which the firms compete is growing, then no aggregate productivity increase will be registered in the short term.

Since an absolute competitive advantage cannot be determined before an investment in IT takes place, any generalisable approach to competitiveness needs to consider generic measures of competitiveness. These measures include:

- the ability to produce a product/service faster (Speed) and cheaper (Cost) (the basic tenets of Fordism)
- a focus on product/service qualities (Quality) that establish a higher value to the customer,
- considerations of mass customisation, and hence to the objective of flexible manufacturing processes (Flexibility).

A company may compete on any or all of these dimensions, as long as they contribute to a value that can be communicated to the customer. It is this 'communicated value' that turns a generic competitiveness improvement into an improvement of the company's competitiveness in its own markets, though again, the gain may not prove sufficient to provide the company with a competitive advantage.

These concerns about generic and specific competitive advantage are illustrated in Figure 4 in relation to organisational hierarchy and system architecture. Here it is the business strategy, championed by the CEO, that will drive the competitive moves of the organisation, and hence it is the business strategy that contains the factors which are believed to be critical for competition in the organisation's own markets. If this strategy is right, and is not pre-empted by competitive moves elsewhere, then a coherent technology strategy should lead to improvements in capability that translate into increased productivity.

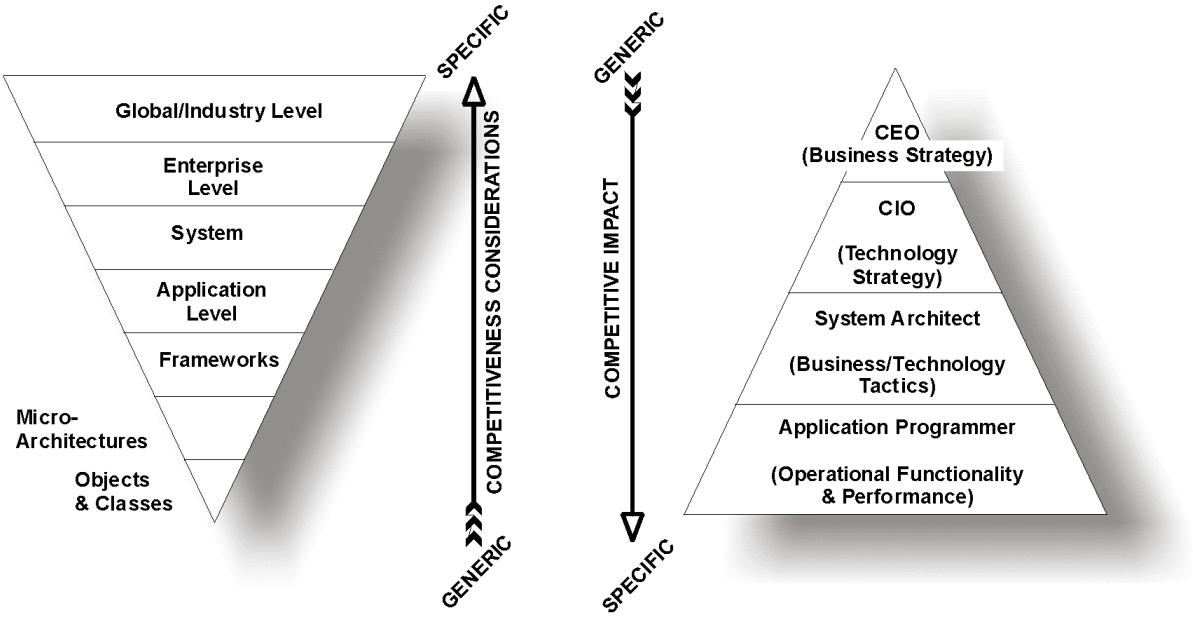


Figure 4. Architectural Hierarchy, and Organisational Hierarchy, Showing Links Between Activity, Consideration of Competitiveness and Competitive Impact. [Architectural hierarchy after Brown et al. [1998]].

A system architect, however, lies between the Business and Technology strategists and the application programmer. Although they are responsible for designing systems to support the business strategy and will usually be aware of any constraints that the technology strategy imposes,

their knowledge of the specific business strategy, and hence the factors of competition, is likely to be less detailed. Indeed as we descend through the organisational hierarchy, decision-making will be guided increasingly by considerations of generic competitive advantage rather than those specific to the market in which the company operates.

Since generic competitive advantage considerations can provide guidance at all levels and are a super-set of those elements which provide specific competitive advantage, it is sensible to use generic considerations as a basis for extending the pattern formalism. Adopting this format of pattern context promotes the opportunity for system architects and application programmers to select patterns that address the business context as well as the technical context. The format also supports a ‘bottom-up’ dialogue about competitiveness that is consistent with the trend towards less formal organisational structures and modern system development methodologies. Note again, that a generic benefit identified at a system level that does not translate into a value that can be communicated to a customer cannot improve the competitive position of the company. For example, reengineering a system solely to 'improve' its speed cannot directly reduce competitiveness, but it need not bring any competitive advantage either. This result would be true if

1. the original system was already the fastest part of the overall business process, or
2. the organisation was a monopoly supplier of a scarce product that elects to use long delivery times to create perceived value, and/or create an additional income stream from customer's deposits.

In either scenario, speed gains are unlikely to be detected by the customer as a result of the reengineering, and hence the competitiveness of the organisation's offering is not directly affected.

Clearly, therefore, establishing a dialogue based on generic concepts of competitive advantage alone does not guarantee the alignment between technical and business strategies that is required if an organisation is to maximise the likelihood that a competitive advantage will result. For this reason, it is important that the dialogue supported by the context element of a

systems reengineering pattern can be domain-specific or even organisation-specific. A broader context not only improves the sensitivity of systems reengineering patterns to specific factors of competition it also widens the range of expertise that can be captured. A broader context therefore not only has implications for improved knowledge management within a company, but also provides the potential for complementary links with other reengineering methodologies, often allowing the re-use of some established solutions. Principal amongst these is the potential for re-using design patterns which specify something about the structure of the target system, where that target has been identified as a solution within a systems reengineering pattern (Figure 10, Section V).

This theme of re-use is one of a number of guiding principles for developing systems reengineering patterns (Dewar et al., 1999) which are intended to help ensure that the formalism keeps pace with developments in related fields without attempting to subsume them. To subsume related fields would confuse the focus of systems reengineering patterns, impede further developments, and quickly inflate their size and detail. Together these would reduce the accessibility of systems reengineering patterns to different user domains and increase the danger of their misapplication, with a consequently negative impact on organisational competitiveness. Remember that systems reengineering patterns are intended to structure communication between domains of expertise, not used by one domain as a substitute for expertise in the other.

## IV. EXAMPLE PATTERNS

If patterns are to be useful to people at different levels of the organisational hierarchy in Figure 4, they will have to reflect solutions within contexts at those levels. These contexts differ widely. Hence systems reengineering patterns at, say, a senior management level are likely to capture solutions reflecting a much broader range of concerns than those of a design engineer. In pure systems engineering terms these patterns may prove too general to be applied directly to a target system design. However, this

capture of different perspectives is a necessary condition for establishing a common language that supports communication between different levels of the organisation. Should detailed systems design patterns exist that are relevant to the stated context, then these patterns may be 'pointed to' by the reengineering pattern (Figure 10, Section V) and reviewed if the reengineering pattern user has the necessary design patterns experience. Table 1 is an example of a simple catalogue.

Table 1. A Reengineering Patterns Catalogue.

| **Reengineering Issue** | **Pattern** |
|---|---|
| 1. MIGRATING FUNCTION AND DATA | Divide and Modernise |
| 2. CHANGING SYSTEM INTERFACES | Wrapping |
| | Middleware |
| 3. MODULARITY IN PHASED PROCESSING SYSTEMS | Externalising an internal representation |
| **Managing Reengineering** | |
| MANAGING COMMUNICATION | War Room |
| | Work Shop |

In this section, we review four reengineering patterns that record business and technical imperatives expressed at different organisational levels to illustrate these points. We also introduce two patterns drawn from our study of the management of reengineering projects which illustrate how patterns might also be used to capture knowledge about the reengineering process itself. These managerial patterns are *War Room* and *Work Shop*, which show how communication issues were addressed in two different projects within the same organisation. Together these six patterns form the partial patterns catalogue of Table 1. They are described in the following subsections.

Two generic types of pattern are included: 'reengineering' patterns that relate system characteristics to business and technical imperatives, and 'managing reengineering' patterns which capture knowledge about the

reengineering process itself within the context of the organisation. Note that, just as there are different contexts in which the decision to re-engineer a system would include 'migrating function and data' or 'changing system interfaces', there may be different solution patterns that fall within each category. These differences are illustrated in the subsection titled 'Reengineering 2: Changing System Interfaces', where two patterns are included in the table. As the number of validated patterns grows, we expect the right-hand column of the catalogue to grow much faster than the left.

**REENGINEERING 1: MIGRATING FUNCTION AND DATA**

Migrating function and data in the context of evolving market demands is the classic legacy problem. As underpinning functionality becomes obsolete or reaches the limits of its flexibility, any organisation that does not migrate to a new system will either lose market share or end up building systems that effectively replicate large parts of the underlying processes. As redundancy increases, transaction costs increase and though this added cost may not affect the ability of companies to win new business, it will certainly affect their profitability. *Divide and Modernise* (presented without a business context in Stevens & Pooley [1998]) is a pattern that can help liberate the data and functionality contained within the legacy system and produce a system which itself should be easier to evolve in future.

**Name:** Divide and Modernise

**Status:** Draft.

**Context:**

***Technical context***: You have a legacy system whose technology is obsolete and soon to be unsupported. An area of functionality, such as a payroll component, is identified which is relatively decomposable from the rest of the legacy. You decided that significant modifications to the dying legacy system are undesirable. You may also have considered *Wrapping* the system. Although *Wrapping* is sometimes useful, it is not a good solution here because it perpetuates the use of the unsupported technology. If a new system is developed to replace part of the old one, the developers will be

expected to provide ideal functionality. Consequently, it will be impossible to manage expectations and the project will become large and risky.

***Business context:*** You have a wide range of data and communication standards limiting the effectiveness of your operations. You need to move to a common platform to rationalise your overlapping operations and support a geographical expansion. No commercial application is yet a de facto standard and you will have to perform the reengineering yourself. However, if you are the 'first mover' and your solution is both successful and flexible then you may be able to recover costs through selling the resultant expertise to your competitors. The competitive elements focussed on here are improving *Speed*, by reducing the number of interfaces traversed; reducing *Cost* of support and software licensing; and improving *Flexibility* of the system to support future product changes and hence the ability to support other customer's markets if the system is to be sold.

**Problem:** How can the system be migrated to supportable technology?

**Solution:** Translate the relevant part of the database to a modern format and rewrite/translate the associated code. Whilst handling any consistency and gateway issues, remove the redundant data and code from the original legacy. Now reengineer the separated component.

**Consequences:** You should consider whether to migrate the code or the data first depending on your own priorities and resources. Once the code (or data) is migrated you have a stable working system using an interim gateway. When the migration of the data (or code) starts, the gateway will have to be replaced or modified. Whichever you choose to migrate first, constructing the gateways is non-trivial, requiring careful planning.

Nevertheless, by not attempting to change the code you have acquired a new system providing part of the functionality of the original legacy. You are now left with a smaller legacy and a separate, manageably sized component of the original legacy on which you can consider further reengineering (Figure 5).

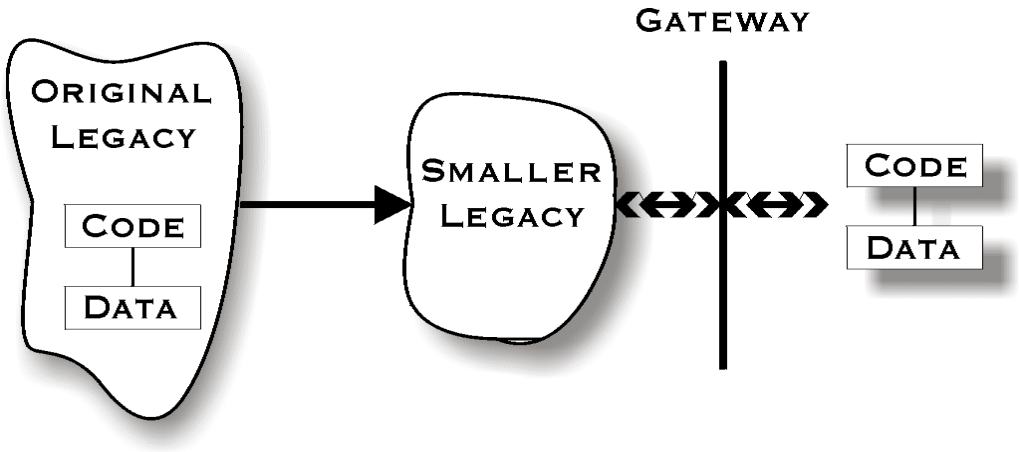


Figure 5 - Divide and Modernise

Over-all, work proceeds in distinct manageable phases. Even if a requirements explosion does overtake the final restructuring step, the main aim, that of removing the dependency of the functionality on the obsolete technology, will have been achieved. In addition, maintenance of the smaller legacy is now less costly and risky. On the negative side, code and data are migrated before the new requirements of the system are analysed, which means that some of this effort may turn out to have been wasted.

**Known uses:** This pattern is being used in BT (British Telecommunications plc) as they attempt to migrate data and functionality away from their monolithic Customer Service System (CSS) [Harrison, 1997]. It is only one of a number of strategies being pursued, such as *Middleware* and *Wrapping*.

Another example can be drawn from Brodie and Stonebraker's [1995, pp 67-102] legacy migration work for a US Telco and Bank. Here they used a combination of moving the data first (forward migration) and the applications first (reverse migration) to achieve their aims. In both these cases Divide and Modernise was effectively applied iteratively until all the legacy was migrated.

## REENGINEERING 2: CHANGING SYSTEM INTERFACES

Two related patterns are now presented that address instances where the interfaces to valuable systems are to be modified. Here the issues differ from Reengineering 1, at least in the importance attached to each. The loss of existing functionality may be fatal and must be clearly costed. The technical

tradeoff is in terms of future flexibility as opposed to an immediate cost.

**Name**: Wrapping

**Status**: Draft.

**Context**:

***Technical context:*** A system, or several sub-systems, whose interface is not ideal but where the underlying code and data are acceptable or are less urgently in need of reengineering. The interface is not readily decomposable from the rest of the system which makes re-writing it difficult and risky.

***Business context:*** Users of a system, paying customers or internal operators, need a more efficient means to access data or invoke processes. At the same time, the current system performs essential functions for the business. These inefficiencies mean competitors threaten to win business opportunities and to take existing business away. The competitive elements here are improving the *Speed* of response and improving the service *Quality* of the system through enhancing ease of use.

**Problem**: How can the system interface be made more efficient?

**Solution**: Design an improved user interface and the wrapper shell. The new interface can then invoke the wrapper's API. The wrapper then makes requests from the legacy by marshalling the arguments in the API call. Subsequently, when the legacy's response is received by the wrapper, it is fed back to the user interface in the appropriate format.

**Consequences**: Wrapping is often the simplest solution and renders the unsuitable interface invisible to outside users and systems. Effectively, the functionality of the legacy is now distributed and can be deployed on heterogeneous systems. However, the pattern merely covers up a legacy problem and would introduce a performance penalty. If the underlying system needs to change in the future, that task, plus changes to the wrapper may become onerous. Since all the underlying systems are intact, it should be possible to access the original interfaces if necessary. If the issue of flexibility is significant, consider the *Divide and Modernise* pattern. Alternatively, if the interface is 'owned' by a number of business processes and future developments cannot be fully characterised, then consider *Middleware.*

**Known uses**: Wrapping is encountered widely, especially in the literature on object oriented design, where a technical clash between relational database approaches and object oriented applications development was often a significant hurdle to innovation. Booch [1994] describes the technical issues in considerable detail.

In terms of systems reengineering, wrapping was used by BT (British Telecommunications plc) in their attempts to migrate away from their monolithic Customer Service System (CSS). Their migration strategy includes the use of a non-invasive 'wrapper' called CHIS (Customer Handling Intermediate Server), which provides a gateway into CSS. The resulting architecture allows the CSS functionality to be migrated to other platforms while still providing an integrated and consistent view to clients [Harrison, 1997].

Wrapping was also used to add a Web-based interface to a command-line driven legacy system [Phanouriou and Abrams, 1997]. One of the authors has first hand experience of this reengineering pattern when consulting for a major UK steel manufacturer, where Smalltalk-based object oriented user access to an existing corporate database was being planned.

Sneed and Majnar [1998] describe a banking application of wrapping where they encapsulated existing mainframe software components so that they can be used in a distributed object-oriented system. Their findings are:

1. inevitably the wrapped program will need modification in order to interact with the wrapper;

2. the bottleneck and weakest link in the chain is the server to host communication, so special attention should be given to its capacity and reliability;

3. testing the resulting system is time consuming.

**Name**: Middleware

**Status**: Draft.

**Context**:

***Technical context:*** A legacy system exists that is stable and successful. It is in constant use and there are no immediate plans to significantly alter or replace it. However, separate systems are being

developed that could usefully exploit the functionality of and data on the legacy.

***Business context:*** New products need to integrate with the legacy to reduce operating costs, and/or to support new (e.g. Internet) or faster (e.g. Fax vs. Letter) communication media to win business. Existing user interfaces are considered too inflexible to support the wider range of products and their promotion, but acceptable user interfaces are not directly compatible with the legacy system. The competitive elements here are reducing *Cost*; improving *Speed* and *Flexibility*; and introducing new product and service *Qualities* through new communication functionality that may lead to the development of new markets.

**Problem**: How can new applications communicate and cooperate with the legacy and access its services?

**Solution**: Purchase a Message Oriented Middleware (MOM) product. Provide gateways on the legacy and other applications to allow the distributed systems to connect and interoperate.

**Consequences**: The legacy is effectively wrapped. New applications can be distributed. They can re-use the existing valuable functionality and data on the legacy without adding coupling and complexity (Figure 6). In addition, the operators of the new applications only have to deal with one familiar interface and will be unaware that they are interacting with other systems.

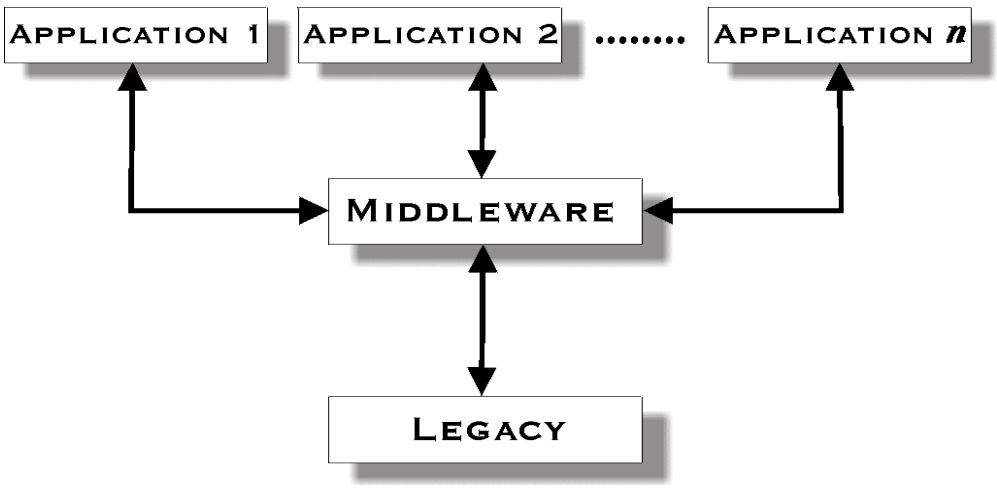


Figure 6 - Middleware Architecture

Because middleware products can experience capacity problems, the volume of transactions must be anticipated. Care should be taken with security and integrity so that the loose coupling between applications and legacy data does not allow data corruption or unnecessary duplication.

**Known uses:** Middleware applications for integrating legacy systems is a growing market, with applications specifically for this purpose available from companies such as IBM, Software AG of Germany and Oberon Software of Cambridge, Massachusetts. In one UK financial services organisation the authors saw the adoption of a MOM product to allow a new call centre to access data on their legacy mainframe. Exactly the same motivation was cited for the adoption of IBM's MQSeries by the Prudential Insurance Company of America [IBM, 1998]

In another UK based financial services company with operations across Europe, reengineering was required to improve the performance of a specific market channel and to support an increasing range of new products. The market channel supported was the sale of annuities through a network of Independent Financial Advisers (IFAs) in which the quotation process was slow with consequent erosion of market share. A middleware solution was used to integrate the new product offerings with the legacy system, eliminate redundant business processes and to access new functionality, i.e. a Fax capability, to speed up the over-all process.

Middleware is one of a number of patterns being used in BT's migration of functionality and data away from their monolithic Customer Service System (CSS) [Harrison, 1997] - see also *Divide and Modernise* and *Wrapping*. The main difference here is that BT have been developing their own MOM systems 'in-house' since 1984 due to a lack of commercial alternatives [Calladine, 1997].

## REENGINEERING 3: MODULARITY IN PHASED PROCESSING SYSTEMS

This pattern relates to reengineering of a product. In this context, the issue of competitiveness is quite clear. A business organisation needs to keep its products' features ahead of its competitors, without compromising its existing customer base. Thus, in introducing new features it must consider:

- implications for reliability of existing product features;
- implications for efficiency of existing products;
- expectations of existing customers;
- likelihood of further modifications in the future.

The reengineering implications can be seen as capturing the expertise embodied in the decision that it is possible and desirable to migrate in a certain incremental way to a new system which itself makes use of those design patterns, rather than writing a new system from scratch.

**Name**: Externalising an internal representation

**Status**: Draft.

**Context**:

***Technical context:*** A system in which data is processed notionally in a number of phases, where the phases are invoked by a driver program which itself is easily modified. Currently the system is tightly coupled, with each phase communicating with its predecessor and successor and taking no account of others. Such an arrangement, sometimes referred to as a pipelined architecture, is shown in Figure 7.

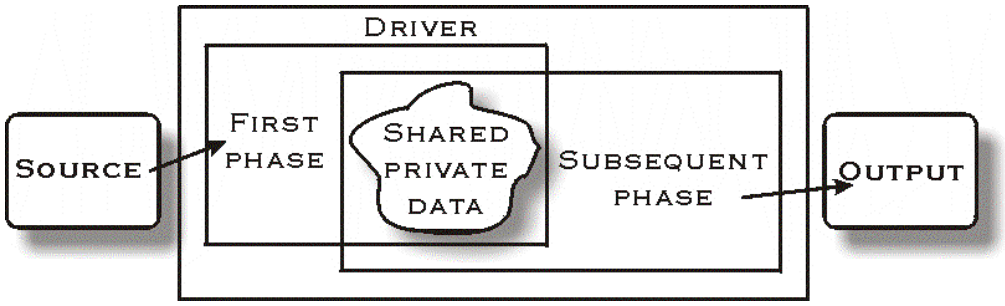


Figure 7 - System Before Reengineering

***Business context:*** In a fast-changing market, new products and services are being constructed by adding new features to, and combining, existing product lines. There is a consequent system requirement to add new (optional) phases between phases that previously were always called consecutively. It is expected that the business model of rapidly evolving new products and services from an existing portfolio is set to continue and that this

requirement is going to be regularly repeated. At the same time, there is a need to maintain the system's existing functionality - especially its reliability. The competitive element here is *Flexibility*.

**Problem**: Phases are not currently well encapsulated. Wherever two phases are currently consecutive, they always share an internal data representation which is adapted to the needs of those two specific phases, not designed to be an interface format for arbitrary processing. Adding the new optional phases to the system as it stands requires either that functionality of the existing phases be duplicated, or that some optional phase use the “internal” format which was not designed as an interface format. Either course will create unacceptable maintenance problems in this context, given that there is an anticipated need to add further phases in future. In addition, the current system’s functionality must not be compromised, otherwise existing customers may move to competing products.

**Solution**: First, incrementally replace the internal format with a newly defined and fully documented interface format, open to use by new phases. Modify the original first to output the new format as an optional alternative to the current means of sharing the information. Second, develop the new optional phase using the new interface format as first input and then output. The working of the new phase can be checked by feeding its own input back into itself. At this point, shown in Figure 8, the original first phase outputs two formats depending on what its successor will be. Third, modify the old subsequent phase to input the new format, at which point the old format can be abandoned, and the ability of the old first phase to output the old internal format can be removed, as shown in Figure 9. The structure of the new system is now modular, with the driver program as a Mediator, as in the design pattern [Gamma et al., 1995].

**Consequences**: The generation of an externally readable version of the representation allows new modules to be attached with no further alteration of the existing system, apart from the easily modifiable driver program. This solution creates a more open system. The original means of communication between the first and subsequent phase can be preserved as an option until the new external representation is fully tested. This approach avoids compro-

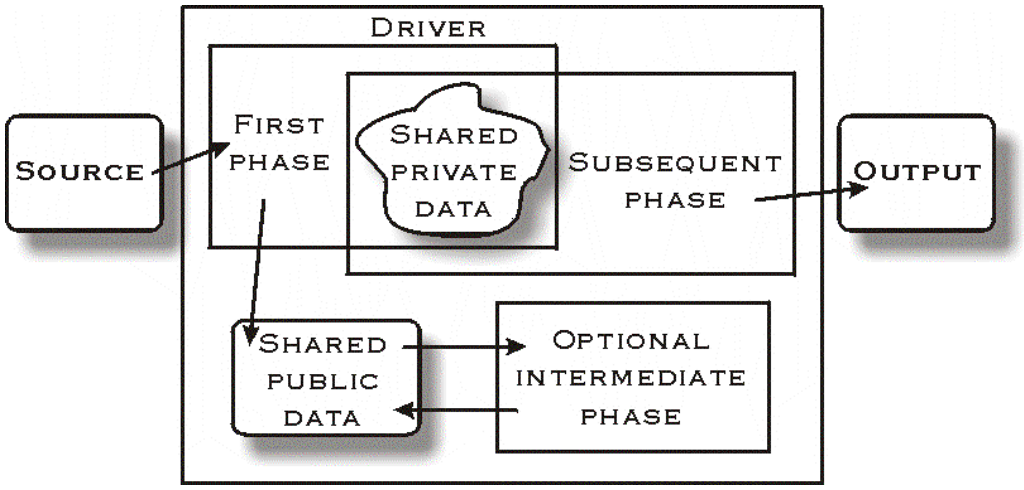


Figure 8 - System During Intermediate Stage of Reengineering

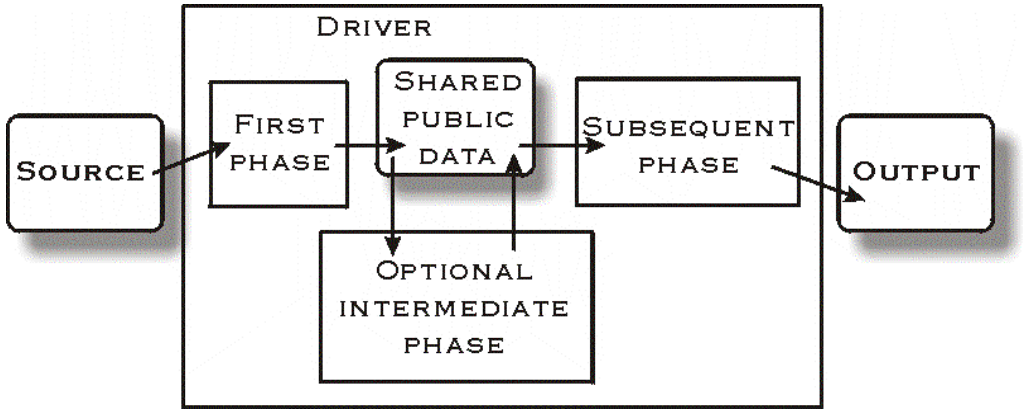


Figure 9 - System When Reengineering is Complete

mising existing uses during reengineering. Depending on the nature and use of the old intermediate format, the new system may possibly be slower than the old, since the interface format is no longer so well adapted to the particular needs of the two originally communicating phases. If speed is critical, this effect needs to be considered in designing the new format and the altered phases. The use of an external file, rather than memory, is an option with the new structure. An external file may be beneficial where memory is at a premium. The new architecture may also allow more portability to new back-ends.

**Known uses:** Two uses were observed personally by one of the authors.

In the first, a commercially marketed Fortran compiler, written at the Edinburgh Regional Computer Centre, was adapted to include a number of optional optimisation phases. Since the user community included compilation of several very complicated scientific programs of several million lines of code, maintaining existing correct behaviour was a major constraint, while fierce competition was a driver for change.

The second use occurred during the development of the Integrated Modelling Support Environment (IMSE) as part of an ESPRIT collaboration [Pooley, 1991]. The graphical interface was supported by a generic platform, known as the Graphical Interface System (GIS), which had been developed in-house by ICL prior to the IMSE collaboration. In opening this front end up to the other eleven partners, ICL developed a shareable text based representation, known as the Graphical Description Language (GDL), to replace a private, internal representation in use originally. This approach led to successful integration of several new facilities.

## MANAGING THE REENGINEERING PROCESS: MANAGING COMMUNICATION

The two patterns in this subsection are drawn from our study of how the process of a systems reengineering project is managed. They show how communication issues were addressed in two different reengineering projects within the same organisation, a large Financial Services company based in the UK. These patterns are summarised to a degree that would not make them useful outside of their immediate organisation and are not proposed formally for verification as patterns. They are presented to illustrate how a catalogue of patterns might be extended, and are drawn on later to illustrate how such a catalogue might be used. Note that the intended use of such patterns dictates the detail in which the context needs to be expressed. Systems reengineering patterns are less culturally sensitive than the application of different approaches to managing the reengineering process. Therefore, patterns presented for formal validation and inclusion in a national, or trans-national, corporate catalogue, will necessarily contain greater detail than the illustrative patterns below.

**Name:** War Room

**Status:** Illustrative only.

**Context:** You are dealing with uncooperative vendor. The project is suffering as a result.

**Problem:** How do I get the vendor to tackle the outstanding issues?

**Solution:** Call a war room with all interested external and internal parties. Prioritise issues and agree on actions and time scales.

**Consequences:** By having all parties together, the scale of the problem can be seen. The collective pressure of the group forum can help to isolate the people inhibiting progress and coerce them to commit to actions. To improve relationships, it may be worthwhile suggesting a *Work Shop*.

**Known Uses:** This pattern was used during a *Middleware* implementation.

[Note that in an organisation-specific reengineering pattern catalogue, this section would also contain contact details of managers involved in the cited implementation.]

**Name:** Work Shop

**Status:** Illustrative only

**Context:** You are trying to build a long-term partnership with a supplier on whose support the project's delivery and its long-term success depends. The vendor is uncooperative and is generally unwilling to offer help and answer questions.

**Problem:** How do I get the vendor to tackle the outstanding issues in the short-term and build a better relationship?

**Solution:** Call a 'work shop'. Document and agree on all the business processes that are dependent on the developing system, map out the interfaces directly supported by the vendor's product and identify those which require extensive customisation. Explore whether user requirements can be met by a standard solution and agree action by the vendor on those that cannot.

**Consequences:** By having all parties together, opportunities for rationalising the work programme without compromising flexibility can be explored. The

work shop may free resources for meeting acute requirements and help build communication links that support earlier identification and response to emergent requirements. If project delivery has to meet an external deadline, such as new legislation or the launch of a new product, then it may be worthwhile suggesting a *War Room*.

**Known Uses:** This pattern was used during a *Divide_and Modernise* implementation. [Note that in an organisation-specific reengineering pattern catalogue, this section would also contain contact details of managers involved in the implementation cited.]

## V. USING PATTERNS

The 'managing process' patterns presented in section 4 should be treated with great care. They are not intended to be prescriptive solutions or an 'ABC' of management. For example, whilst the coercion described in War Room can produce immediate compliance it can also have a number of negative side effects [Vecchio, 1991]. Rather, these particular patterns are a way of capturing and presenting knowledge about the process of reengineering which establishes the context in which the reengineering was performed and identifies people within the organisation who were responsible for managing the process. These are the people with whom a broader understanding of the problem and solution domains can be achieved, and tracking these people is an important aspect of knowledge management. The combined activities of using these two different types of pattern to identify a solution, a sequence of activities required to reach the solution, and methods for managing the process of implementing the solution is shown in Figure 10.

Here integration of a particular Legacy system with Enterprise systems leads a system architect to a review a catalogue of reengineering patterns, and one is selected that resolves (aligns) stated technology and business strategies. For example, *Wrapping* (Section 4) may be seen as the lowest cost and quickest technical solution for dealing with a legacy system interface, but it may serve to further obscure the underlying structure of the system and

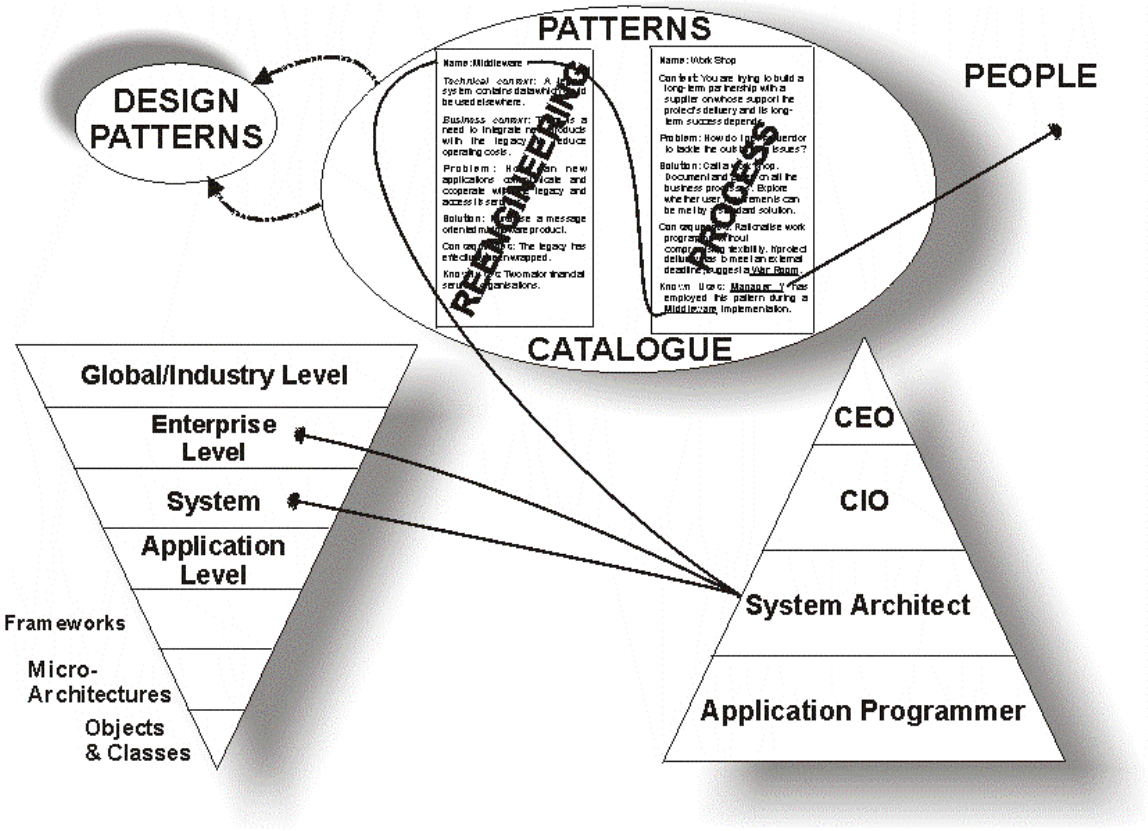


Figure 10. Using a Pattern Catalogue.

(Note that systems reengineering patterns may use, but are not merely, design patterns)

complicate future improvements in flexibility. In terms of the direct impact on business competitiveness, this might represent a trade-off between future *Flexibility* and the immediate requirement to change the presentation of the system to the user in order to enhance the perceived service *Quality*. Note that italics are used here to differentiate characteristics of the business process, such as *Flexibility* to respond to changing market conditions, from those technical characteristics built into the reengineered system,

On the other hand, *Externalising an internal representation* is an example of a pattern that may take some time and effort to accomplish, but yields a more flexible final solution. Furthermore, this pattern provides the added security of stable milestones along the way to allow for any delays in funding the project's completion. This solution may not have an immediate impact on competitiveness, but does build in future *Flexibility* with minimal risk

to a core system, without which *Speed* and service *Quality* may approach zero.

The systems reengineering pattern chosen in turn leads to consideration of management approaches used in previous implementations of that pattern, and to people within the company who have been responsible for managing this process in the past. The dialogue established through these patterns between different domain experts can be used to confirm the validity of the solution in the current context, help establish the composition of the implementation team, and even used to select specific target designs for consideration from a design patterns catalogue.

## VI. CONCLUSIONS

Senior executives should be all too familiar with the specific issues that affect their organisation's competitiveness and the competencies that need to be preserved or developed. They may use this knowledge to help define which of their business processes and technical sub-systems need to be reengineered, but they are unlikely to appreciate the specific benefits that attend one technical system's design versus another. Conversely, system architects and application programmers understand in detail what their technical choices will do to the system's characteristics, but will be less aware of how such change will specifically impact the competitiveness of their organisation. Resolving these two valuable perspectives in a way that maximises the potential for competitive advantage and minimises risk is recognised as a significant management challenge for which no over-all solution exists, but elements of good practice have been identified. Accepted wisdom results in system development that is often guided by objectives 'handed down' from an over-all business strategy, constrained by a technology strategy, and driven by over-all beliefs that certain characteristics, such as integration, are desirable.

The decision to proceed with an information systems project is typically justified on the basis of a positive impact on profitability resulting from reduced operating costs or preserved/increased revenue. On the 'cost' side of the balance sheet, pressures to improve efficiencies increase the coupling

between distinct systems as they evolve through a process of incremental improvement. On the ‘revenue' side, the need to react to changing market conditions to protect and develop future revenue streams leads to the introduction of new systems whose integration establishes further couplings. This organisational ‘intra-structure' complicates the future reengineering of business or technical systems, increases the risks associated with change, and tends to promote incremental approaches to systems reengineering that do little to change the core functionality. Successive revisions tend to obscure the original structure, increase resistance to change, and eventually produce what may be recognised as a ‘legacy' system. These revisions, in turn, may 'lock-in' redundant elements of business processes, reduce an organisation's ability to adapt to market changes and consequently erode competitiveness. Reversing this trend still requires that legacy system reengineering is based on technical choices that are consistent with the organisation’s technology strategy. If reengineering is going to lead to a more competitive business process over-all. it will also require that technical design choices (Figure 4) include consideration of how each option supports the business strategy,

Pattern languages are recognised for their ability to communicate expertise about technical choices and implementation approaches. It may be argued that their use on this basis alone could result in a reengineered system that makes the greatest contribution to generic factors of competition. However, this paper notes that investment decisions guided by considering only generic factors of competition need not lead to any direct improvement in competitiveness for the organisation as a whole. We therefore propose that 'systems reengineering patterns' formally incorporate factors relating to generic competitiveness, such as Speed, Cost, Quality and Flexibility, but use these factors to communicate issues relating to the specific competitive environment faced by the firm. This 'context' can then be used to help select systems reengineering patterns from a catalogue, and the embedded concepts of generic competitiveness used to establish a dialogue between the business and technical domains that resolves the technical and business imperatives.

This pattern formalism is extremely flexible. We demonstrated how it can be adapted to capture knowledge about managing a reengineering project within a specific organisational context, and thereby provide links to people that were previously involved in similar projects and with whom further dialogues can be established. It also provides the potential for links with established systems engineering methodologies, such as design patterns, allowing an extremely efficient re-use of expertise codified for other reasons.

The apparent strengths of this approach for tackling legacy issues within organisations and for capturing the tacit knowledge associated with their reengineering, indicates significant potential for improving the competitive impact of decision-making about expenditure on systems development. As the number of validated systems reengineering patterns grow and the synergistic relationships among them become better understood, the grammar of a systems reengineering pattern language may emerge that could also help improve the sustainability of any resulting competitive advantage.

## ACKNOWLEDGEMENTS

We gratefully acknowledge P. Stevens for many stimulating conversations, the EPSRC for supporting this work (GR/M02491) and helpful comments from the referees.

Editor's Note: This article was accepted by associate editor Christopher P.Holland. He received the manuscript on January 14, 1999 and supervised its peer review. The paper was received by the Editor on November 4, 1999 and was published on December __ 1999.

## REFERENCES

EDITOR'S NOTE: The following reference list contains hyperlinks to World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the paper on the Web, can gain direct access to these linked references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.
2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.
4. the author(s) of this article, not CAIS, is (are) responsible for the accuracy of the URL and version information.

Alexander, C., Ishikawa, S., Silverstein, M. (1977*) A Pattern Language: Towns, Buildings, Construction*, New York: Oxford University Press.

Alexander, C. (1979) *The Timeless Way of Building*, New York: Oxford University Press.

Appleton, B. (1997) "Patterns for Conducting Process Improvement", *Proceedings of PLoP'97*.

Beedle, M. (1997) "Pattern Based Reengineering", *Object Magazine*.

Benjamin, R.I. and M.S. Scott Morton (1988) "Information Technology, Integration, and Organizational Change", *Interfaces*, (18)3, pp.86-98.

Boltho, A. (1996) "Convergence, Competitiveness and the Exchange Rate", in Crafts N. and Toniolo G. (eds.) *Economic Growth in Europe since 1945*, Cambridge: Centre for Economic Policy Research.

Booch, G. (1994) *Object Oriented Analysis and Design with Applications*, 2nd Edition, Benjamin Cummings, Chapter 10.

Brodie, M.L. and Stonebraker, M. (1995) *Migrating Legacy Systems: Gateways, Interfaces and the Incremental Approach*, Morgan Kaufmann Publishers.

Brown, A.W., E.J. Morris and S.R. Tilley (1996) "Assessing the Evolvability of a Legacy System", Pittsburgh, PA:Carnegie Mellon University Systems Engineering Instutitute draft white paper.

Brown, W.J., R.C. Malveau, H.W. McCormick and T.J. Mowbray (1998) *Anti Patterns: Refactoring Software, Architectures, and Projects in Crisis*, New York: Wiley

Brynjolfsson E. and Hitt, L. (1996) "Paradox Lost? Firm-level Evidence on the Returns to Information Systems Spending", *Management Science*, (42)4, pp.541-558.

Calladine, J. (1997) *BT Middleware - Software as Infrastructure*", BT Technology Journal, (15)1, pp.135-146.

Caminer, D., Aris, J., Hermon, P. and Land, F. (1996) "*User-Driven Innovation, The World's First Business Computer*", London: McGraw-Hill.

Cecil, J.L. and Goldstein, M. (1990) "Sustaining Competitive Advantage from IT", *The McKinsey Quarterly*, (1990)4, pp.74-89.

Coombs, R. and Hull, R. (1995) "BPR as 'IT-enabled change' - an Assessment", *New Technology, Work and Society. Special issue on BPR*: September.

Coplien, J.O. (1995) "A Development Process Generative Pattern Language", *Proceedings of PLoP'95*.

Coplien, J. (1997) "Organizational Patterns Web Page", http://www.bell-labs.com/cgi-user/OrgPatterns/OrgPatterns.

Cronin, B., A. Cavaye, and L. Davenport (1988) "Competitive Edge & Information Technology", *International Journal of Information Management,* (8)3, pp.179-187.

Cunningham, W. (1995) "EPISODES: A Pattern Language of Competitive Development", http://c2.com/ppr/episodes.html.

Davenport T.H. and J.E. Short (1990) "The New Industrial Engineering: Information Technology and Business Process Redesign", *Sloan Management Review,* (31)4, pp.11-27.

Dewar, R., Lloyd, A.D., Pooley, R. and Stevens, P. (1999) "Identifying and Communicating Expertise in Systems Reengineering: a Patterns Approach", *IEE Proceedings - Software*, (146)3, June 1999.

Ein-dor, P., E. Segev, D. Blumenthal, and I. Millet (1984) "Perceived Importance, Investment, and Success of MIS: or the MIS Zoo", *Systems, Objectives, Solutions*, (4), pp.61-67.

Gamma, E., R. Helm, R. Johnson, J. Vlissides (1995) *Design Patterns*, Reading, MA: Addison Wesley,.

Hammer and Champy (1993) *Re-engineering the Corporation - a Manifesto for Business Change.* London: Nicholas Breasley.

Harrison, P.F. (1997) “Customer Service System - Past, Present and Future”, *BT Technology Journal*, (15)1, pp. 29-45.

Heany, D.F. (1968) *Development of Information Systems*, New York: Ronald Press.

HMSO (1994) “The Proper Conduct of Public Business”, *Committee of Public Accounts Paper 154 HMSO*.

IBM (1998) “Case Study: Prudential Ensures Rock Solid Future with MQSeries”, *http://www.software.ibm.com/ts/mqseries/* page created 26 February 1999.

Jacobson, I. and F. Lindstrom (1991) "Re-engineering of Old Systems to an Object-Orientated Architecture", *Proceedings OOPSLA ‘91, ACM SIGPLAN Notices*, (26)11, pp.340-350.

Kempis R-D. and J. Ringbeck (1998) "Manufacturing’s Use and Abuse of IT", *The McKinsey Quarterly*, (1), pp.138.

Kirkpatrick, G. (1994) “Wealth Creation from Information Technology”, *The 1994 UK IT Forum*.

Lambert R. and J. Peppard (1993) "Information Technology and New Organizational Forms: Destination but no Road Map?", *Journal of Strategic Information Systems*, (2)3, pp.180-205.

A.D. Lloyd, A.M. McCosh, A.U. Smart and P.R.N. Barrar (1997) “Technology, Innovation and Competitive Advantage: A Business Process Perspective” BAM97, The British Academy of Management Annual Conference, London.

Manchester, P. (1999) “Y2K: Killing Two Birds with One Stone” *Financial Times,* Wednesday May 26.

Newswire (1999) “Analysis: SAP Looks to Users - But is it too Late?” *PC Week*, 18 May, pp 13.

OECD (1988) “New Technologies in the 1990s - A Socio-Economic Strategy.”, OECD.

Peppard, J. (1996) “Broadening Visions of Business Process Re-engineering”, *OMEGA* (24)3, pp. 255-270.

Phanouriou C. and Abrams M. (1997) “Transforming Command-Line Driven Systems to Web Applications”, *Computer Networks AND ISDN Systems*, (29)8-13, pp.1497-1505.

Pooley R.J. (1991) "The Integrated Modelling Support Environment", in G. Balbo and G. Serazzi eds. *Computer Performance Evaluation - Proceedings 5th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pp 1-16, Turin, February 1991, Amsterdam: North Holland,

Ransom, J., I. Sommerville and I. Warren (1998) "A Method for Assessing Legacy Systems for Evolution", *Proceedings of the Reengineering Forum*.

Sale, A.E. (1999) “The Rebuild of Colossus: The World’s First Computer”, Bletchley Park, Milton Keynes. www.cranfield.ac.uk/ccc/bpark/colossus.htm

Skinner, W. (1969) “Manufacturing - Missing Link in Corporate Strategy”, *Harvard Business Review*, May-June, pp 136.

Sneed, H.M. and R. Majnar (1998) “A Case Study in Software Wrapping”, *ICSM’98*, pp.86-93.

SRAH (1997) "Software Reengineering Assessment Handbook v3.0", Reengineering Home Page, http://stsc.hill.af.mil/RENG/.

Stevens, P. and R. Pooley (1998) "Systems Reengineering Patterns", *ACM-SIGSOFT*, F*oundations of Software Engineering*, (6), pp 17-23.

Strassmann, P.A. (1990) *The Business Value of Computers*, New Canaan, CT: Information Economics Press.

Swenson, D.W. and J. Cassidy (1993) “The Effect of JIT on Management Accounting”*, Cost Management*, Spring.

Talwar, R. (1993) "Business Re-engineering - a Strategy-Driven Approach", *Long Range Planning*, (26)6, pp.22-40.

Vecchio, R. P. (1991) *Organizational Behavior*, Orland, FL:The Dryden Press.

Willcocks, L. P. and Lester, S. (1997) “In Search of Information Technology Productivity: Assessment Issues”, *Journal of the Operational Research Society*, (48), pp 1082-1094.

## ABOUT THE AUTHORS

**Ashley D. Lloyd**: After completing a PhD in Physics and an MBA, Ashley joined The University of Edinburgh Management School as the Lecturer in Information Management. His current research interests focus on Technology, Business Processes, and their impact on Competitive Advantage, with projects funded by the EU, SHEFC, ESRC, EPSRC and Industry. (http://omni.bus.ed.ac.uk/adl/)

**Rick Dewar**: After several years with Rolls-Royce, Rick took up a research post at Heriot-Watt University before joining the Division of Informatics as a research fellow at the University of Edinburgh. (http://www.dcs.ed.ac.uk/~rgd)

**Rob Pooley**: Rob is a Professor in the Department of Computing and Electrical Engineering at Heriot-Watt University. His interests in software engineering began with object oriented programming in SIMULA in 1978. He worked for several years writing compilers at the University of Edinburgh, before joining academic life. Together with Perdita Stevens, he helped develop the ideas for Reengineering Patterns. (http://www.cee.hw.ac.uk/~rjp/)

Copyright ©1999, by the Association for Information Systems. Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and full citation on the first page. Copyright for components of this work owned by others than the Association for Information Systems must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or fee. Request permission to publish from: AIS Administrative Office, P.O. Box 2712 Atlanta, GA, 30301-2712 Attn: Reprints or via e-mail from ais@gsu.edu

Communications of the Association for Information Systems

**EDITOR**
**Paul Gray**
**Claremont Graduate University**

## AIS SENIOR EDITORIAL BOARD

| | | |
|---|---|---|
| Henry C. Lucas, Jr.<br>Editor-in-Chief<br>New York University | Paul Gray<br>Editor, CAIS<br>Claremont Graduate University | Phillip Ein-Dor<br>Editor, JAIS<br>Tel-Aviv University |
| Edward A. Stohr<br>Editor-at-Large<br>New York University | Blake Ives<br>Editor, Electronic Publications<br>Louisiana State University | Reagan Ramsower<br>Editor, ISWorld Net<br>Baylor University |

## CAIS ADVISORY BOARD

| | | |
|---|---|---|
| Gordon Davis<br>University of Minnesota | Ken Kraemer<br>University of California at Irvine | Richard Mason<br>Southern Methodist University |
| Jay Nunamaker<br>University of Arizona | Henk Sol<br>Delft University | Ralph Sprague<br>Universityof Hawaii |

## CAIS EDITORIAL BOARD

| | | | |
|---|---|---|---|
| Steve Alter<br>University of San Francisco | Barbara Bashein<br>California State University | Tung Bui<br>University of Hawaii | Christer Carlsson<br>Abo Academy, Finland |
| H. Michael Chung<br>California State University | Omar El Sawy<br>University of Southern California | Jane Fedorowicz<br>Bentley College | Brent Gallupe<br>Queens University, Canada |
| Sy Goodman<br>University of Arizona | Chris Holland<br>Manchester Business School, UK | Jaak Jurison<br>Fordham University | George Kasper<br>Virginia Commonwealth University |
| Jerry Luftman<br>Stevens Institute of Technology | Munir Mandviwalla<br>Temple University | M.Lynne Markus<br>Claremont Graduate University | Don McCubbrey<br>University of Denver |
| Michael Myers<br>University of Auckland, New Zealand | Seev Neumann<br>Tel Aviv University, Israel | Hung Kook Park<br>Sangmyung University, Korea | Dan Power<br>University of Northern Iowa |
| Maung Sein<br>Agder College, Norway | Margaret Tan<br>National University of Singapore, Singapore | Robert E. Umbaugh<br>Carlisle Consulting Group | Doug Vogel<br>City University of Hong Kong, China |
| Hugh Watson<br>University of Georgia | Dick Welke<br>Georgia State University | Rolf Wigand<br>Syracuse University | Phil Yetton<br>University of New South Wales, Australia |

## ADMINISTRATIVE PERSONNEL

| | | |
|---|---|---|
| Eph McLean<br>AIS, Executive Director<br>Georgia State University | Colleen Bauder Cook<br>Subscriptions Manager<br>Georgia State University | Reagan Ramsower<br>Publisher, CAIS<br>Baylor University |