A DESCRIPTIVE FRAMEWORK FOR INVESTIGATING PROBLEMS IN THE APPLICATION OF SYSTEMS DEVELOPMENT METHODOLOGIES

Brian Fitzgerald
Executive Systems Research Centre
University College Cork,
Cork, Ireland

Key Words: Information systems, systems development, software development methods, methodologies.

Abstract

It is generally taken as axiomatic that systems development methodologies (SDMs) play a useful role in guiding the development process, and that their increased adoption would improve the product and process of systems development. This paper begins by briefly reviewing the arguments and pressures in favour of SDMs. Following this, a descriptive model of the system development process is formulated, and this is then used as a framework to map a number of fundamental problems in relation to the use of SDMs and their contribution to systems development.

Introduction

The importance of successful systems development persists as an issue of central significance and concern in the IS field, especially in view of the increasingly-complex applications that need to be developed in today's environment, and the well-documented problems associated with system development which have given rise to what has been termed the "software crisis" (cf. Brooks, 1987; Martin, 1984; Naur *et al*., 1976). Much research seems to view the solution to the software crisis in terms of increased adoption of development methodologies, and, indeed, there are several significant arguments and pressures which support the use of methodologies (cf. Fitzgerald, 1995). These arguments and pressures are briefly summarised in Table 1.

A Framework for the Systems Development Process

It is difficult to criticise methodologies, at least in the abstract, as the arguments and pressures discussed above are indeed significant. However, notwithstanding the 'software crisis' and the prescriptions of the literature, practitioners remain reluctant to adopt formalised methodologies (Aaen, 1986; Avison & Fitzgerald, 1988; Ward, 1992), yet systems continue to be developed, some of which at least perform successfully. Indeed, the assumption that methodologies contribute significantly to the process or product of systems development in practice is by no means proven by research to date (cf. Cerveny & Joseph, 1988; Fitzgerald, 1994b; Wastell & Newman, 1993; Wynekoop & Russo, 1993).

Despite the abundance of conceptual and empirical research on methodologies (cf. Fitzgerald, 1994a), there is no universally accepted framework for studying or evaluating methodologies (cf. Sol, 1983). This is all the more problematic given the fact that a huge number of methodologies exist, some of which differ in fairly trivial aspects—the many variations of the structured approach, for example (cf. Ward, 1991, 1992), while others differ fundamentally, both in paradigm and in coverage. Thus, methodologies range from hard, rationalistic with a technical focus—SSADM, for example (Downs *et al*., 1992) to soft, human-oriented ones with a social focus—ISAC, for example (Lundeberg, 1982). There are also those such as ETHICS

(Mumford, 1983**)** which attempt to bridge the social-technical gap Also, methodologies differ markedly in coverage; some do not address preliminary analysis phases, while others ignore later implementation activities (cf. Avison & Fitzgerald, 1988; Sakthivel, 1992).

Table 1: Summary of Issues Supporting Formalised System Development Methodologies

Conceptual basis: Development process more amenable to project management and control, thus minimising risk and uncertainty Economic rationale: division of labour affords skill specialisation and elimination of irrational activities Epistemological rationale: provide a structural framework for the acquisition of knowledge Standardisation of development process facilitates communication and interchangeability of developers
Conceptual basis: Government SDM standards: SSADM (UK, Ireland, Malta, Hong Kong, Israel) Dafne (Italy) Merise (France) NIAM (Netherlands) Department of Defense Std. 2167 (US) Software Capability Evaluation (SCE) programme from the Software Engineering Institute Desirability of ISO-certification Literature bias which views methodologies as step towards solution of software crisis

One comprehensive framework for methodology research and evaluation is the NIMSAD (Normative Information Model-based Systems Analysis and Design) one, proposed by Jayaratna (1986, 1994). The framework has its underpinnings in 'systems' philosophy and theory, and has been validated and refined through industrial 'action research' experience and consultancy. The NIMSAD framework explicitly acknowledges the importance of three factors, namely, the methodology context (the problem situation), the skills and experience of the developer/methodology user (the intended problem-solver), and the methodology-in-action (the problem-solving process). The author’s direct commercial experience as a systems developer, coupled with the results of some empirical research (cf. Fitzgerald, 1994b) indicated the vital importance of these factors. Thus, the NIMSAD concepts were modified to form the framework for this paper which represents a descriptive formulation of the development process (see Fig. 1). The framework components are briefly presented next.

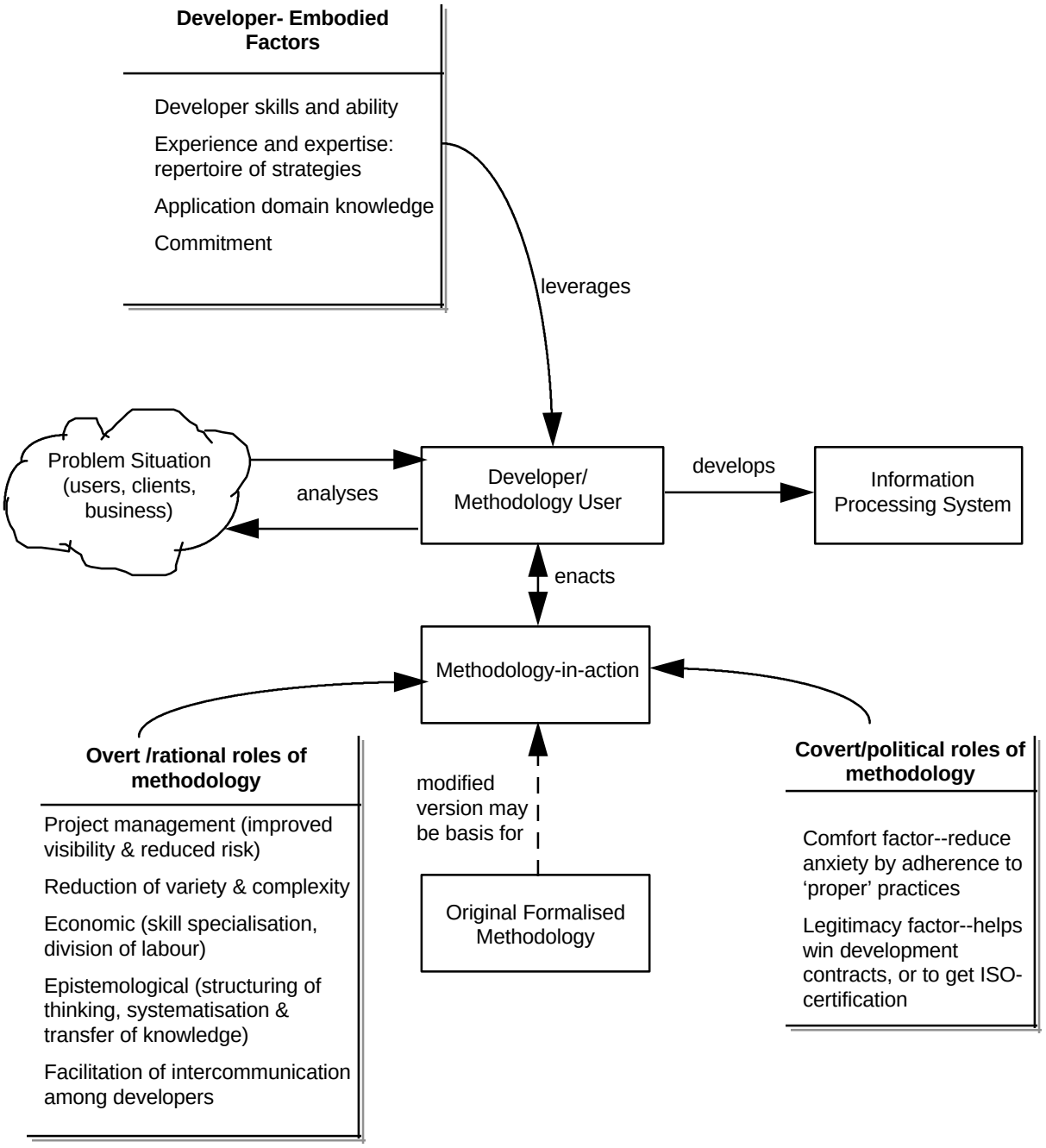

Fig. 1 Conceptual Framework for Systems Development Process

Explanation of the Framework

Firstly, the framework acknowledges the complexity and dynamic nature of the problem situation, that is, the methodology context. The multiplicity of stakeholders, system users, developers, problem owners, is recognised. Here, the stakeholder term is used in the sense of Mason and Mitroff (1981, p.43) who define stakeholders as "all those claimants inside and outside the organisation who have a vested interest in the problem and its solution". Also, the developer/methodology user is accorded a central role in the framework, thus reflecting the fact that it is people, not methodologies who develop systems, and that the latter are merely frameworks which must be leavened by

the wisdom of human developers if they are to be effective. In relation to this, the importance in the development process of developer-embodied factors is explicitly acknowledged. There are several strands of research in the literature which confirm the importance of developer factors. For example, huge variances in the capabilities of different developers and a consequent impact on development productivity have been reported (Boehm, 1981; Brooks, 1987). Also, researchers have identified the importance of learning over time as developers increase their level of expertise (Vitalari & Dickson, 1983). The significant contribution that developer knowledge of the application problem domain has also been acknowledged (Davis & Olson, 1985). A final developer-embodied factor which does not appear to have been considered in the literature is that of developer motivation or commitment. This factor was identified during the author's field research, where a project manager with a vast amount of development experience suggested that the commitment of individual developer's to the task was the most significant factor in ensuring that systems were successfully completed.

Additionally, the framework makes the useful distinction between the original methodology as interpreted by its creator and the methodology-in-action as interpreted by the developer. This distinction has parallels with the distinction drawn by Argyris and Schon (1974) between an "espoused theory" and a "theory-in-use". Thus, methodologies are never applied exactly as originally intended. Different developers will not interpret and apply the same methodology in the same way; nor will the same developer apply the same methodology in the same way in different development situations. Thus, on any development project, the methodology-in-action is uniquely *enacted* by the developer, much as a musician interprets a musical composition in real-time.

Finally, the framework explicitly identifies two broad, but diametrically opposed, categories of roles that methodologies can play in the development process. The overt rational ones were discussed earlier as part of the conceptual basis and rationale behind the use of methodologies, viz., the facilitation of project management and control etc., but a set of covert, political roles are also proposed. These latter have been discovered through empirical research (Fitzgerald, 1994b). They include factors such as the role of methodology as ‘comfort factor’, suggesting that ‘proper’ practices are being followed, and also the ‘legitimacy factor’ scenario whereby organisations claim to use a methodology to win contracts with government agencies, or to help achieve ISO-certification.

Problems in the Application of Methodologies

In this section, a number of fundamental problems in relation to the actual use and contribution of methodologies in practice are discussed and mapped by means of the framework of Fig. 1. These include the poverty of the rational paradigm on which many methodologies are implicitly based, the goal displacement phenomenon whereby following the methodology overshadows actual development, the assumption that methodologies are universally applicable in all development situations, and the inadequate recognition of developer-embodied factors.

Poverty of Rational Technical Paradigm

Many methodologies are based on the rational reductionist paradigm of technical and engineering disciplines (Dumdum & Klein, 1986; Goldkuhl & Lyytinen, 1984). One of

the central tenets of this paradigm is that developers can obtain detailed knowledge about the problem situation; that the *true* requirements can be specified (McMenamin and Palmer, 1984). However, this has been questioned by Jones and Walsham (1988) who argue that there are limits, both to what *can* and what *should* be known. Also, methodologies such as Soft Systems Methodology (SSM) (Checkland, 1981) recognise that the problem situation will be interpreted differently according to the world-view (*Weltanschauung*) of those involved, a similar argument to that proposed by Boland (1979).

Figure 2 illustrates the phenomenon of methodologies following an overly-rational and technical paradigm. The regular oval pattern of the system requirements in contrast to the cloud of the problem situation in Fig. 1 reflects the view that specifying the true system requirements is not seen as problematic. Also, the methodology is accorded central prominence in this scenario, as are the overt rational roles of the methodology. In contrast, the role of the developer is relegated to a position of lesser importance, and the particular developer-embodied capabilities are not acknowledged at all. It is generally assumed that the original formalised methodology can be applied in pure unmodified form in the attainment of rational development goals. The development process is viewed as a rational activity which can be idealised such that following a prescribed methodology can guarantee successful development.

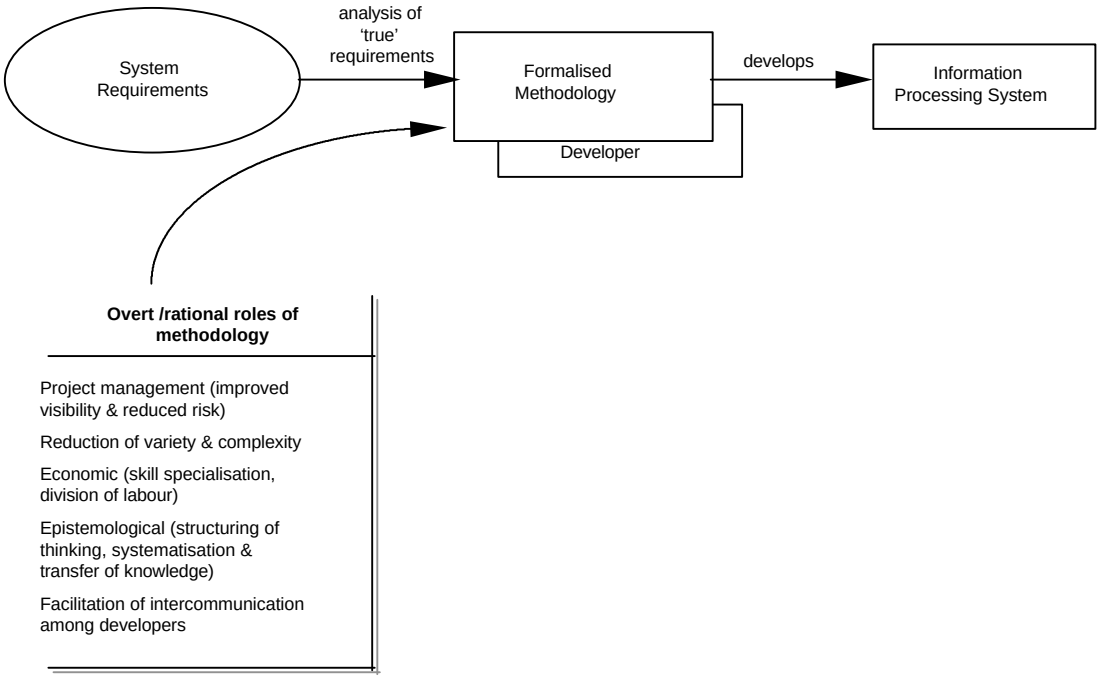

Fig. 2 Poverty of the Rational Technical Paradigm

Goal Displacement

The goal displacement phenomenon is one whereby developers become so engrossed in following the methodology that they lose sight of the real goal which is systems development. Thus developers become specialists in following the methodology rather than conducting development. The dangers inherent in following a formalised development approach were long ago identified by Smith (quoted in Naur *et al*., 1976, p.88) who described the situation where developers produced reams of documentation

but little actual development. He characterised the situation aptly as “confusing the menu with the meal.” This phenomenon was also very evident in the author’s empirical research, where one developer was critical of the time spent constructing entity life histories which did not seem to offer any real added value to the development process, but merely "served to clarify the obvious".

There are a number of reasons underpinning the goal displacement phenomenon. Perhaps, the main one is the fact that development is a complex stressful process (Wastell & Newman, 1993), and any rational prescriptive mechanism—as a methodology generally is—which purports to provide some comfort and reassurance that this complexity can be addressed, will be seized upon by management and developers alike. There is little concern as to whether the methodology acts more as a placebo than a panacea, and a subtle conspiracy takes place with developers and management taking comfort from the fact that a rational methodological approach is being followed.

Figure 3 illustrates the goal displacement phenomenon. Again, the methodology is accorded central prominence in this scenario. The developer role is relegated to the background, and the unique enactment of the methodology-in-action by the developer is not recognised. The complexity of the problem situation is however made explicit. As a consequence, the covert, political role of the methodology become paramount, with the methodology acting as a “comfort factor” in the face of the complexity of system development, assuring all the participants that the ‘proper processes’ are being followed. Ironically, the development of the actual system becomes almost an afterthought (indicated by the dashed line) as developers concentrate on blind and slavish adherence to the methodology instead.

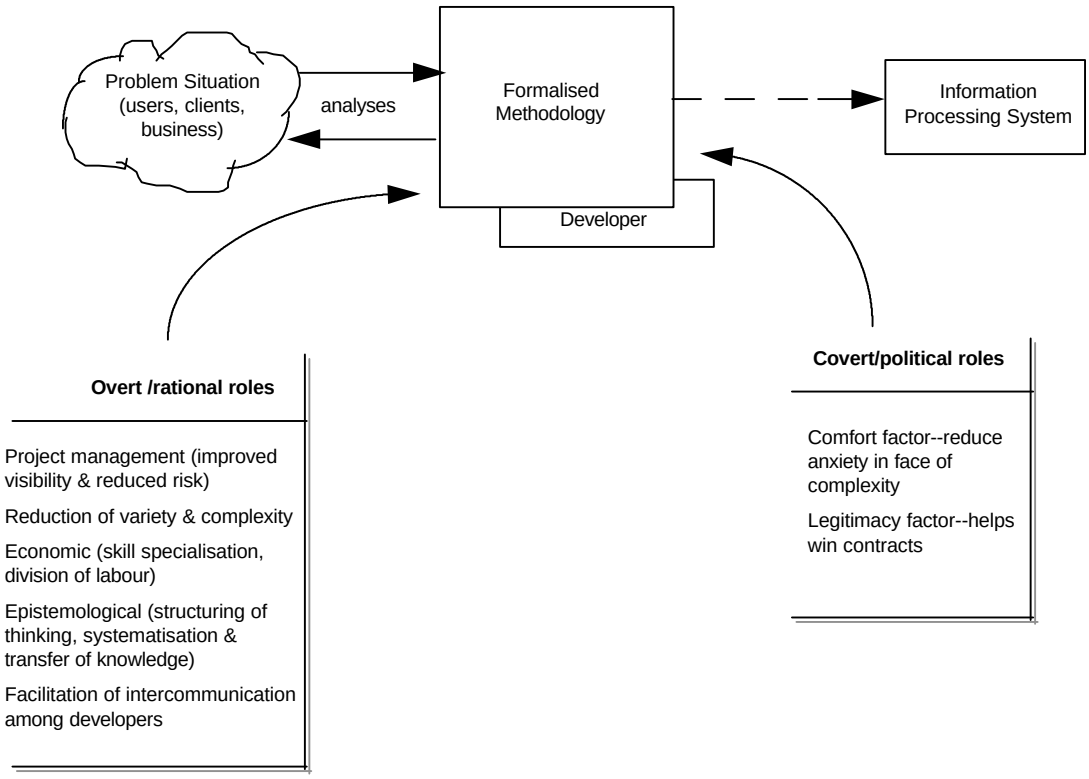

Fig. 3 Goal Displacement

Assumption that Methodologies are Universally Applicable

There is a tendency in Western culture to value rational processes, and view them as universally applicable (Kindler & Kiss, 1984). Suchman (1987) illustrates this by comparing Western navigational procedures with the heuristic situated methods used by primitive tribes since ancient times, the latter being equally effective. Suchman argues for the need for situated action, that is, action tailored for the specific contingencies of the situational context. In the methodology literature, this is reflected in the recent research focus on contingency approaches (Avison *et al*., 1988; Curtis *et al*., 1988; Iivari & Koskela, 1987). However, methodologies, often generalised from very limited practical application (cf. Ward, 1991, 1992) attempt to prescribe idealised approaches without due regard to the contingencies and uniqueness of each development situation.

Figure 4 illustrates this phenomenon. Again, the problem situation tends to be viewed in a simplistic rational manner. The methodology is accorded a central role, the overt rational role of the methodology is recognised as paramount, and the contribution of the developer is played down. This is particularly problematic as it fails to recognise the importance of learning over time as developers increase their expertise. Another problem arises in that strict adherence to the methodology may impose a considerable inertia on the development process as the individual steps are scrupulously followed. This is indicated in Fig. 4 by the lengthened development arrow which depicts the cumbersome nature of the development process. Any methodology is at best merely an organising framework and it is vital that they be tailored to the situation and that developers do not follow them blindly or dogmatically, plodding through idealised checklists.

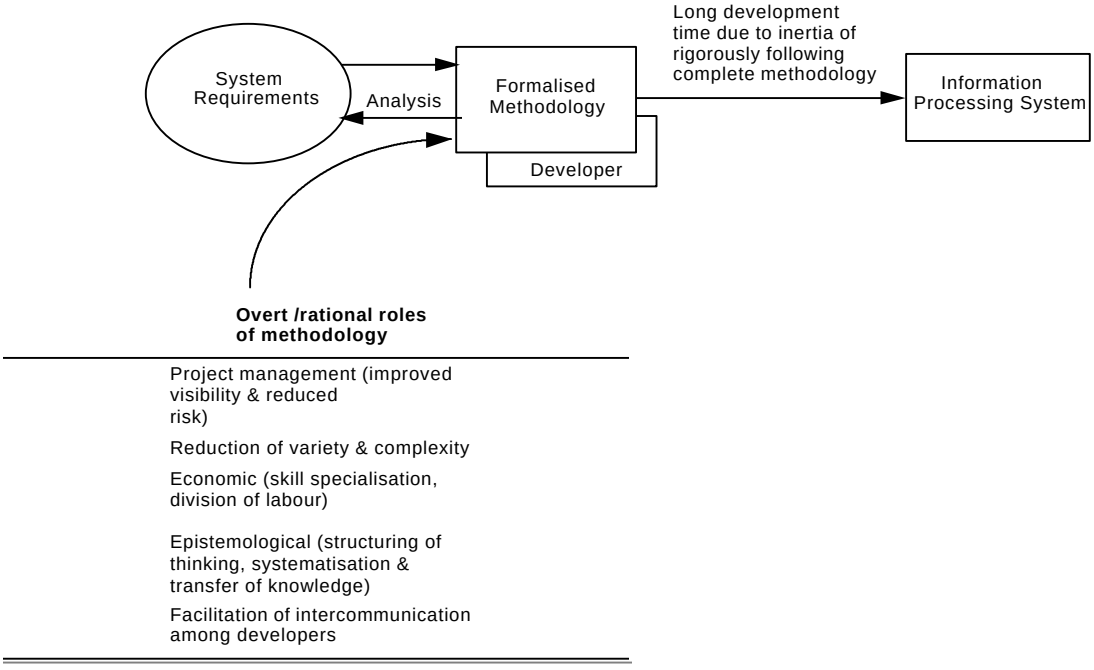

Fig. 4 Assumption that Methodologies are Universally Applicable

Inadequate Recognition of Developer-Embodied Factors

The rational scientific paradigm inherent in many system development methodologies now appears to be somewhat bankrupt, and the recognition that development is an artistic activity taking place in a complex social context is increasingly recognised (cf. Baskerville *et al*., 1992; Bubenko, 1986; Land *et al*., 1980). The importance of people factors has been confirmed by several researchers (cf. Boehm, 1981; Brooks, 1987; Glass, 1991). Quite simply, developers over time acquire a "repertoire of strategies" to apply in different development situations (Vitalari & Dickson, 1983). Researchers have called for increased recognition of this learning process (Floyd, 1987) whereby methodologies, rather than being applied in a standardised stereotyped manner, would be viewed as second-order learning processes affording the opportunity for developers to feedback learning from past projects.

Figure 5 illustrates this phenomenon. Again, the methodology assumes a central role as if it is sufficient of itself. Thus, developer-embodied factors such as ability, past experience, knowledge of the particular problem domain are discounted. The overt, rational roles of the methodology are explicitly recognised Also, the methodology is viewed as applicable in unmodified form, as the unique enactment of the methodology-in-action by the methodology user is not acknowledged. For example, one of the stated benefits of the Jackson System Development (JSD) methodology is that it reduces reliance on developer inspiration and creativity, and should be applied in the same way by all developers in all situations (King & Pardoe, 1985). This is certainly not in accord with the widely-held view that developer ability and experience are the most important factors in ensuring successful system development.

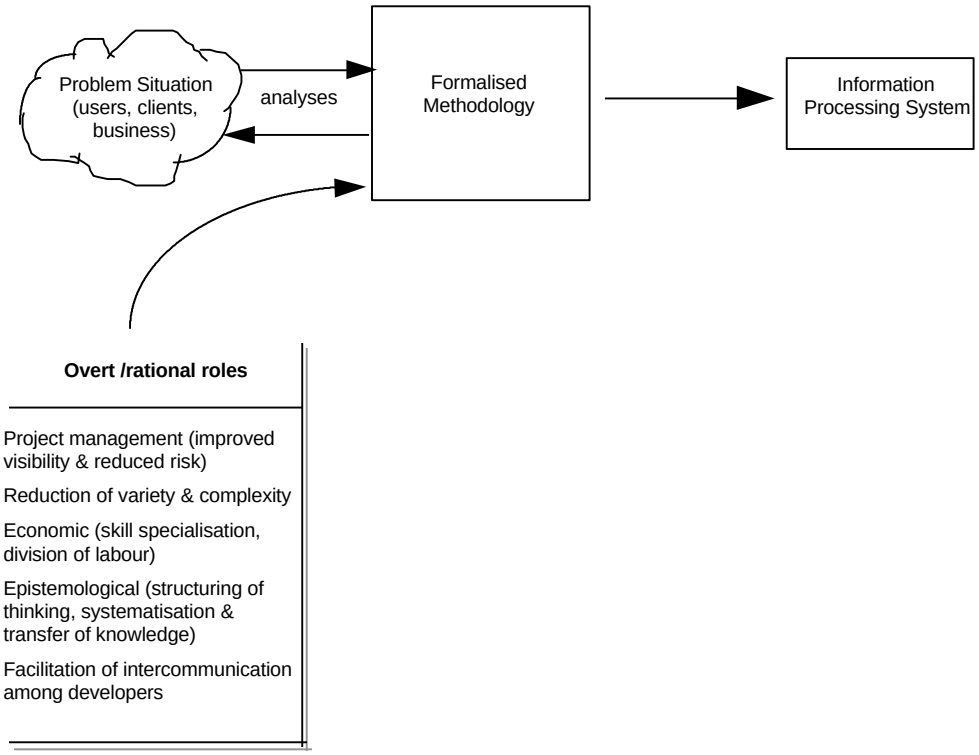

Fig. 5 Inadequate recognition of Developer-Embodied Factors

Conclusion

Many researchers and practitioners continue to see the solution to the software crisis in terms of increased control and the more widespread adoption of system development methodologies. The arguments and pressures which support the use of such methodologies, as presented in Table 1 above, are indeed significant, but the problems associated with the use of methodologies have not perhaps received adequate attention in the literature (cf. Fitzgerald, 1995). Thus, the assumption that increased adoption of methodologies would help address the problems inherent in systems development is by no means proven (Wynekoop & Russo, 1993). In fact, while methodologies may contribute little to either the process or product of systems development, they continue to be used in organisations, principally as a 'comfort factor' to reassure all participants that 'proper' practices are being followed in the face of the stressful complexity associated with system development. Alternately, they are being used to legitimate the development process, perhaps to win development contracts with government agencies, or to help in the quest for ISO-certification. In this role, methodologies are more a placebo than a panacea, as developers may fall victim to goal displacement, that is, blindly and slavishly following the methodology at the expense of actual systems development. In this mode, the vital insight, sensitivity and flexibility of the developer are replaced by automatic, programmed behaviour.

References

Aaen, I. (1986), Systems development and theory—in search of identification. In Nissen, H. and Sandstrom, G. (Eds), Quality of Work versus Quality of Information Systems, Lund University, Sweden, pp.202-223.

Argyris, C. and Schon, D. (1974), *Theory in Practice: Increasing Professional Effectiveness*, Jossey-Bass, USA.

Avison, D. and Fitzgerald, G. (1988), *Information Systems Development: Methodologies, Techniques and Tools*. Blackwell Scientific Publications, Oxford.

Avison, D., Fitzgerald, G. and Wood-Harper, A. (1988), Information systems development: a tool kit is not enough. *The Computer Journal*, **31**, 4, pp.379-380.

Baskerville, R., Travis, J. and Truex, D. (1992), Systems without method: the impact of new technologies on information systems development projects. In Kendall, K. *et al*. (eds.), pp.241-269.

Boehm, B. (1981), Software Engineering Economics. Prentice Hall, Englewood Cliffs, New Jersey.

Boehm, B. (1988), In Gilb, T. Principles of Software Engineering Management, Addison Wesley, UK.

Boland, R. (1979), Control, causality and information systems requirements. Accounting, Organizations and Society, **4**, pp.259-275.

Brooks, F. (1987), No silver bullet: essence and accidents of software engineering. IEEE Computer Magazine, April, pp.10-19.

Bubenko, J. (1986), Information system methodologies -- a research view. In Olle et al., (1986), Information Systems Design Methodologies: Improving the Practice, North-Holland, pp.289-318.

Cerveny, R. and Joseph, D. (1988), A study of the effects of three commonly used software engineering strategies on software enhancement productivity, *Information & Management*, **4**, pp.243-251.

Checkland, P. (1981), *Systems Thinking, Systems Practice*, Wiley, Chichester.

Curtis, B., Krasner, H. and Iscoe, N. (1988), A field study of the software design process for large systems. Communications of the ACM, November, pp.1268-1287.

Davis, G. and Olson, M. (1985), Management Information Systems: Conceptual Foundations, Structure and Development, McGraw-Hill, New York.

Downs, E., Clare, P. and Coe, I. (1992), Structured Systems Analsis and Design Method: Application and Context. Prentice-Hall International(UK), Hertfordshire.

Dumdum, U., and Klein, H. (1986), The need for organizational requirements analysis. In Nissen, H. and Sandstrom, G. (Eds), Quality of Work versus Quality of Information Systems, Lund University, Sweden, pp.393-420.

Fitzgerald, B. (1994a), The systems development dilemma: whether to adopt formalised systems development methodologies or not?, In Baets, W. (Ed), Proceedings of Second European Conference on Information Systems, Nijenrode University Press, Breukelen, pp.691-706.

Fitzgerald, B. (1994b), Whither systems development: time to move the lamppost?, in Lissoni *et al*., (eds), *Proceedings of Second Conference on Informatioon Systems Methodologies*, pp.371-380.

Fitzgerald, B. (1995), Formalised systems development methodologies: a critical perspective, *Information Systems Journal*, (forthcoming).

Floyd, C. (1987), Outline of a paradigm change in software engineering. In Computers and Democracy: A Scandinavian Challenge. Bjerknes, G., Ehn, P., and King, M. (Eds), Avebury Gower, Brookfield Vermont.

Glass, R. (1991), Software Conflict: Essays on the Art and Science of Software Engineering. Yourdon Press, Prentice Hall, Englewood Cliffs, New Jersey.

Goldkuhl, G. and Lyytinen, K. (1984), Information systems specification as rule construction. In Kendall, K., DeGross, J. and Lyytinen, K. (eds.), (1992), The Impact of Computer Supported Technologies on Information Systems Development, Elsevier Science Publishers B.V., North Holland Press, pp.79-94.

Iivari, J. and Koskela, E. (1987), The PIOCO model for information systems design, *MIS Quarterly*, **September**, pp.400-419.

Jayaratna, N. (1986), NIMSAD: a framework for understanding and evaluating methodologies, *Journal of Applied Systems Analysis*, **13**, pp.73-78.
Jayaratna, N. (1994), *Understanding and Evaluating Methodologies*, McGraw-Hill, London.
Jones, M. and Walsham, G. (1992), The limits of the knowable: organizational and design knowledge in system development. In Kendall, K. et al., (eds), pp.195-213.
Kendall, K, Lyytinen, K. and DeGross, H. (eds) (1992) *The Impact of Computer Supported Technologies on IS Development*, Elsevier Publishers, North Holland.
Kindler, J. and Kiss, I. (1984), Future methodology based on past assumptions. In Tomlinson, R. and Kiss, I. (eds.), *Rethinking the Process of Operational Research and Systems Analysis*, Pergamon Press, pp.1-17.
King, M. and Pardoe, J. (1985), Program Design using JSP: A Practical Introduction, Macmillan, London.
Land, F., Mumford, E. and Hawgood, J. (1980), Training the systems analyst of the 1980s: four analytical procedures to assist the design process. In Lucas, H., Land, F., Lincoln, and Supper (eds.), *The Information Systems Environment*, North Holland Press, pp.239-256.
Lundeberg, M. (1982), The ISAC approach to specification in information systems etc. In Olle *et al*. (Eds), (1982), *Information Systems Design Methodologies: A Comparative Review*, North-Holland, pp.173-234.
Martin, J. (1984), *An Information Systems Manifesto*, Prentice-Hall, Englewood Cliffs.
Mason, R. and Mitroff, I. (1981), *Challenging Strategic Planning Assumptions*, Wiley & Sons, New York.
McMenamin, S. and Palmer, J. (1984), *Essential Systems Analysis*, Yourdon Press, Prentice Hall, Englewood Cliffs, New Jersey.
Mumford, E. (1983), *Designing Human Systems*, Manchester Business School, Manchester.
Naur, P. Randell, B. and Buxton, J. (1976), Software Engineering: Concepts and Techniques, Charter Publishers, New York.
Sakthivel, S. (1992), Methodological requirements for information systems development. Journal of Information Technology, **7**, pp.141-148.
Sol, H. (1983), A feature analysis of information systems design methodologies, in Olle, T., Sol, H. and Tully, C. (1983), *Information Systems Design Methodologies: A Feature Analysis*, North-Holland.
Suchman, L. (1987), *Plans and Situated Actions*, Cambridge University Press, Cambridge.
Vitalari, N. and Dickson, G. (1983), Problem solving for effective systems analysis: an experimental exploration. Communications of the ACM, November, 948-956.
Ward, P. (1991), The evolution of structured analysis: Part I--the early years. American Programmer, **4**, 11, pp.4-16.
Ward, P. (1992), The evolution of structured analysis: Part II--maturity and its problems. American Programmer, **5**, 4, pp.18-29.
Wastell, D. and Newman, M. (1993), The behavioral dynamics of information systems development: a stress perspective, Accounting, Management & Information Technology, **3**, 2, 121-148.
Wynekoop, J. and Russo, N. (1993), System development methodologies: unanswered questions and the research practice gap. In De Gross, J. et al., (eds), Proceedings of the 14th International Conference on Information Systems, ACM, New York, pp.181-190.
Zuboff, S. (1988), *In the Age of the Smart Machine*, Heineman, London.

Cutting area

Certainly, the profile of development that practitioners face in organisations today is very different from that faced by developers when these methodologies were first mooted some 10 to 20 years ago. Thus, the viability of formalised methodologies is open to question.

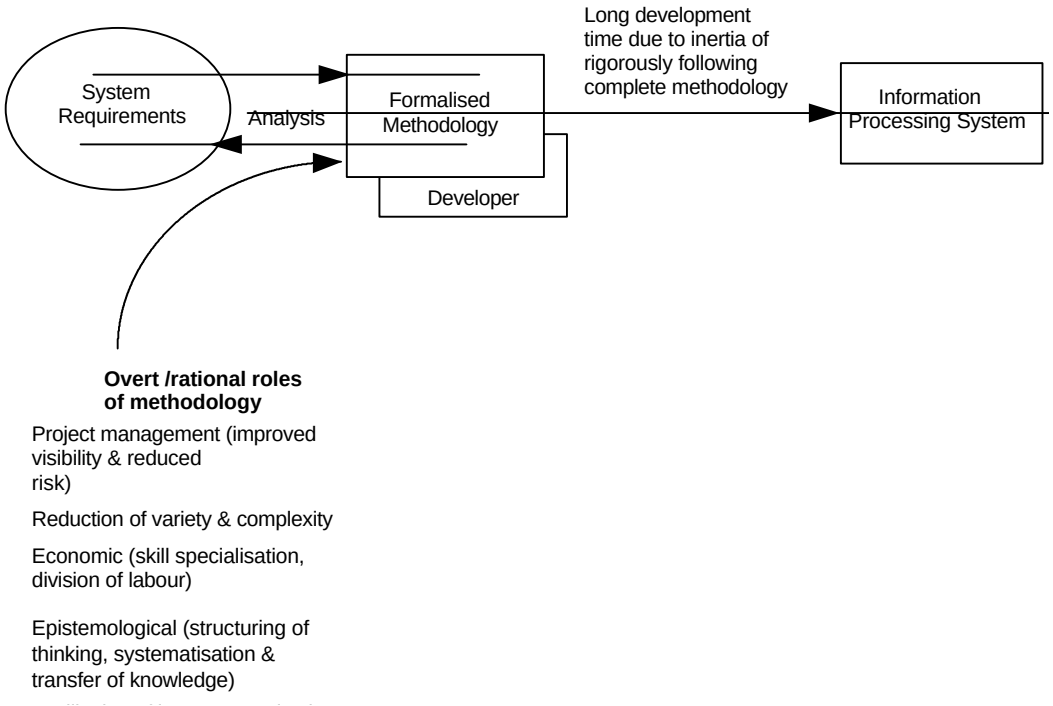

Fig. 4 Assumption that Methodologies are Universally Applicable

Review bit below...

Systems development issues occupy a position of central importance in the information systems field and, indeed, much has been prescribed in the quest for successful systems development. However, given the well-documented "software crisis", success is far from guaranteed for systems development projects. Many researchers see the solution to the software crisis in terms of increased control and the more widespread adoption of rigorous and formalised system development methodologies. This paper first discusses the arguments and pressures which support the use of methodologies. Some evidence of the literature bias which favours methodologies is also presented.

- Methodologies facilitate project management and control by allowing some visibility into the complexity of the systems development process. They may provide a coherent structural framework within which steering committees, audit procedures, quality control practices may be incorporated. This helps minimise the risk inherent in systems development projects, and contributes to ensuring that all necessary activities are performed (cf. e.g., Ahituv et al., 1984; Bantleman & Jones, 1984; McDonald et al., 1986).
- The division of labour afforded by the phased approach inherent in most methodologies also has important implications. For example, the reduction of the development process into individual component phases helps cope with complexity. Also, this division of labour allows some economics of specialisation as different skills are required in different phases—analysis, design, coding, testing etc. (cf. Baskerville *et al*., 1992; Friedman, 1989; Olerup, 1991).

- From an epistemological perspective, methodologies allow for the systematisation and formalisation of knowledge relevant to systems development, thereby changing the subject-dependent knowledge of individual developers into an objective form. Thus, in theory at any rate, the knowledge of skilled developers who have worked on successful development projects can be acquired and classified, and then transferred to those less skilled (cf. Baskerville *et al.*, 1992; Stage, 1991; Stolterman, 1994).
- This formalisation of knowledge also allows for some standardisation in the development process, which in turn can improve coordination and communication among multiple developers on complex development projects, and also facilitates interchangeability among developers. Following a standardised development process may also ease subsequent maintenance (cf. Avison & Fitzgerald, 1988; Friedman, 1989; Holloway, 1989).

In addition, there are a number of complementary pressures which support the use of formalised system development methodologies. These pressures are typically at the macro-level and are briefly summarised here:

- Formalised methodological standards have been recommended by a number of national governments, including SSADM (UK, Ireland, Hong Kong, Malta), Dafne (Italy), Merise (France), NIAM (Holland), and Department of Defense (DoD) Std. 2167 (US). Governments are probably the largest consumers of software; consequently, this constitutes a major pressure in the industry to adopt these methodologies (cf. Coad & Yourdon, 1991; Downs et al., 1992).
- The Software Engineering Institute's programme for software capability evaluation (SCE) which has been the focus of much interest in the US also emphasises the use of formalised development procedures (Humphrey *et al.*, 1991). Software companies strive to achieve a high SCE rating as this confers legitimacy on their development approach and may afford them some competitive edge over rivals. Ironically, the development approach used by Microsoft—an approach which must be regarded as successful given the software products produced by the company—which recognises the critical importance of the individuals involved would only merit a Level 2 on the SCE assessment scale (Yourdon, 1994).
- In a similar vein, the desire for ISO-certification, keenly aspired to by many software development organisations, is also a source of pressure. ISO-certification is perceived as a source of legitimacy for the development process and an imprimatur of quality, but it also mandates the use of formalised developent approaches.
- It is also possible to discern a bias in the literature which views the use of formalised methodologies as an appropriate step towards solving the problems associated with systems development (cf. e.g., King, 1984; Page-Jones, 1991; Ramamoorthy *et al.*, 1986; Yourdon, 1991). This is especially problematic when one considers the circular pressure this creates, in that even though the literature may not reflect actual practice, it certainly influences it. Thus, developers who are not following the methodological prescriptions advocated in the literature may feel they are deficient in some way.