MANAGING ORGANISATIONAL CHANGE: THE IMPACT OF INFORMATION SYSTEM DEVELOPMENT METHODS[1]

Susan Gasson,
Operational Research & Systems,
Warwick Business School, University of Warwick
Coventry, West Midlands CV4 7AL

Key Words: Management of change, I.T. impact, methods

Abstract

Those aspects of development which are controlled as part of most I.S. development methods are not those aspects which contribute most to organisational support and flexibility. A framework for the selection of development methods and their application, which considers the organisational impact of the resulting information system, is discussed.

Introduction

The impact upon organisational effectiveness of different approaches to the development of information systems is a subject widely seen within the academic community as of critical importance to the success of organisations, yet managers and developers of organisational information systems are largely unaware of these concerns. Information systems development is seen, within organisations, as a primarily technical task. The functional needs of the work group are "analysed" by a technical systems analyst and a computerised system is specified and designed which enables the required functions to be performed optimally. This is the theory.

However, this approach totally ignores the fact that organisations consist of a collection of human beings, engaged in the activities which enable the organisation to function. Land & Hirschheim (1983) define an information system as a social system, which may or may not use information technology to support its operation. Galliers (1987) defines the concept of information as "that collection of data which, when presented in a particular manner and at an appropriate time, improves the knowledge of the person receiving it in such a way that they are better able to undertake a particular activity or make a particular decision". Galliers further makes the point that information does not necessarily arise from a designed information system but may derive from informal sources.

This, then, is our information system: a collection of human beings, using data which may be provided by informal means, as well as by "designed" information systems, to better undertake activities or make decisions. The design of formal (i.e. computerised or document-based) information systems is only *part* of the task of supporting this "information system". The other part is to design a work system which provides opportunity for all the various types of work activity to take place and permits, not prevents, the integration of human beings (each of whom has differing abilities, experience and ways of understanding information) and work activities in an "information system".

---

[1] Published in Conference Proceeedings of British Computer Society Special Interest Group on Information Systems Methodologies, Heriot-Watt University, UK, Sept. 1994

With this understanding, the chief problem of developing an information system moves out of the technical sphere: it becomes the problem of predicting and managing changes in practice imposed upon human beings in order to make them more effective (not just more efficient - you can do the "wrong" thing very efficiently indeed!). Changes to the provision of information within the social system which we call "work" will lead to changes in the balance of power: current holders of organisational power are likely to resist this. Anyone who has ever managed a group of people will vouch for the fact that their biggest management problems arise from the nature of the people whom they manage. People are awkward, and rightly so. It is the ability of human beings to question decisions and to test "evidence" by applying their own perceptions which makes them invaluable to the organisation.

Human beings are a greater source of error-detection and correction than any computer system can possibly be. A computer system has, embodied within it, all of the assumptions and perceptions of its designer. Those assumptions and perceptions may be inappropriate; they may also be wrong. Thimbleby (1991) quotes a personal experience: he, with a colleague, rented a cabin in the U.S. for a short holiday. When they arrived, at about 11 p.m., they found the power turned off. On turning on the power, they discovered that the microwave oven needed to have its clock set before it would operate, yet there were no instructions on how to do this. As they were hungry, they spent a great deal of time attempting to set the clock, but to no avail - they could not operate the oven clock-setting mechanism. The following morning, they telephoned the owner of the cabin and discovered that they could not set the clock because they had been using a 24-hour time setting and all appliances in the U.S. work with a 12-hour clock. This design failed to be appropriate in all circumstances because the designer of the microwave had based his model of its use on his own context - this is a very common problem when design methods do not question the context of use (Booth, 1989).

Human-beings can change their frame of reference to adapt to changing circumstances; Argyris & Schon (1978) refer to this as "double-loop learning". This concept is illustrated in figure 1. An organisation, through its normative work-processes repeats a "recipe for success" which it has learned from experience. The outcome of this work loop is evaluated by a monitoring and control mechanism, in the case of the organisation, this is management sanction. When environmental conditions change, the control mechanism (management) must detect that the outcomes from work processes are no longer appropriate and cause the organisation to shift to a new operating loop, with more appropriate "recipes for success".

The shift to another loop of operation, is seen as difficult for humans, as people in any work-group develop norms and roles relative to each other which become fixed. This adaptive behaviour is seen as so desirable, in the turbulent product-markets of the 1990s, that there is a constant quest for what makes an organisation a "learning organisation". If it is difficult for humans to display adaptive behaviour in organisations, it is even more difficult for computer-based technology, which operates from a pre-programmed set of rules. Even when systems are programmed to be adaptive (e.g. "learning" systems), they still operate within a frame of reference which embodies that of the designer and so are unable to cope with unforeseen circumstances.

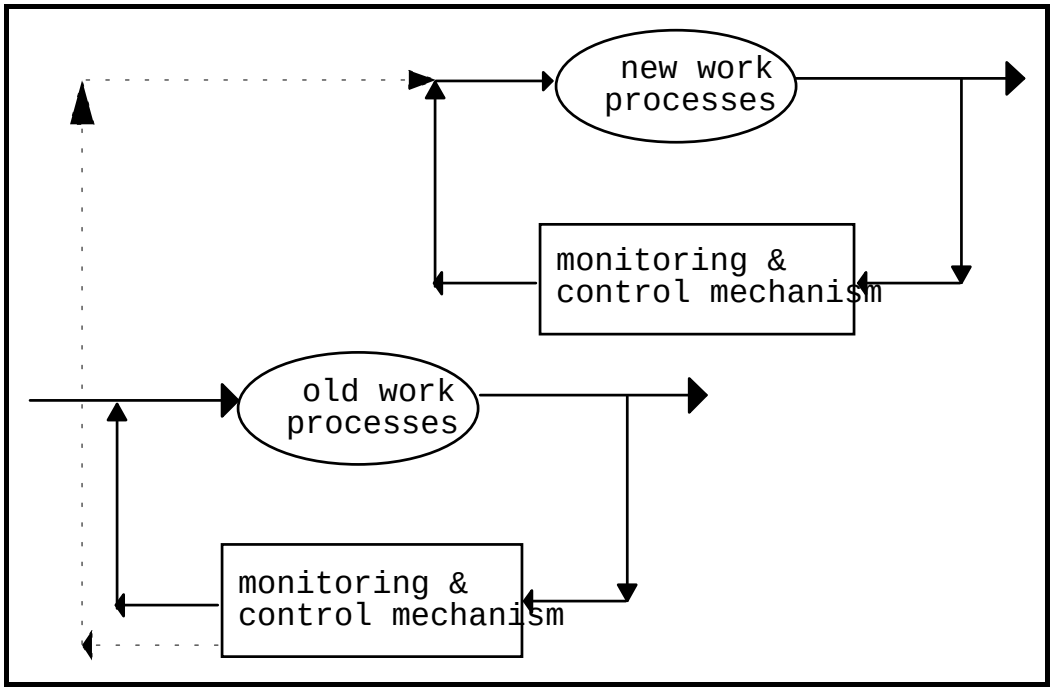


Figure 1: The Concept of Double-Loop Learning
(Source: derived from Argyris & Schon, 1978)

Designing for adaptive behaviour is central to any effective approach to information system development. A *human-centred* approach to information system development sees human beings as a source of error-detection, whereas a *technology-centred* approach to information system development sees human beings as a source of error, because they are "imperfect" and make mistakes. However, the desire to predict system behaviour from a technology-centred perspective may be detrimental to an organisation's survival in a rapidly-changing economic or political environment, where adaptive behaviour is needed for the organisation's survival.

An effective information system needs to support a set of activities which enable human beings to most effectively achieve the changing objectives of the organisation, *supported* by computer-based information technology (I.T.) - rather than dominated by it. If human beings, rather than Information Technology, are seen as the core of the information system, the effectiveness of that system must depend upon the motivation and satisfactions of those human beings, not upon optimisation of the computer-based technology.

A Framework To Consider The Impact Of I.S. Development Methods

Because information systems development is seen, throughout the organisation, as managing change to a *technical* system, rather than the *social* system of work, development methods are normally selected by I.S. professionals, rather than being the subject of more general managerial choice. Development methods emphasise technical optimisation, as this is the basis of I.S. professionals' power (Markus & Bjorn-Andersen, 1987). There is evidence that this emphasis leads to adverse effects upon the organisation, such as the fragmentation of jobs (Corbett et. al., 1991), poor system usability (Hedberg & Mumford, 1975) and inappropriate match to work tasks (Heller, 1987).

The choice of development methods reflects an underlying philosophy of system design, which can affect a wide range of organisational factors. A framework for the selection of development methods and their application is presented in figure 2.

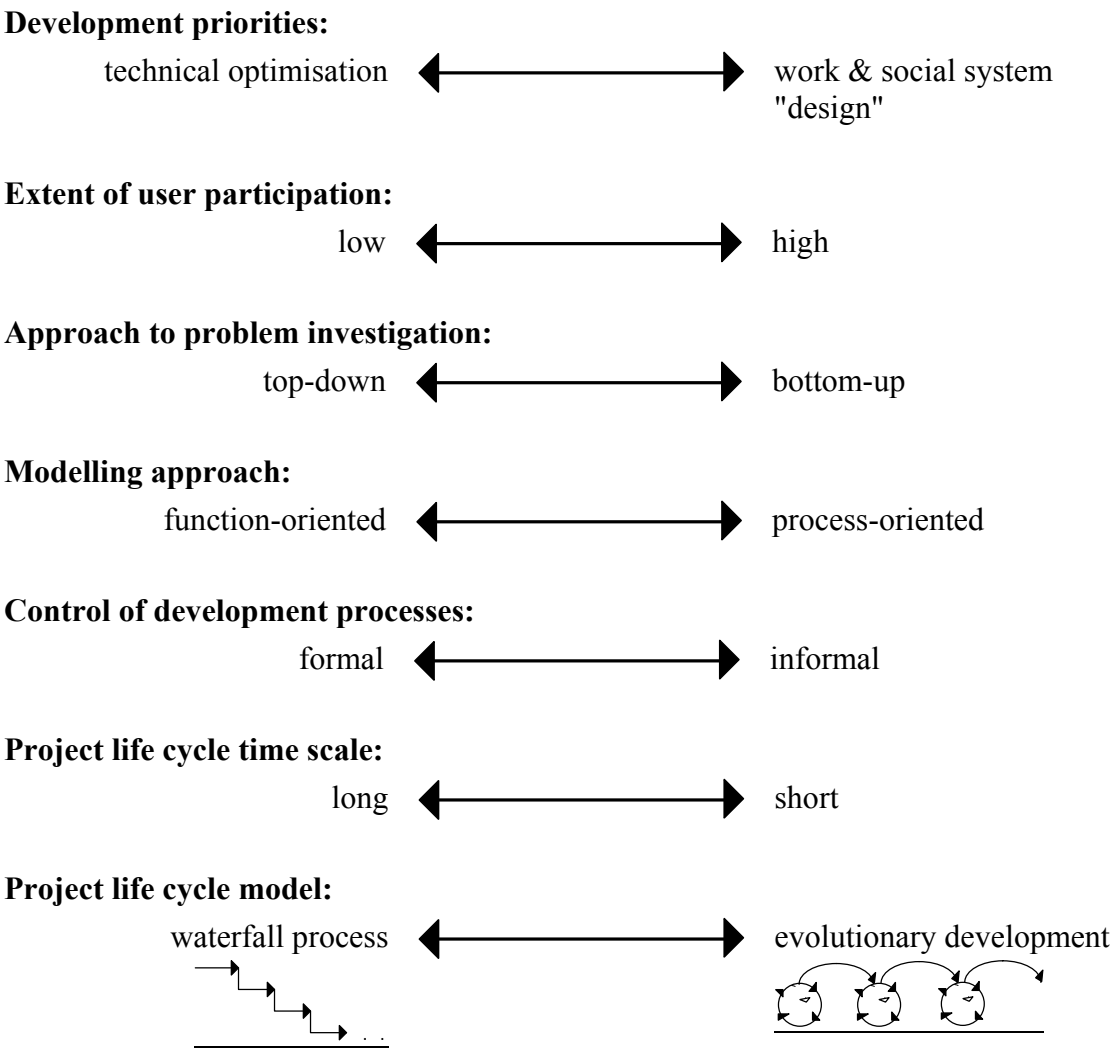


Figure 2: Framework For The Selection of Development Methods & Their Application

The considerations which are important when applying the framework are:

Development Priorities

The question of whether the information system is technically optimised, socially optimised, or optimised for socio-technical use, is central to the system's effectiveness. For example, the initial computerisation of the Driver Vehicle Licensing Centre, at Swansea, split the previously integrated work of clerical officers, who were previously based at local licensing offices, into three distinct jobs: dealing with licence applications, answering telephone queries, and dealing with written correspondence. Contrary to popular report, after an initial teething period, the system performed to its specification; it was the design of the *organisational* part of the system which was deficient, not just its technical implementation (Heller, 1987).

There were inadequacies in the computer system which made it difficult to obtain information on a licence application which had already been processed but not despatched and the pressure on those officers dealing with telephone queries was intense, leading to high employee turnover. Organisational psychologists were engaged to suggest redesign of the work system and a job-rotation scheme was introduced. However, this social optimisation caused other problems to the work-process because the layout of the computer-system meant that a great deal of disruption was caused to work-processes by changes of staff. Eventually the complete work system was redesigned upon a basis which attempted to optimise both social and technical factors (although Heller points out that this design was still deficient as it relied for implementation upon engineers, who installed the telephones in a way which separated their operators from the other work areas).

Hornby et. al. (1992) found that the most common development methods in use have a technical, rather than a social, orientation; developers did not see it as part of their job to consider human and social aspects of design. They additionally concluded that there is evidence that development methods affect how the developer behaves and how they see the world. The selection of a method based solely on technical system optimisation is therefore inappropriate to the design of an organisational information system.

Extent Of User Participation

The choice of IS professionals is, in general, to select IS development methods which exclude users from active participation in system specification and development, as these methods give them greater control over the process (Markus & Bjorn-Andersen, 1987). Where user-participation is used, the experiment may not be repeated, as technical managers cite longer development time scales and higher cost as a reason for project "failure". However, in any work system, priorities are determined by the way in which performance is measured. In a study of IS project failure (Gasson, Ormerod & Martin, 1994), it was found that only half of the respondents ever conducted any sort of formal investigation into the reasons for a project's failure. The other half left IS developers to draw their own conclusions about reasons for failure. Of course, post-rationalisation occurs in any post-mortem procedure, but this figure is worrying because those who define the criteria for success can impose their own value-system on the processes and nature of work; if no investigation is performed into reasons for failure, these value-systems remain unchallenged.

The organisation, in terms of management external to the IS department, was not challenging the IS perspective, or learning from organisational mistakes in the cases reported in our survey. Additionally, organisations do not evaluate how successful a completed development was (Land & Hirschheim, 1983). Those aspects of development which are traditionally used to give performance measures - development to time and to cost - are not those which actually contribute most to *organisational* support and flexibility. The hidden costs of user exclusion are less usable and less appropriate systems (Hirschheim, 1986).

Approach To Problem Investigation

The top-down approach is the epitome of scientific reductionism: a system solution to an organisational problem is investigated by breaking it into sub-problems, which in turn are broken down until a level is reached at which a computer-based solution can be identified for each of its components. At each stage of problem decomposition, issues which are considered to be extraneous to the technical requirements, such as user and organisational requirements, are excluded from the developer's mental model of the system requirements (Corbett et. al., 1991).

The set of sub-solutions are amalgamated into system components on the basis of shared sets of data, or shared I.T. components such as external communication lines. The job of the system component user becomes a set of disparate tasks which have arisen from this "logical" assembly of tasks into a system component; these tasks may have no significance to that user as a meaningful piece of work or as a "complete" job.

An example of scientific reductionism in work design is the way in which computer-aided manufacturing is implemented in many factories. A skilled machine-tool lathe operator has skills which enable them to judge the tolerance of a part to a thousandth of an inch, when operating the lathe manually. When computer-controlled lathes are introduced, instead of building on existing skills, the designers separate the programming tasks into separate jobs, turning the job of operating the lathe into one of loading a metal block onto the lathe, pushing a button to start, then to stop, the process and unloading the finished part. As the operator now has much less work to do, they are commonly given several machines to operate instead of one; they have no connection with, or control over the machining process and so become alienated from their work, continuing operation even when the parts are being produced incorrectly, as they no longer relate their work to the detection of problems.

The bottom-up approach starts from a pre-conception of a system solution, based upon the system designer’s educational background and previous experience. If the designer is an IS professional, whose training and organisational importance derive from their technical expertise, they are likely to also take an approach based upon scientific reductionism of the problem (albeit less explicitly than the top-down approach) as their solution will be based upon previous systems which have been technically optimised, rather than based upon holistic work processes. An organisational psychologist, on the other hand, will derive system solutions based upon holistic work-processes, which require a minimal technology base, as they will not normally possess technical expertise.

For a system which provides effective information-handling as well as effective work-processes, the system designer must possess skills which enable them to use both perspectives of a system. Alternatively, cross-disciplinary development teams can be used, although this approach brings the problems of managing discrete sets of people with conflicting objectives for the system.

It can be seen that, although a top-down approach alone is inappropriate for the design of a system which will ensure organisational effectiveness, a bottom-up approach is also not a sufficient guarantee of this. Ideally, top-down and bottom-up approaches should be combined with a perspective of the system as a social work system which uses technology to support its processes and operations.

The Approach To System Modelling

System modelling can be performed in two ways, depending upon how the system is perceived. At one end of the spectrum, the system requirements are presented as a set of **functions** which must be performed by the system to meet its objectives. Each function is decomposed using the top-down approach described above; the result is a set of data-entities needed for those functions and a set of associated data-flows and computer-processes which will form the basis for system components, with the ensuing job-fragmentation impact previously described.

The other end of the spectrum (process-oriented) takes a much more holistic approach: the system requirements are presented as a set of human **work processes** which meet the objectives of the system. These processes are derived using holistic task analysis methods; the information requirements of each process are then analysed and the information flows are amalgamated to provide computer support for the required work processes. The resulting jobs are meaningful to the human system-user as they have been designed around a set of meaningful, interrelated tasks and work processes. Jayaratna (1988) argues that process-oriented methods are appropriate for less well-structured organisational contexts. It can be argued that, as organisations have to increasingly respond to highly complex and turbulent product-market environments, all organisational contexts are becoming less well-structured and a function orientation is no longer appropriate.

Control Of Development Processes

The management control exerted over the development team also affects the organisational impact of the system. A structured, formal development process is document-driven: the end of each phase of development is determined by the validation of a document representing the output of that phase, by a group normally consisting of developers and user-representatives (who may not be the actual intended users of the system). These user-representatives do not normally possess sufficient technical expertise to validate technical specification and/or design documents; they rely on verbal interpretations and signals given at formal validation meetings to interpret the system design.

There is evidence (Curtis et. al, 1988) that formal development processes do not permit the degree of interaction between developers and users necessary for members of the development team to gain sufficient understanding of the application domain. Developers have been observed to subvert the formal processes of development (i.e. walkthrough and validation meetings) for the purposes of informal discussions about issues unrelated to that part of the design under discussion, so that they can gain an understanding of wider system requirements and of user task perceptions. However, the perception of technical designers is that these opportunities are not sufficient to enable them to develop a real understanding of the application domain.

An informal development process permits much wider interactions, both between technical developers and users, and also between technical developers whose system components may not be directly related as part of the logical design of the system. These informal discussions permit designers to gain a wider understanding, both of the application domain and of general system design issues; it is a major deficiency of formal development processes that learning (about the application domain, about alternative work-processes, about previous types of solution to this type of problem and about the processes of design) is not seen as a legitimate activity (Gasson, 1993). The extent to which managers attempt to control the activities undertaken as part of the development process is inversely proportional to the extent to which such learning can take place.

Project Life Cycle Time Scale

Associated with the formality of the development process is the time scale of the project: both its duration and how activities are managed with respect to projected time scales. A long time scale inevitably leads to the "big bang" phenomenon: such a high level of resources have been invested in the development project that managerial expectations of its successful outcome will be very high and there will be a reluctance to commit further resources to address any shortcomings in the system design. The result is what Eason (1982) calls “fire fighting” - software maintenance fixes, which do not address design deficiencies (and so the root causes of system problems), but which make the system usable in the short term.

A short time scale can affect expectations within the organisation in one of two ways: either the system is perceived as non-strategic, in which case expectations of its organisational impact will be low and any benefits will come as a pleasant surprise, or else the project will be perceived as evolutionary in its impact. An evolutionary project (also known as a phased roll-out) is much more beneficial in terms of organisational learning (see figure 1). Even when structured methods are used to control the development, the effect of the project is much nearer to that of a prototyping approach, as the outcome of a particular cycle of the evolution is not seen as the end of the system life cycle but merely a stage in its development.

There are other trade-offs to be considered in the impact of system development project time scales, which are more related to the issue of project control than of the time scale itself. A short project time scale may bring with it more formal control measures: a two-month slip on a project of six months duration is much more embarrassing to the project manager than a two-month slip on a project of two years duration. Formal project control, as discussed above, may mean a less effective system outcome; a system development is only evolutionary if system developers are allowed the space to learn from the outcome of the previous system evolution.

Project Life Cycle Model

All of the issues above can be brought together in a consideration of the process model of the project life cycle which is supported by the development method in use. A method based upon the “waterfall” model, where each stage of the development process is considered complete before the next stage begins necessitates formal control measures, longer time scales, and a more function- and data-oriented approach than does one based upon an evolutionary model.

When using a waterfall life cycle model, developers are not answerable to the user: with a long development time scale, developers are usually working on another system by the time users appreciate the organisational impact of the previous one, so little feedback is received by the designers of the system about its organisational outcome. Such methods can be seen to be top-down in their approach to problem investigation: requirements must be fully determined before a requirements specification can be "signed off" and the system design can begin. The method's very nature excludes users from development processes: user-contact with the system design is limited to the validation of documents which they have a very low chance of understanding. By the time there is a concrete example of system functions (normally at the system testing stage), the system design will have become fixed - all the user can affect at this stage are cosmetic "user-interface" aspects of the system. Because the development project is managed by conformance to timed delivery of each phase's validated document, there is a concentration upon the technical aspects of the system: these, after all, are what are validated in the document output.

An evolutionary development method, on the other hand, necessitates a much more holistic approach to system design, which engages much more in a consideration of its organisational impact. The development process can be likened to the progress of a Catherine Wheel: as each cycle ends, another cycle begins. Development time scales are shorter, so the design can evolve rather than being determined "for ever" in the early stages of the development lifecycle. The process is more participative: even when "structured" development methods are used, user feedback on one evolution of the system will affect the design of the next system evolution. It is important, for this process that each cycle ends with an evaluation of the system in its organisational context. Because developers are then answerable, in a relatively short time scale, for adverse organisational impacts, there is more incentive for designers to consider organisational and social aspects of the design, as they proceed.

The Impact Of Development Methods Upon Organisational Work

It is instructive to map existing development methods onto the model given in figure 2; this is done in figure 3. Along the extreme left of the model's spectra lie the traditional, structured methods of development. Along the extreme right of the model lie the methods used for end user development, which are similar to a user-centred version of the code and fix approach described by Boehm (1988); these methods may be as inappropriate to many contexts as structured, technical development methods are. Readers should note that the placement of various methods are derived from the author's experience - others may place these methods differently. It is the *process* of using the framework that is important: applying thought to the likely impact of the use of alternative methods in one's own organisation.

The framework represented in figure 2 has been presented here as an aid for the selection and application of methods; many methods can have varying impacts depending upon their application; two examples of this are presented in figure 3.

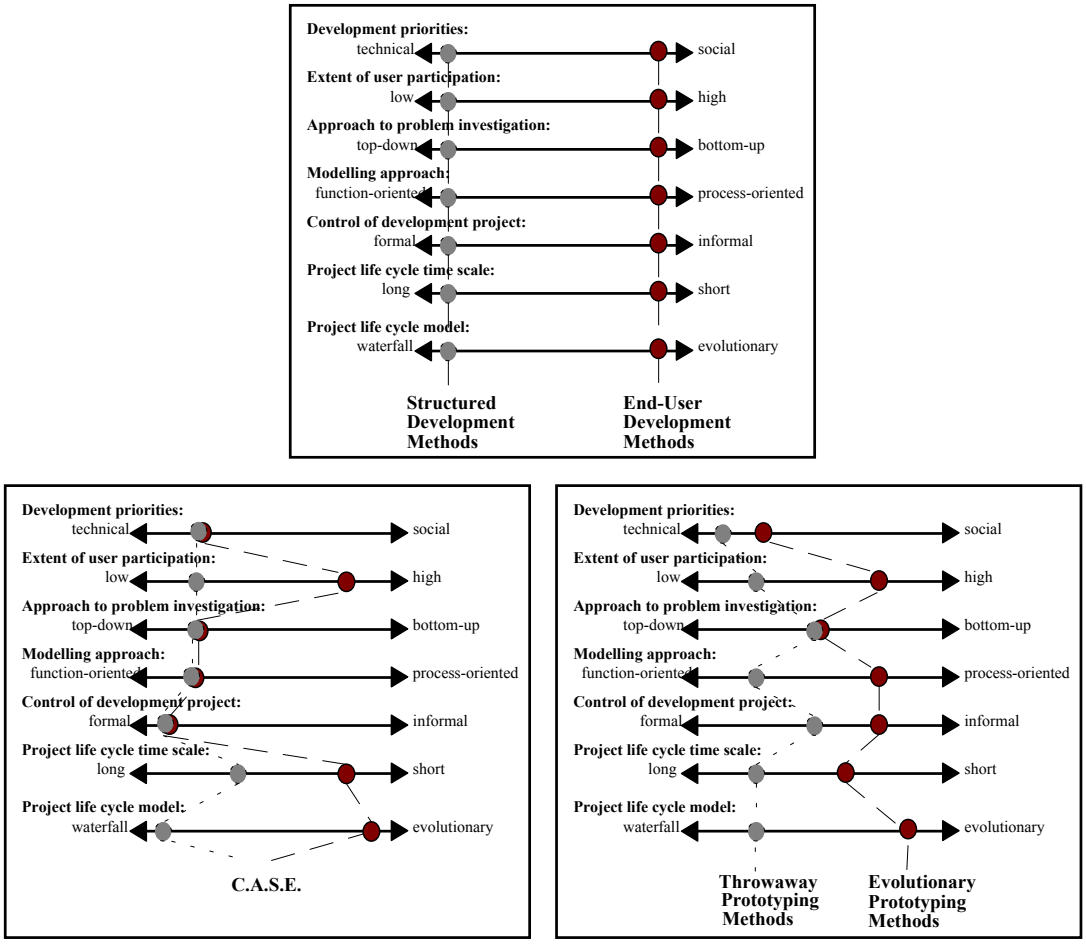


Figure 3: Using The Framework To Map Development Methods

Computer-Aided Software Engineering (CASE) can permit varying degrees of user participation and be used as the basis for both "big bang" projects with long time-scales and evolutionary development projects. Similarly, it can be seen that prototyping does not guarantee user participation: prototyping may be used to elicit user feedback as part of an evolutionary development process, or it may be used to try out a functional "experiment" - this may or may not involve users. The author has knowledge of a project where this "throwaway" prototyping was used as a device to keep users occupied while the technical developers evolved a completely different system design! Therefore, when selecting development methods, managers need to consider both their underlying philosophy and their application.

Technical change can have an impact at three levels within the organisation: the organisation as a whole, the work group and the individual. At the level of the **organisation**, the scope, purpose and information requirements of a system need to be defined flexibly, to support the "learning organisation" of the 1990s. This can best be done by seeing system development as a evolutionary process, where designers are involved in two ongoing dialogues: with users about the organisational effectiveness of previous system evolutions and with organisational managers about the strategic and business information needs of the organisation.

The organisation consists of a set of human beings, performing integrated and inter-related activities which use information as a basis for action and decision-making; work activities are often motivated by organisational politics, rather than by a desire for effectiveness. Changing the system of work will change individuals' responsibilities and their access to, and need for, information and will thus change the balance of power within the organisation. Such changes may well be resisted, as in the recent London stock-market computerisation, leading to ineffective systems.

The process of development is as much a process of ensuring managerial ownership of and commitment to the changes arising from new information systems, as it is of developing a technical system which will supply the information required. Development methods cannot be expected to cater for these aspects of development: the selection and training of project managers for political skills is more important than the selection of a development method.

At the level of the **work-group**, information system developers must consider support for *informal* information systems and for communication between group members, as well as the technical and information requirements of formal information systems. If an information system is seen as a social system, supported by technology, then computer support for information systems needs to be based upon an analysis of the human-activity system. This means selecting development methods which provides an opportunity for the analysis of work processes: a complete system design will specify what process changes need to take place for a desired outcome, as well as what the information requirements of that outcome are. This enables the political commitment needed for a successful system outcome to be obtained early in the process: the organisational changes consequent upon the system implementation are both explicit and agreed in advance.

At the level of the **individual**, there needs to be a positive effort, on the part of managers and system developers, to see people as a source of error detection and correction, not as a source of error in the organisation. In this way, their skills, experience and high-levels of motivation will contribute positively to the organisation's effectiveness. Individual user contributions should be sought as an input to the development process: this means selecting development methods which encourage short, evolutionary project life cycles and permit high levels of informal dialogue between users and developers.

The organisational impact of an information system is central to the success of that system in supporting the work of the organisation. It is strange, then, that it is either ignored by development methods, or consigned to an initial “feasibility study” stage of system development. Users are not powerless in the organisational processes by which development methods are selected but it is not seen as legitimate that functional managers and their staff engage in the selection and the organisational application of information system development methods. Functional managers have at least as much expertise in the management of political change as technical managers - the challenge is for functional managers to perceive it as legitimate to ensure that system development methods provide explicit consideration of organisational impact. What is needed is for functional managers to realise this challenge and to actively participate in those decisions on system development which have such a high impact on the organisation.

References

Argyris, C & Schon, D.A. (1978), Organizational Learning: A Theory of Action Perspective, Addison-Wesley

Boehm, B.W. (1988), "A Spiral Model Of Software Development And Enhancement", IEEE Computer Journal, May 1988

Booth, P. (1989), "An Introduction to Human-Computer Interaction", Lawrence Erlbaum Associates

Corbett, J.M., Rasmussen, L.B. & Rauner, F. (1991), Crossing the Border: The social and engineering design of computer integrated manufacturing systems, Springer-Verlag, London

Curtis, B., Krasner, H. and Iscoe, N. (1988), "A Field Study Of The Software Design Process For Large Systems", Communications of the ACM, Vol. 31, No. 11, Nov. 1988

Eason, K.D. (1982), "The Process Of Introducing Information Technology", Behaviour and Information Technology, Vol. 1, No. 2, April-June 1982. Reprinted in Paton, R., Brown, S., Spear, R., Chapman, J. & Hamwee, J. (Eds.), Organisations: Cases, Issues and Concepts (1984), Paul Chapman Publishing, London

Galliers, R.D. (1987), "An Approach to Information Needs Analysis", in Galliers, R.D. (Ed) Information Analysis, Addison-Wesley

Gasson, S. (1993) “A Behavioural Analysis Of Models Of The Information Systems Development Process”, Conference on the Theory, Use and Integrative Aspects of IS Methods, BCS Information Systems Methods Specialist Group, Heriot-Watt University, Edinburgh, September 1993

Gasson, S., Ormerod, R., Martin, A. (1994), "Report On A Survey of IS Project Failure", internal report, Information Systems Research Unit, Warwick Business School, University of Warwick

Heller, F.A. (1987), "The Technological Imperative And The Quality Of Employment", New Technology, Work and Employment, 1987

Hedberg, B. and Mumford, E. (1975), "The Design Of Computer Systems: Man's Vision Of Man As An Integral Part Of The System Design Process", in Mumford, E. and Sackman, H. (Eds.) Human Choice and Computers, North-Holland, Amsterdam

Hirschheim, R.A. (1986), "Participative Systems Design: User Experiences, Evaluation and Conclusions", The Australian Computer Journal, Vol. 18, No. 4

Hornby, P., Clegg, C.W., Robson, J.I., Maclaren, C.I., Richardson, S.C., O'Brien, P. (1992) "Human and Organizational Issues In Information Systems Development", Behaviour & Information Technology, Vol. 11, No. 3

Jayaratna, N. (1988), "Guide To Method Understanding In Information Systems Practice", International Journal of Information Management, Aug. 1988

Land, F. and Hirschheim, R. (1983), "Participative Systems Design Rationale, Tools and Techniques", Journal Of Applied Systems Analysis, Vol. 10.

Markus, M.L. & Bjorn-Andersen, N. (1987), "Power Over Users: Its Exercise By System Professionals", Communications of the ACM, June 1987, Vol. 30, No. 6

Thimbleby, H. (1991), "Where Computer Science Fails Us", Research seminar addressed to the Department Of Computer Science at Aston University, Birmingham.