Managing Information Systems Construction

Peter Middleton, (E-mail: p.middleton@qub.ac.uk), The Queen's University of Belfast

Abstract

This paper examines the efficiency and effectiveness of a prescriptive systems development methodology in practice. The UK Government's mandatory Structured Systems Analysis and Design Method (SSADM) was examined to determine its value to software projects. The evidence was collected from interviews with 17 project managers, discussions with participants on 3 large SSADM projects and from observing 90 end users in training. The conclusions are that prescriptive information systems methodologies are unlikely to cope well with strategic uncertainty, user communication or staff development. The recommendations are to focus more on soft organisational issues and to use approaches tailored to each project.

Introduction

This research examines the use of prescriptive methodology as a way of improving software quality. It does this by evaluating how effective the UK Government's Structured Systems Analysis and Design Method (SSADM) is in raising the performance of software developers. The idea of defining a way to develop software successfully and then training people to follow it is an attractive one.

The arguments for a methodology are at first sight persuasive and include: (1) Financial and technical resources could be harnessed to create the 'best' method possible; (2) It would facilitate the mobility of labour and the formation of project teams; (3) It would aid industrial collaboration by providing a common framework for complex software projects; (4) Systems written using a standard methodology could be maintained by a wider range of suppliers; and (5) A commonly used standard would increase the speed of software development because support tools would become available.

To examine the validity of these arguments the experience of using SSADM within the UK will be analyzed.

The Structured System Analysis and Design Method (SSADM) is important because it is mandatory for UK central government software development projects. Its sponsorship by the Central Computer and Telecommunications Agency (CCTA) and the National Computer Centre (NCC), means that it strongly influences the UK IT industry.

Since 1981 the UK software industry has gone into rapid decline and it now has virtually no international standing (Holway, 1992). It was during this period that SSADM was introduced. SSADM is claimed to be the most popular third party development methodology within the UK, holding 25% (Ashworth, 1992) or 41% (Springett, 1993) of the market.

Readers with comments or questions are encouraged to contact the authors via e-mail.

The government is the largest purchaser of software in the country, spending about £1 billion a year (Ashworth, 1992). However, the House of Commons Public Accounts Committee highlighted software as the major culprit in an examination of project cost over-runs. The National Audit Office has also produced reports detailing the serious waste caused by the poor management of government software projects (National Audit Office 1987; 1989; 1990).

This research was undertaken to determine how SSADM has performed in practice by interviewing users of SSADM.

SSADM in the context of other methodologies

According to Jones (1990) a methodology is: "a body of knowledge and techniques.. methodologies should be algorithmic, in the sense that they furnish rules for achieving their desired result." DeMarco & Lister (1987) would agree defining a methodology as: "a proven method for undertaking a repeated task."

The above definition is very broad as the more detailed analysis of methodologies provided by Avgerou & Cornford (1993) makes clear. For example, methodologies can have varying degrees of prescription; they can be more or less contingent on the context in which they are to be used; they can be complete or only provide rules for parts of the task and they can have an engineering or a sociological bias.

There are many views on how to develop information systems. Some of these perspectives have been captured in particular methodologies, for example: ad hoc (Jones, 1990), waterfall (Royce, 1970), participative (Mumford & Weir, 1979), soft systems(Checkland, 1981), prototyping (Naumann & Jenkins, 1982), incremental (Gilb, 1988), spiral (Boehm & et al., 1984), reuse (Matsumoto & Ohno, 1989), formal (Andrews & Ince, 1991), rapid application development (Martin, 1991), object-oriented (Coad & Yourdon, 1991) and software capability (Humphrey, 1990).

SSADM would be classified as a heavily prescriptive methodology, although some streamlining is allowed. It takes a rational, engineering view of the world. It is driven by analysis of the data within a proposed information system and it requires a 'waterfall' life cycle to be followed.

Background

The UK government is not alone in trying to use methodology to raise the effectiveness and efficiency of software developers. Capers Jones (1986) examined the impact of standards and formal development methods in over 100 large enterprises in the United States and Europe. He concluded that people felt a certain comfort from their existence but the evidence on their benefits was ambiguous. Boehm (1981) provides comprehensive data which indicates that methodology is far less important than the ability of the developers and the complexity of the project.

DeMarco (1982) summarizes his experience of methodology in the following terms: 'The idea that a single methodology should govern even two different projects is highly suspect: The differences between projects are much more important than the similarities.' (p. 131)

DeMarco goes on to point out that the need to tailor a methodology is always recognized, but that the senior and lower levels of the hierarchy interpret this differently. Data reported by DeMarco and Lister (1987) indicates that working conditions are critical for raising productivity. They also observed that detailed prescriptive methodologies reduce rather than increase productivity. The reasons they identified for this were: a morass of paperwork; a paucity of methods; an absence of responsibility and a general loss of motivation. (ibid. p.116)

SSADM is commonly perceived to be 'prescriptive, burdensome and difficult to apply' (Thomson, 1990). Other criticisms are that staff do not really understand SSADM and are just 'going through the motions' (Crinnion, 1991) or 'learn it by rote, then use it as an excuse not to

think' (Holloway, 1993). That the top down structured approach of SSADM is too rigid and does not reflect the way people work in practice (Whitefield & Sutcliffe, 1992) (Trull, 1990). That it attempts to substitute methodology for management (Simpson, 1990) and it puts too much emphasis on functionality, analysis and design at the expense of people and organisational issues (Cockcroft, 1990).

On the positive side, Hares (1990, p.39) asserts: 'Failure to produce high quality deliverables is due to poor application of the method, not the method itself'. Young, (1993) comments: 'I strongly believe that any method is better than none and SSADM is certainly worth using if the alternative is ad hoc software development'.

The SSADM Version 3 manuals (Longworth & Nicholls, 1986; 1986a), the handbook, (Longworth, Nicholls & Abbott, 1988) and the SSADM Version 4 manuals (CCTA, 1990) offer no empirical data to underpin the methodology. The official training materials (AIMS Systems, 1990) from the creators of SSADM v.4 also offer no supporting data or references to explain the construction of SSADM. The large number of books which have emerged for the SSADM training market, (for example: Cutts, 1991; Downs, Clare & Coe, 1992; Eva, 1990; Skidmore, Farmer & Mills, 1992) tend to focus on presenting the methodology rather than evaluating or criticising it.

Research methodology

The initial idea for collecting data on how methodologies functioned in practice was to approach the Civil Service at a senior level and carry out a project with their formal cooperation. As a condition of cooperation it was agreed that any papers could be revised by the Civil Service before release. After several months work an initial paper was submitted for review. The returned, revised document was so bland as to be of little value. This approach to collecting raw data had therefore to be abandoned.

A more informal method was therefore adopted for this research study which collected information from 3 different types of source.

Firstly, to establish how SSADM performs in practice 3 multi-million pound SSADM projects were followed over a 3 year period. They were all public sector although outside central government - housing, education and local government. The progress of the projects was followed by meetings with users, developers and sponsors. These were carried out at 3 monthly intervals using semi-structured interviews. The people were approached informally and assured of complete confidentiality.

Secondly, the author was awarded a contract to provide SSADM training to 90 end users, to enable them to understand their SSADM documents. These training courses were in seminar form in groups of 10 people each lasting for 2 days. The course training materials were live SSADM documents produced by the end users' project teams. The comments of the participants were written down as were any observations about how they coped with the SSADM material.

Thirdly, to provide a broader sample, 17 semi-structured, 2 hour interviews with other project managers were carried out. These were from 12 public sector and 5 private sector organisations. The public sector organisations were the Information Systems Units of a range of government departments. They all used SSADM except for their small projects. Of the private sector organisations only one of them used SSADM and then only if requested by a public sector client.

The reason for including non SSADM projects in this sample was to try to establish what difference SSADM was making to the complete software development process. The private sector representatives were the software development organisation of a major multi national company, two systems software companies that export their products world wide and two national bespoke software developers. These last

two companies were accredited with the ISO9000 quality standard.

The questions asked in the 17 interviews included: What was the methodology used for analysis and design?, How much was the methodology tailored?, Life cycle used?, What was the background of the project leader?, What was the role of the user?, Which methodology was used for project management?, How much was this methodology tailored?, How does this project compare to others?, and How is service delivery measured?

All respondents were project managers and came from an IT background. In all cases SSADM was reported to be modified. Respondents were asked to give a response on a 5 point scale if appropriate and then encouraged to talk as they wished about the topics raised.

This approach to collecting empirical data is not perfect. It is open to interviewer bias distorting the answers given by respondents. The people interviewed are not chosen at random; they will tend to self select. There is a limit to how much hard data can be obtained, for example, respondents sometimes did not know the budget for the system or the timetable. The main justification for this more informal approach is because it was seen as the only way to produce reasonably comprehensive data.

Key Findings

Modifying SSADM

SSADM sees the software development process as a cascade flowing from a clear strategic direction and firm requirements as illustrated on page 59 (CCTA, 1990, F-OVE-6).

This investigation found that in the central and local government sites visited there were roughly 100 small PC based projects for one large mainframe based project. For these small projects SSADM was either disregarded or tailored beyond recognition. The approaches used

Summary of the 17 Interviews

Ref.	Public/ Private	Months Duration	Computer Language	Project IT staff	Method	Proj. Mgt.
1	Private	30	COBOL	12	MAP	none
2	Private	25	COBOL	5	Yourdon	Prince
3	Private	22	Client	18	SSADM4	Prince
4	Private	10	C	9	Own	Task
5	Private	Variety of small projects				
6	Public	30	Oracle	10	SSADM3	Prince
7	Public	7	Oracle	7	SSADM3	Prince
8	Public	12	Oracle	5	SSADM4	Prince
9	Public	10	Oracle	6	SSADM3	Prompt
10	Public	24	Quickbuild	5	SSADM3	Prompt
11	Public	4	Network	5	ad hoc	Prince
12	Public	24	Oracle	3	SSADM3	Prince
13	Public	28	Proprietary	10	Own	Prince
14	Public	24	Dataflex	10	SSADM4	Prince
15	Public	27	Dbase	2	ad hoc	none
16	Public	Variety of small projects				
17	Public	12	Package	3	SSADM4	Prince

were either ad hoc, prototyping, or incremental development. This goes far beyond the 'streamlining' recommended for using SSADM.

STRATEGIC PLANNING
|
FEASIBILITY STUDY
|
REQUIREMENTS ANALYSIS
|
REQUIREMENTS SPECIFICATION
|
LOGICAL SYSTEM SPECIFICATION
|
PHYSICAL DESIGN
|
CONSTRUCT AND TEST

Even when the core 'waterfall' life cycle was retained considerable changes to SSADM were being made. There were no reports of 'pure' SSADM being used and all respondents stated that significant modifications to the methodology were made. Samples of quotes were:

"Entity Life History diagrams not done - no time - wouldn't add that much value"

"Used prototyping with no particular process. Couldn't fit SSADM to Oracle easily."

"Because a modified package solution - no Entity Life History diagrams, no Logical Design, no analysis done, no technical options needed."

The need by all respondents to modify SSADM in most cases significantly would indicate that its prescriptive approach is found hard to apply.

Iteration

SSADM recommends that the work for each of SSADM's five modules is completed in sequence. It should be noted that the applications projects are essentially linear, albeit with some opportunities for integrated tasks. (CCTA, 1990, F-OVE-6). The resulting input to the next module must contain all the required information. (CCTA, 1990, F-OVE-11).

In practice this linear approach was not happening, for the following reasons. Firstly, in every case there was a preferred computer language - either because of their existing skills or due to an organisational standard. Secondly, the needs of the lengthy budgeting, and procurement procedures required decisions to be made on hardware early in the process. Often as will be discussed in the next section the 'upstream' work could not be completed.

Strategy

SSADM states that: 'It is assumed that business planning, IS strategy and tactical planning, will have been carried out before an SSADM project is initiated. Whether formally or informally, the types of analysis implied by these tasks; must be undertaken before an SSADM project can be initiated.' (CCTA, 1990, F-OVE-6)

The interviews showed that this was rarely the case.

"When a change in strategy - users don't know how to interpret - can't see what to do."

"Often strategy not defined; yet have to act."

Developers mentioned that strategies were often contradictory and vague at key points which were vital for implementation. The strategies were also liable to confuse the line managers who were not really sure what they meant. They were also non existent on occasions when action was unavoidable. There were several mentions of 'planners blight', when action was halted while the strategy was completed, which was disruptive. Finally, in four cases, there were fundamental changes of strategy in the middle of projects due to changes in the senior management.

Requirements Analysis

SSADM advocates the use of a priori-

tized list of requirements. Requirements can be added to and deleted from this list as the project progresses. Requirements are to be quantifiable and measurable, for example, Step 370 (CCTA, 1990, F-RD-7) requires: '.....ensure that all requirements, particularly non-functional requirements, have been identified, are described correctly, and are fully detailed.'

This need to start with firm requirements was in every case a stumbling block. The interviews found:

"Users sign off documents - this doesn't mean they understand or agree."

"Users often don't understand their own systems."

"Internal politics override administrative and technical sense."

In summary the reasons this part of the methodology caused so many problems are the following: (1) The users did not know what they wanted; (2) The users did not know the possibilities of the technology; (3) Users' perceptions changed as the system was being developed; (4) The developers did not understand the intricacies of the users' work; and (5) There were constant changes in the external environment that were not anticipated.

This need in SSADM for developers to establish firm requirements was for all practical purposes impossible to achieve. As a consequence of this the following inevitably happened with SSADM projects - either the users were obliged to take the system they asked for even though it did not meet their requirements or the projects degenerated into a 'Code and test' cycle to try to create a useful product. This caused particular problems when there was a legal contract with an outside developer, based on the firm set of requirements. The requirements, although firm, were of poor quality and frequent alterations were needed.

Approach to users

In SSADM users are asked their requirements which are then catalogued and prioritized. This process is described in (CCTA, 1990, F-CON-4):

'Users needs are given high priority in SSADM and user involvement is defined and highly visible. They have a major involvement in the expression of their business needs, in the decision-making process at all levels, and throughout all phases of the method.'

This approach tended to lead to communication being ritualized with long formal meetings and documents signed off unread. The lack of partial deliveries of software, the long time scales and the large quantities of documentation sapped morale and were the reasons given for the low commitment to the projects.

The better projects, defined as delivering working systems within reasonable cost and time parameters, did much more than ask or involve the users. To establish the depth of communication necessary the successful projects ensured that developers and users shared the same office space, went on joint site visits and that the developers spent time doing the users' job. They also went to talk to the users' customers to learn their perspectives. Joint informal meetings with the project sponsors helped the team to gain a depth of insight and consensus into what their objectives were.

It would appear that, in practice, the need was to develop trust and a shared vision of what the project was trying to achieve: users are part of the system and therefore it is necessary that their capabilities are explicitly grown with the system. The SSADM recommendation of asking and involving users was observed to be too superficial an approach. The perceived need was to foster commitment, develop the users' skills and to consciously handle the politics of the project. The evidence from these projects is that SSADM tends to limit the contribution of the user to 'involvement' and 'expression' rather

than the needed participation and commitment.

The use of diagrams

SSADM states that: 'SSADM's graphical notations can be readily understood by users, and greatly contribute to effective communication between them and the analysis team.' (CCTA, 1990, F-CON-4)

Diagrams and conventions for their use are central to SSADM, but practical evidence shows that they may not contribute as expected for the following reasons that emerged from the observations and the interviews: (1) End users easily confuse the different diagramming techniques and are rarely clear if they are describing an existing system or designing a new one. Users who are asked to communicate through an unfamiliar method with apparently arcane conventions, often just retreat from the project and virtually all communication is lost; and (2) The lack of effective communication is confirmed by routinely finding several errors per page on diagrams which have been quality assured. This high level of errors for these diagramming techniques has also been reported by Gilb (1988) and Parkinson (1991).

The most successful projects, in terms of practical results delivered, focused on creating a supportive atmosphere in which communication could flourish. There seems to be no reason why users cannot scribble and express themselves as they wish. These ideas can then by refined as necessary. The important point is to ensure that the users enjoy the process and may become curious about how to develop their system skills. Boddie (1987) observes that formal communication is often ineffective, compared to using 'social' mechanisms of leadership and peer group example.

SSADM focuses the developer on the technical drawing techniques, not the soft skills of facilitating the human process of communication. By encouraging this orientation, real communication is lost which is partly why SSADM diagrams tend to be inaccurate in practice.

SSADM with large systems

These are systems with over 3 years development time and budgets over £1 million. Following 3 of these projects for 3 years showed that they all failed to deliver the system to users on time and to meet users needs. The large systems adhered to SSADM very closely. The reasons for this were: (1) Because of the amount of money involved staff at all levels become very cautious about deviating from the recommended way; (2) Using SSADM would provide political protection should the project run into trouble; (3) Large projects are of necessity more formalized and it was easier to agree to use the complete method; and (4) In all 3 cases this was the biggest project the staff had ever attempted and they were willing to trust SSADM even if its recommendations did not seem sensible.

In all 3 projects technical problems emerged very late on in the development process. For example on one project the database response times were hopelessly slow. This was in spite of the extensive use of experienced SSADM consultants. This was only solved by an expensive upgrade of the hardware. In another a serious design flaw was not picked up until just before implementation. Because the problems were discovered late on in the project they were costly and disruptive.

All 3 projects experienced severe problems in their relationships with users. The users for two of the large systems who were 'involved' with their SSADM project were also trying to undermine the project, by actively lobbying to obtain their own separate system. After 2 years the SSADM projects had produced nothing tangible and they had lost confidence in the process. In the third the users had to wait 4 years for any tangible results that they could work with. This had damaged their morale and that of the developers.

Staff turnover and capability

SSADM is intended to compensate for staff turnover and make inexperienced staff more

productive (Longworth, 1989; 1992).

In this sample of projects examined, staff turnover within the developers was not observed to be the reason that the projects were running into problems. A main reason was that the developers had generally only two or three years of project experience and junior staff often less than that. The development of their skills did not seem to have been very methodical. Handling I.T. projects with or without SSADM takes a lot of skill which the inexperienced people did not have. The two most successful projects both had project managers with over 10 years good experience.

Learning SSADM

The introduction to the 4 volumes of the SSADM manual which weigh 10 pounds states:

'This document has been produced as a Reference manual for analysts trained in SSADM. It does not, by itself, constitute an adequate training guide and is not therefore suitable for use by trainees.' (CCTA, 1990, 1-INT-vii)

To understand SSADM v.4 it is necessary to attend an authorized three week training course and work through three files (AIMS Systems, 1990). The training, although expertly carried out, lacked conviction because there was no empirical validation of the methodology. There were no references showing SSADM's development or comparisons with other approaches. There was no data on developer productivity, product quality, timeliness of delivery or user satisfaction. SSADM is such a large and sophisticated methodology that students tended to be overwhelmed with the detail and complexity. As Skidmore and Wroe (1990) observe, SSADM models often become more elaborate than the system they are attempting to explain.

Discussion

SSADM is a data driven, 'waterfall' methodology which takes a rational and technical view of the world. It is prescriptive although streamlining is allowable. But the results of this present study raise questions about key parts of the methodology.

The 'waterfall' approach?

The drawbacks of this 'waterfall' approach have been well documented by Jones (1990), Parnas & Clements (1986), Spence & Carey (1991) and others. The difficulties confirmed by this research were those of managing ever shifting requirements, poor relationships with users and the emergence of serious problems late in a project. This indicates that the waterfall method may not be the best way to develop the majority of public sector IS projects.

A recommended and efficient way?

There are many different models of the software process, for example: ad hoc, waterfall, participative, soft systems, prototyping, incremental, spiral, reuse, formal, rapid application development, object-oriented and software capability. All of these approaches have different cost profiles, strengths and weaknesses. Each model has many variants and to some extent they can be combined to produce hybrid approaches.

Given the wide range in the size, complexity, risk, context and urgency of the IS projects in the public sector, it would seem sensible for the developers and users to agree the optimum approach for their particular needs. It is suggested that, rather than streamlining a prescriptive approach, a specific development model for each project should be created.

Strategic Stability?

The assumption of a stable and coherent strategic context within which an SSADM project would take place was shown to be invalid in practice. While it may be possible to improve the strategies within organizations, it seems unrealistic to assume a clear strategy before starting an IS project. Therefore a software development model which is more comfortable with strategic

ambiguity and uncertainty is required. The models most at home in this environment would be: Rapid Applications Development (Martin, 1991) and Incremental Development (Gilb, 1988; Humphrey, 1990).

Firm requirements?

This research shows that good quality, firm, detailed requirements are very hard to obtain. This has been observed by others, for example Humphrey (1990) states;

'For large-scale programs, the task of stating a complete requirement is not just difficult; it is practically impossible. Unless the job has previously been done and only modest changes are needed, it is best to assume the requirements are wrong' (p.25)

As this seems to the general case then the best software development approach would seem to be the incremental one. This entails defining a small core of major requirements which are to be delivered within a few months, rather than years. The system is then grown rather than built.

Project size?

Examining the range of projects being worked on showed that there were roughly 100 small PC projects to 1 large mainframe project. SSADM which has its roots in MODUS from the mainframes of the early 1970s (Hares, 1990), did not adapt easily to these micro computer based projects. It would therefore seem sensible to revise SSADM to focus on the majority of projects which are small. This would help the junior staff learn good practice early on. Tailoring SSADM takes considerable skill (Ince, 1991) which these staff are unlikely to possess.

There seemed no reason why the larger projects could not each be broken down into several smaller ones and handled with an incremental development process. The data from Putnam & Meyers (1992) shows that larger projects require exponentially more effort than smaller ones. Many of the objectives of SSADM could be achieved by restricting developers and users to small projects until their skills develop. This indicates that SSADM should have small projects as its primary rather than secondary target: an indication that can be utilized at a later stage of this investigation.

One track approach?

SSADM implies a one track approach and does not encourage the creation of alternative scenarios for the project as it develops. In contrast experienced analysts were always planning what action to take if problems arose with staff turnover, hardware performance, procurement delays and so on. The successful analyst is careful to focus his attention on high risk areas, not on the whole project as indicated by SSADM,

Staff development?

Cheap literature and authorized training is claimed to be one of the main advantages of SSADM as an open methodology. The problem is that if the standard approach is flawed, as this evidence indicates, then the books, training and consultancy are obliged to propagate the errors.

Recommendations

This present work confirms the findings of Boehm (1981), Jones (1986) and DeMarco & Lister (1987) that people rather than methodology are the key factors in raising productivity. The recommendations are therefore concerned with ways to develop staff.

Project register: for funds to be released for a software project, large quantities of documentation have to be created and submitted. A few parameters of budget, size, time scales should be held centrally and updated as the project progresses. This would quickly provide a profile of actual performance and a basis for benchmarking.

Process Maturity: assessment of the software development capability of the various parts

of the public sector, would provide a firmer base for training and development plans. If this sample is typical, virtually all of the public sector would rank at the bottom of the Software Engineering Institute's Software Process Maturity model.

User communication: The emphasis needs to be shifted from diagramming techniques and CASE tools to the sociology of projects. The evidence of the poor relationships with users, indicates that much larger and cheaper gains could be made from tackling the 'soft' organisational rather than the 'hard' technical issues within IS development.

Tailored life cycles: for each project the proposed approach: Evolutionary, Participative, Soft Systems or others as appropriate, would be explicitly stated and related to the context and risk factors of the project.

Project managers: The need for a 'cadre' of well trained and experienced project managers was mentioned by several practitioners. The use of a fast track, career development scheme to improve the skills of staff is required.

Conclusion

SSADM has done a service in promoting an ordered approach to software development and spreading the use of valuable techniques. SSADM offers political protection to civil servants should projects go wrong. It also helps to give the appearance of administrative control over the complex process of software development. It is therefore useful to the public sector.

The evidence presented in this study would confirm the observations of others that SSADM is flawed. (Thomson, 1990; Crinnion, 1991; Holloway, 1993; Trull, 1990; Simpson, 1990; Cockcroft, 1990).

SSADM has three main weaknesses.

Firstly, it is based on a waterfall model of software development which is appropriate for only a minority of projects. This finding is confirmed by data from Verner & Cerpa (1997) that shows that only 27% of organisations use the waterfall approach for all projects.

Secondly, as Avgerou & Cornford (1993) point out there are limitations in trying to standardize professional practices for activities which are so little understood. The fact that SSADM has no empirical base and has not been systematically monitored in use by the CCTA would indicate that it is unlikely to be an effective way to direct the efforts of software developers.

Thirdly, the high level of prescription which accounts for much of the size and complexity of the methodology is not useful for practitioners. This is because the prescriptions are based on the assumption that the waterfall life cycle is appropriate for many projects when clearly it is not. Also the techniques prescribed in such detail, without any empirical justification, are not found to produce benefits and are therefore ignored.

The idea of a 'best' method is misleading because of the diverse range of projects and developers. The generic lesson from this research is that an organisation is probably unwise to use a heavily prescriptive methodology to improve its software development performance. If an organisation feels that methodology is the appropriate route it should ensure there is some empirical data to underpin the proposed approach and that there is some objective monitoring of the methodology when in use.

Implications For Future Research

Software developers work in groups on intellectually complex tasks. Current attempts at knowledge collection and sharing are still crude using such mechanisms as metrics or methodology. The ideal future research would be with an organisation which would explicitly experiment with different approaches to managing software projects. A diversity of approaches would assist organisational learning and would probably reduce the proportion of project failures for the

entire organisation.🕮

References

1. AIMS Systems. *SSADM Version 4 Training Manuals (Vols. 1, 2 & 3)* . Aims Systems, England. 1990.
2. Andrews, D., & Ince, D. *Practical Formal Methods with VDM*. McGraw Hill, London, 1991.
3. Ashworth, A. *The Current Position of SSADM*. Paper presented at the NCC Software Seminar, ICL Belfast, 1992
4. Avgerou, C., & Cornford, T. *Developing Information Systems: Methodologies, Techniques and Tools*. Macmillan, London, 1993.
5. Boddie, J. *Crunch Mode. Building Effective Systems on a Tight Schedule*. Yourdon Press, Prentice-Hall, Englewood Cliffs, NJ, 1987.
6. Boehm, B. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ, 1981.
7. Boehm, B. W., & et al. "A Software Development Environment for Improving Productivity." *Computer*, 17, 6, 30-42, 1984.
8. CCTA. *SSADM Version 4 Reference manuals (Vols. 1, 2, 3 & 4)* . NCC Blackwell, Oxford. 1990.
9. Checkland, P. *Soft Systems Methodology*. Wiley, Chichester, 1981.
10. Coad, P., & Yourdon, E. *Object-Oriented Analysis*. Prentice-Hall, Englewood Cliffs, NJ, 1991.
11. Cockcroft, M. *Structured Methods - not seeing the wood for the trees?* Paper presented at the Southcourt conference Making Structured Methods Work, London, 1990, April.
12. Crinnion, J. *Evolutionary Systems Development*. Pitman, London, 1991.
13. Cutts, G. *SSADM*. (2 ed.). Blackwell Scientific Publications, Oxford, 1991.
14. DeMarco, T. *Controlling Software Projects : Management Measurement and Estimation*. Prentice-Hall, Englewood Cliffs, 1982.
15. DeMarco, T., & Lister, T. *Peopleware: Productive Projects and Teams*. Prentice-Hall, Englewood Cliffs, Dorset House, New York, 1987.
16. Downs, E., Clare, P., & Coe, I. *SSADM : Application and Context*. (2 ed.). Prentice Hall, Hemel, Hempstead, 1992.
17. Eva, M. *SSADM: A Practical Approach*. McGraw Hill, Maidenhead, 1990.
18. Gilb, T. *Principles of Software Engineering Management*. Addison-Wesley, Wokingham, United Kingdom, 1988.
19. Hares, J. S. *SSADM for the Advanced Practitioner*. John Wiley & Sons, Chichester, 1990.
20. Holloway, S. "The Method and the Madness." *Government Computing*, 7, 2, 10, 1993.
21. Holway, R. *A Review of the Financial Performance of United Kingdom Computing Services Companies, The Holway Report* . Richard Holway Ltd., Farnham, Surrey. 1992.
22. Humphrey, W. S. *Managing the Software Process*. Addison-Wesley, Reading, MA, 1990.
23. Ince, D. "The Making of a Modern Methodology." *Informatics*, 1991.
24. Jones, G. W. *Software Engineering*. John Wiley & Sons, New York, 1990.
25. Jones, T. C. *Programming Productivity*. McGraw-Hill, New York, 1986.
26. Longworth, G. *Getting the System you want; a User's Guide to SSADM*. NCC Publications, Manchester, 1989.
27. Longworth, G. *Introducing SSADM (Version 4)*. NCC Blackwell, Manchester, 1992.
28. Longworth, G., & Nicholls, D. *SSADM Manual (Version 3)*. NCC Publications, Manchester, 1986.
29. Longworth, G., & Nicholls, D. *SSADM Manual (Version 3), Vol. 2 Techniques and Documentation*. NCC Publications, Manchester, 1986a.
30. Longworth, G., Nicholls, D., & Abbott, J. *SSADM Developer's Handbook*. NCC Publications, Manchester, 1988.
31. Martin, J. *Rapid Application Development*.

Macmillan, New York, NY, 1991.
32. Matsumoto, Y., & Ohno, Y. *Japanese Perspectives in Software Engineering*. Addison-Wesley, Singapore, 1989.
33. Mumford, E., & Weir, M. *Computer Systems in Work Design - The ETHICS Method*. Associated Business Press, London, 1979.
34. National Audit Office. *Inland Revenue : Control of Major Development in the use of Information Technology*. HMSO, London. 1987.
35. National Audit Office. *Department of Social Security: Operational Strategy* . HMSO, Session 1988-89, HC111, London. 1989.
36. National Audit Office. *Managing Computer Projects in the National Health Service* . HMSO, London. 1990.
37. Naumann, J. D., & Jenkins, A. M. "Prototyping: The New Paradigm for Systems Development." *MIS Quarterly,* September 1982.
38. Parkinson, J. *Making CASE Work*. Blackwell, Oxford, United Kingdom, 1991.
39. Parnas, D. L., & Clements, P. C. "A Rational Design Process : How and Why to Fake It." *IEEE Transactions on Software Engineering,* 12, 2, 251-257, 1986.
40. Putnam, L., H, & Meyers, W. *Measures for Excellence : Reliable Software on Time and on Budget*. Yourdon Press, Englewood Cliffs, NY, 1992.
41. Royce, W. W. *Managing the Development of Large Software Systems: Concepts and Techniques*. Paper presented at the WESCON, 1970
42. Simpson, I. *quoted in Black, G. ibid.* 1990.
43. Skidmore, S., Farmer, R., & Mills, G. *SSADM Version 4 Models and Methods*. NCC, Manchester, 1992.
44. Skidmore, S., & Wroe, B. *Introducing Systems Design*. NCC Blackwell, Manchester, 1990.
45. Spence, I. T. A., & Carey, B. N. "Customers do not want Frozen Specifications." *Software Engineering Journal,* 6, 6, 175-180, 1991.
46. Springett, P. "The Method and the Madness." *Government Computing,* 7, 6, 10, 1993.
47. Thomson, I. "SSADM: The Last Word." *Government Computing,* 4, 5, 28-29, 1990.
48. Trull, H. quoted in Black, G. Promises to be Fulfilled. *Financial Times,* pp. 7, 1990, October 19.
49. Verner, J. M., & Cerpa, N. "Prototyping: Does Your View of its Advantages Depend on Your Job." *Journal of Systems Software,* 36, 1, 3-16, 1997.
50. Whitefield, A., & Sutcliffe, A. "Case Study in Human Factors Evaluation." *Information and Software Technology,* 34, 7, 443-453, 1992.
51. Young, G. "quoted in Springett, P. ibid." 1993.