FORMALISED SYSTEMS DEVELOPMENT METHODOLOGIES: A CRITICAL PERSPECTIVE

(from *The Information Systems Journal*, 1996, Vol. 6, No. 1, pp. 3-23)

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

Abstract

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 presents some arguments and pressures which support the use of methodologies. Some evidence of the literature bias which favours methodologies is also provided. However, the problems associated with the use of methodologies have not perhaps received as much attention in previous research. This paper identifies a number of arguments and pressures which question the value of methodologies, and reports the results of a field study which investigated the role of methodologies in practice. The critical perspective adopted in this paper shows that, contrary to the predominant literature view, the assumption that increased adoption of methodologies would help address the problems inherent in systems development is by no means proven.

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

FORMALISED SYSTEMS DEVELOPMENT METHODOLOGIES: A CRITICAL PERSPECTIVE

Abstract

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 presents some arguments and pressures which support the use of methodologies. Some evidence of the literature bias which favours methodologies is also provided. However, the problems associated with the use of methodologies have not perhaps received as much attention in previous research. This paper identifies a number of arguments and pressures which question the value of methodologies, and reports the results of a field study which investigated the role of methodologies in practice. The critical perspective adopted in this paper shows that, contrary to the predominant literature view, the assumption that increased adoption of methodologies would help address the problems inherent in systems development is by no means proven.

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

1. Introduction

The problems inherent in systems development which have led to the coining of the term 'software crisis' are well-documented (cf. e.g. Brooks, 1987; Naur *et al*., 1976; Pressman, 1987; Shemer, 1987). However, the systems development issue continues to be one of central concern in the information systems field, and much has been prescribed in the quest for successful systems development. This paper begins by considering some of the arguments and pressures which support the use of formalised[1] systems development methodologies (SDMs). An examination of the literature reveals a two-fold bias, which, firstly, construes the "software crisis" as a problem arising from the sloppy, ad hoc and irrational approaches of systems developers in practice; and, secondly, views the solution to the software crisis in terms of more widespread adoption of rigorous and formalised systems development methodologies. Some influential sources illustrating this literature bias are presented. Following this, some of the arguments and pressures which mitigate against the use of formalised SDMs are presented.

---

[1] The term 'formalised' is used here to denote formally-defined, brand-named or published development methodologies, of which there are many examples in the literature, rather than an ad-hoc approach to systems development, of which there are many examples in practice. Some writers use the term 'formal' in this context. However, this leads to confusion with those methodologies which have a mathematical basis for specification and design, which are also labelled as formal.

The issues discussed in this paper have been synthesised from a number of sources. In some cases the arguments have been mentioned by previous researchers in the literature—these are duly credited. Other issues have arisen from the author's long commercial experience (more than 12 years) as a systems developer in several different organisations. A final source has been a field study conducted by the author which investigated the nature of systems development in practice (cf. Fitzgerald, 1994a). This study involved interviewing a limited number of systems developers about relevant development issues. The interviewees were drawn from five different organisations and all were very experienced in terms of systems development (between 8 and 18 years development experience with an average of 13 years) and were involved at a senior level in their IS departments. These empirical findings, although they must be regarded somewhat tentatively since they are based on a limited sample, are quite interesting and will be investigated further, as the field study is part of an ongoing research project which will explore these issues in more detail.

2. Arguments and Pressures in Favour of Formalised Methodologies

Early efforts at systems development often relied on unsystematic and random methods (Olerup, 1991, Yeh, 1991), although some systematic approaches to systems development were actually available (cf. Colter, 1984; Couger, 1973, Taggart and Tharp, 1977). By the end of the 1960s, the systems development problems which gave rise to the term, 'software crisis', were widely acknowledged. Simply stated, the software crisis referred to the fact that systems took too long to develop, cost too much, and did not work very well. As the complexity of the systems which needed to be developed was increasing (Friedman, 1989), the situation would inevitably be further exacerbated. A more disciplined and methodological approach to systems development was thus advocated. In this section, some of the fundamental concepts underpinning the general rationale for formalised methodologies are discussed. Also, some of the pressures which support the adoption of formalised methodologies are identified. These are summarised in the framework in Fig. 1.

Fig. 1: Arguments and Pressures Supporting Formalised Methodologies

Conceptual Underpinnings:

- Reductionist subdivision of complex development process
- Facilitation of project management and control, thus minimising risk and uncertainty
- Purposeful framework for application of techniques and resources
- Economic rationale: skill specialisation and division of labour
- Epistemological rationale: structural framework for the acquisition and systematisation of knowledge
- Standardisation of the development process:
  - Facilitation of interchangeability among developers
  - Increased productivity and quality

Pressures for Increased Formalism:

- Desirability of ISO-certification
- Government development 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 Software Engineering Institute
- Literature bias:
  - Irrational practitioners assumed to be reason for problems in systems development
  - Formalised methodologies seen as solution to development problems

2.1 Conceptual Underpinnings of Methodologies

Reductionist Subdivision of Development Process

Olerup (1991) defines a methodology as a strategy which implies a subdivision of the development process. The 'divide and conquer' principle has long been a feature of mathematical and engineering approaches to solving complex problems, where it has had significant success, and the early development methodologies were very much influenced by technical and engineering disciplines (Dumdum & Klein, 1986; Goldkuhl & Lyytinen, 1984). An important early contribution was that of Langefors (1973) who, in arguing for a more formal approach to systems development, outlined the foundations for a theory of information systems. He conceptualised systems development as a rational and scientific process, and proposed a subdivision of the development process into deciding *what* an information system must do, and *how* it should do it.

By adopting this rational scientific view, prevalent in many other disciplines, the complex development process is broken into the broad categories of analysis of requirements, design of a solution, and implementation of that solution (Olerup, 1991). Other broad categories to be added include an initial investigation of the organisational problem context, and a final maintenance stage (Downs *et al*., 1992). Systems development is thus conceptualised as essentially a linear model, albeit with some iteration across stages, relying on phases ordered in time and leading to increasingly-formal defining documents until the system eventually emerges.

Facilitation of Project Management and Control

One of the widely-cited benefits of the phased approach to systems development is that it makes the development process more amenable to project management and control (cf. e.g. Ahituv *et al*., 1984; Avison & Fitzgerald, 1988; Floyd, 1987; Friedman, 1989). Some visibility into the progress of the development process is afforded, in that at the end of each phase there is an opportunity to review progress and to monitor actual costs and benefits. These can then be compared with expected figures, thereby helping to minimise the risk inherent in systems development projects. Thus, methodologies facilitate a control dimension by providing a coherent framework within which steering committees, walkthrough techniques, audit procedures, quality control and inspection practices can be incorporated (Ahituv *et al*., 1984). Indeed, Griffiths (1978) describes the project control role of a methodology in strong terms:

> What makes or mars a (development) methodology in the market-place is how close it comes to satisfying the principal project management requirements.

Purposeful Framework for Application of Techniques and Resources

Development methodologies also afford a structural framework for the development process by providing a taxonomy of the necessary component activities of development. Redundant, irrational and counter-productive activities can therefore be eliminated. Additionally, the grouping of activities improves efficiency by ensuring that necessary work is completed without oversight. Ahituv *et al*. (1984) propose a number of dimensions to development including the activity dimension (individual stages of the life-cycle), the control dimension (code inspections, walkthroughs, etc.), the human resource dimension (analysts, designers, programmers) and the non-human resource dimension (e.g. computer time, materials, capital investment). Again, a development methodology can provide the appropriate structural framework to utilise these resources. Following a similar argument, Bantleman and Jones (1984) adapt the systems theory maxim, suggesting that "the whole (the methodology) is greater than the sum of its parts (the techniques).

Economies Afforded by the Division of Labour

Another implication which arises from the reduction of the development process into a series of individual phases is that a division of labour is possible. Since each phase is comprised of different tasks requiring different skills—analysis, programming, technical writing, for example—some economies of specialisation are afforded. For example, not every labour category participates in every development stage. Thus, the finer granularity achieved through the subdivision of the labour process allows organisations to use differential pay rates depending on the particular skills required, rather than having to pay for all-round ability—termed rather ironically the 'Babbage' effect (Friedman, 1989).

Systematisation of Knowledge and the Epistemological Rationale

Closely related to the division of labour rationale is the notion of systematising knowledge. Both are fundamental principles of Taylor's scientific management which sought to find the 'one best way' to perform work, and which had the underlying objective of changing all subject-dependent knowledge into knowledge in an objective form. Stolterman (1994, p.535) describes it as follows:

> The idea was that knowledge separated from a specific subject is valued more because it makes knowledge into something possible to handle and manipulate. The transfer of knowledge from the designer to the method leads to a lesser dependency on specific persons.

Methodologies allow knowledge to be stored, systematised, disseminated and exchanged (Stage, 1991). This formalisation of knowledge allows the transfer of knowledge from skilled and knowledgeable developers to those less skilled, effectively shortening the learning curve for the latter.

Baskerville *et al.* (1992) also propose an epistemological rationale for development methodologies referring to the structural framework which methodologies provide, thereby allowing professionals working in the field to acquire and classify knowledge. Indeed, Holloway (1989) suggests that this formalisation of knowledge implicit in a development methodology ensures that the developer "does not have to reinvent the wheel for each project".

Standardisation of the Development Process

One of the implications of this formalisation of knowledge is that it allows some standardisation of the development process. This facilitates interchangeability among developers—particularly important given that developers have traditionally shown more loyalty to their profession than to the individual organisations within which they have worked (Land & Hirschheim, 1983). Following a standardised development process may improve co-ordination and communication among the participants in complex development projects. It has also been suggested that this can ease subsequent maintenance problems (Avison & Fitzgerald, 1988).

Downs *et al.* (1992) argue that development methodologies promote discipline among developers by specifying a structure for the development process. This is suggested to lead to increased productivity, in that specific resources may be provided to ensure that systems are built faster (Avison & Fitzgerald, 1988). Another benefit cited is that of increased quality, which follows from the issues just discussed, and also from the fact that systems are developed to be flexible and are adequately documented.

2.2 Pressures in Favour of Formalised Methodologies

Complementing these significant arguments in favour of formalised methodologies, there are a number of very influential sources of pressure in favour of the use of formalised approaches (see Fig. 1). For example, at a broad level the ISO-certification process, much sought after by organisations, requires the use of formalised development processes. Thus, formalised methodologies are perceived as playing a useful role in this regard, although this may be seen as a case of the tail wagging the dog in so far as systems development is concerned (cf. Fitzgerald, 1994a). Also, major institutions such as the UK government have mandated the use of the SSADM (Structured Systems Analysis and Design Method) methodology for systems development. SSADM is now used on projects totalling billions of pounds each year (Downs *et al.*, 1992), and this causes a significant pressure in the industry to move in this direction—a fact which is borne out in the large numbers of organisations supplying consultancy, training, and CASE tools supporting the SSADM methodology (Downs *et al.*, 1992; Holloway, 1989). Several other national governments have also adopted

SSADM as the required development methodology, while countries such as France, Holland, and Italy have their own formalised development methodologies.

Similarly in the US, the Department of Defense (DoD) have established development standards (e.g. DoD Std. 2167) for software development which developers working on DoD projects must follow. These standards have emerged from several years research and are intended to allow the DoD more visibility and control with respect to the development process (Coad & Yourdon, 1991). The DoD have recently collaborated with the Software Engineering Institute on the Software Capability Evaluation (SCE) programme. This programme is concerned with assessing the capability of organisations to produce quality software in a timely and repeatable fashion and it has generated intense focus and some controversy in the US software industry (Bach, 1994; Bollinger and McGowan, 1991). However, this programme places great emphasis on adherence to formalised development procedures. Indeed, its advocates suggest that effective development requires that all steps in a development methodology should be carried out regardless of circumstances (Humphrey *et al*., 1991). This is a controversial issue as it fails to take account of contingencies of any particular situation—a factor discussed below.

2.2.1 Literature Bias

The general implication in much of the literature is that practitioners are moving, albeit very slowly, towards more widespread adoption of formalised methodologies, and that this should help to alleviate the problems inherent in systems development. The following quotes represent some of the prevalent thinking on this issue. These quotes are divided into two sections. The first set depicts the poor regard in which practitioners are held—their irrational practice being viewed as the main source of systems development problems. The second set illustrates the widespread belief in the literature that the formalisation of practice by adherence to methodological approaches is as an appropriate step towards a solution.

Irrational Practitioners at Fault

> The average coder...(is)...generally introverted, sloppy, inflexible, in over his head, and undermanaged.
>
> (Boehm, 1976)

> The majority of software development organisations operate in a 'Realm of Darkness', blissfully unaware of even the rudimentary concepts of structured analysis and design.
>
> (Yourdon, 1991)

> ...the lack of professional discipline among the great unwashed masses of systems developers.
>
> (Ward, 1992)

Formalised Methodologies are an Appropriate Solution

> The major forces working for diversity appear to be ignorance, lassitude, deficiency...Employers have been very relaxed about setting and enforcing local standards for their employees to follow.
>
> (Chapin, 1980)

> The problems faced in developing large software include...enforcing a methodology on the developers.
>
> (Ramamoorthy *et al*., 1984)

> The first effect of teaching a methodology—rather than disseminating knowledge—is that of enhancing the capacities of the already capable, thus magnifying the difference in intelligence.
>
> (Dijkstra, 1972)

> Losers consist of unnamed, unspecified, up to the individual, or 'written-but not formalised' types of methodologies.
>
> (Zolnowski & Ting, 1982)

> The use of a formalised SDM is perceived as positive and well advised.
>
> (Jenkins *et al*., 1984)

> Software development in professional communities often is completely ad hoc, or at best supported by structured methods...and JSD.
>
> (Plat *et al*., 1991)

> One startling and somewhat disturbing observation is that many (systems development) methods are used very little
>
> (Palvia & Nosek, 1993)

These are just a small sample which illustrate the widespread bias in the literature. However, one of the effects of this bias has been the "search for the holy paradigm" (Hackathorn & Karimi, 1988) as researchers have striven to identify the methodology which represents the 'one best way' to develop systems. This has led to a trend whereby competing methodologies have been developed, a trend labelled as "methodolotary" (Beynon-Davies, 1989). Tagg (1983) also discusses this phenomenon, drawing a parallel with the proliferation of religious sects:

> Despite the fact that they are 95% agreed in their aims and their broad areas of getting there, they nevertheless manage to stay separate. Each sect jealously guards its own style and magic ingredients.

However, this literature bias 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 creating a significant additional pressure in support of the use of formalised methodologies.

3. A Critical Consideration of Formalised Methodologies

Estimates vary as to the number of different published methodologies that exist. Longworth (1985) put the figure at more than 300, while a more recent estimate suggests that there may be more than 1000 (Jayaratna, 1994). However, the question arises as to whether there are actually that many substantially different ways to develop systems. Indeed, the assumption that formalised methodologies actually represent the most appropriate means of solving the software crisis is open to question. Systems development methodologies are attractive and have an intuitive appeal, but a systems development methodology is not systems development, rather it is a framework for organising the system development process. Indeed, a large part of the methodology may go towards justifying the methodology itself. This section presents a number of criticisms which serve to question the validity of formalised methodologies, and also considers a number of pressures which mitigate against the use of formalised methodologies. These are summarised in the framework in Fig. 2.

Fig. 2: Arguments and Pressures Mitigating Against Formalised Methodologies

Definitional Anomalies:
- Problems as to what exactly constitutes a methodology
- Fundamental differences between some methodologies and artificial, contrived differences between others

Generalisation Without Adequate Conceptual and Empirical Foundation

Inadequacies of Rational Scientific Paradigm:
- Systems development is not actually an orderly rational process but most methodologies view it as such
- Over-emphasis on technical rationality at the expense of softer, social aspects

Goal Displacement:
- Slavish and blind adherence to methodology while losing sight of the fact that development of an actual system is the real objective

Assumption that Methodologies are Universally Applicable:
- Failure to recognise contingency factors and the uniqueness of every development situation

Inadequate Recognition of Developer-Embodied Factors:
- Methodologies do not cater for factors critical to successful development, such as individual creativity and intuition, or learning over time

Pressures for New Approaches to Systems Development:
- Changing Nature of Business Environment
  Short-term needs dominate given 'faster metabolism' of business today
- Altered Profile of Systems Development Environment
  Configuration development: integration of package software to incorporate local practices
- Rapid Application Development
  Replacement of large-scale monolithic approaches with approaches based on the 'frequent tangible results' philosophy

3.1 Arguments Which Mitigate Against the Use of Formalised Methodologies

Definitional Anomalies

When it comes to deciding what actually constitutes a development methodology, the definitional quagmire so common in the computing field becomes apparent. Methodologies have been variously defined in terms of models, management practices, technical practices, tools, training procedures and so on (De Grace & Stahl, 1990). In fact, as researchers have pointed out, the term itself is misused as it actually means 'study of method' (Olle *et al.*, 1991; Stamper, 1988). There is a certain elaborate and laudatory connotation to the term 'methodology' which makes it sound more impressive than the more correct, but less grandiose, 'method'. However, 'methodology', even if not now considered politically correct, is widely used in the literature and practice, and has been used in this paper as a consequence.

Even though a large number of methodologies are in existence, there are relatively insignificant and artificially contrived differences between some methodologies, while there are fundamental differences between others. In the former category, for example, Constantine, one of the founding figures of the structured approach, admits that different development approaches are actually "based on product differentiation, personal ego, and territorial imperative" (Constantine, 1989). Indeed, this policy of striving to preserve an artificial methodology differentiation is mentioned by Yourdon (1991, p.42) who describes it in clear terms:

> In 1975 Michael Jackson's JSP methodology was greeted with some curiosity and mild interest by the technical staff at Yourdon inc., but the marketing staff saw it as a distinct threat and attacked it with the fervor of white blood cells going after a virus. 4GLs, prototyping, data modeling, and all manner of alternative approaches to systems development were also threats.

However, other researchers point to the fundamental differences between methodologies in terms of philosophy, objective and techniques (Avison & Fitzgerald, 1988). Also, methodologies differ fundamentally in paradigm. Some are 'hard' scientific and rationalistic—the structured approach or SSADM, for example, while others are 'soft' human-oriented ones (cf. Lyytinen, 1987). Methodologies also differ considerably in focus, as some do not cover requirements analysis while others do not cover implementation (Sakthivel, 1992).

Generalisation without Adequate Conceptual and Empirical Foundation

Methodologies have been constructed by abstracting practices and techniques from a successful development project, and formalising these into a set of guidelines and procedures to form a development methodology, but there may be little philosophical justification (Maddison *et al.*, 1984). Apropos to this, Lyytinen (1986) identifies an inadequate conceptual base as a weakness in many methodologies, pointing out that most have "an

ambiguous and narrow conception of the phenomena IS developers confront", with few methodologies providing definitions of basic terms.

Ward (1992) describes the general process which should be followed before a method or technique may be recommended for professional use. Firstly, there needs to be a sound conceptual foundation. Generally, this is achieved in the physical and social sciences through a peer review process in which experts in the field, who have no vested interest in the method or technique, examine it for theoretical shortcomings. The second strand of the process is to obtain empirical evidence of usability, which requires the method or technique to have been successfully applied to a non-trivial problem situation. However, the structured approach to systems development, the "most popular systems development methodology in North America and Europe" (Yourdon, 1991), was based on the intuition of its founders that it would work rather than on any real-world experience (Ward, 1992). According to one of the founders, early investigations of the structured approach were just "noon hour critiques" (Yourdon & Constantine, 1977). Indeed, several areas of weakness in the structured approach have been identified (cf. Bansler & Bodker, 1993; Coad & Yourdon, 1991; Henson & Hughes, 1991; Ward, 1991, 1992).

Inadequacies of the Rational Scientific Paradigm

The underlying paradigm for many development methodologies is the scientific reductionist one (Baskerville *et al*, 1992). They rest on the *a priori* assumptions that the solution can be arrived at through a sequence of technically devised steps (Bubenko, 1986; Hackathorn and Karimi, 1988), and that the developer can obtain detailed knowledge about the problem situation (Giddings, 1984). Both of these assumptions are critically examined here.

The conceptualisation of development as following a linear sequence of phases is implicit in the systems development life-cycle (SDLC). However, the SDLC approach has come in for considerable criticism (cf. e.g. Glass, 1991; Henson & Hughes, 1991; McCracken & Jackson, 1981; Taylor & Standish, 1982). Summarising very briefly, the principal weakness of the SDLC is that later phases depend on the successful completion of earlier phases which requires "perfect foresight" (Henson & Hughes, 1991). However, later phases are much more likely to take place in the context of "imperfect predecessors" (Taylor & Standish, 1982).

Researchers have noted that systems development in practice is not an orderly systematic phased process (Wastell & Newman, 1993) but tends to happen “all at once” (De Grace & Stahl, 1993). Developers do not develop systems by completing a single task and moving on to the next task following a rational sequence; rather they will have a range of tasks not concluded due to the interdependent nature of development. Swartout and Balzer (1982) argue that there is an inevitable intertwining of specification and design, but in the phased

approach to development the *what* of requirements specification is strictly separated from the *how* of design. In practice, however, this has proven to be an inadequate and infeasible. Shemer (1987) uses a jigsaw analogy, suggesting that the ultimate design is not achieved in a rational top-down manner. Rather, information is obtained in random order; some design is done bottom-up by guessing at a local pattern, and, simultaneously, some design is done top-down by following an apparent pattern. A study by Zelkowitz (1988) lends validity to this, reporting that 50 percent of design activity occurs in phases other than the design phase.

This view of the systems development process as an intertwined, all-at-once process was also borne out in the author's field study (op. cit.), as formalised SDMs were modified in all the organisations in which they were in use. Indeed, the practice of systems development seemed to have more similarities than differences in the organisations studied, as interviewees reported that they generally omitted, or paid very scant attention, to early phases such as planning, and later phases such as testing, and some form of prototyping was generally taking place. Also, it was recognised that the development approach often differed from that prescribed by the methodology, but that the work was later 'retrofitted' to comply with methodology requirements, thus creating the illusion that the methodologically-prescribed sequence was followed.

The assumption that the developer can obtain detailed knowledge about the problem situation (Giddings, 1984) is questioned by Jones and Walsham (1992) who argue that there are "limits to what *can* be known", both from a theoretical and a practical standpoint. They further contend that there are "limits to what *should* be known", both from a social and ethical point of view. Stage (1991) argues in a similar vein, suggesting that some of the knowledge relevant to the problem situation may be impossible to express due to limitations in the actual descriptive techniques available. Boland (1979) also questions the extent to which an organisational problem exists as an independent reality that can be modelled beforehand, and he identifies the critical importance of how the situation is interpreted by the actors in the situation.

As already discussed, a major rationale underpinning the belief in development methodologies is that practitioners are perceived as biased towards following irrational design practices, as discussed above, and thus require methodological support to provide a rational basis for development. However, methodologies tend to conform to a very particular technical rationality in which, to use Schon's (1987) terms:

> Rigorous professional practitioners solve well-formed instrumental problems by applying theory and technique derived from systematic, preferably scientific knowledge.

Several researchers have questioned the validity of viewing systems development as a rational process. For example, Robey and Markus (1984) argue that while the various phases of systems development can be explained by rational motives, they can also be explained as

political rituals which are used to negotiate the private interests of the various parties concerned. Thus, the stages in systems development can be explained by two diametrically-opposed sets of motives—rational and political. However, the rational motives are the ones assumed in many methodologies, and consequently, they do not cope well with social and human factors (Bostrom and Heinen, 1977; Floyd, 1987; Goldkuhl and Lyytinen, 1984). Systems development is not just a technical process; social needs to be considered also. Yet, researchers have suggested that most development methodologies only pay lip-service to social aspects and have argued that treating systems development as a purely-technical process is a "recipe for disaster" (Hirschheim and Newman, 1991). Consequently, much research has focused on softer approaches to systems development which counter these criticisms (cf. e.g. Checkland, 1984; Dumdum & Klein, 1986; Hirschheim and Newman, 1991; Mumford, 1984).

Goal Displacement: Confusing the Menu with the Meal

One of the most harmful potential implications arising from the use of development methodologies is that of goal displacement. The goal displacement phenomenon basically refers to the situation whereby developers become preoccupied with following the methodology at the expense of actual development; that is, the developer becomes engrossed in blind and slavish adherence to the methodology and loses sight of the fact that development of a system is the real goal. De Grace & Stahl (1990) illustrate this phenomenon in their analysis of the development documentation from several projects which showed that more than 90 per cent of the content was devoted to reporting about the status of the development project, while less than 10 per cent described what was to be done or how to do it.

The goal displacement phenomenon was identified as a major problem in the author's field study (op. cit.), with one interviewee referring to the methodological requirement to produce entity life histories to illustrate this problem:

> Drawing entity life history models can be an industry in itself, and, in the end, it may only serve to clarify the obvious.

The clear moral is that when it takes so much effort to comply with methodological requirements, it may be more worthwhile to spend the time on actual development. There is a need therefore to distinguish between form and substance when following a methodology. Ironically, even in the Garmisch conference of 1968 where the term "software crisis" was coined, and the need for more rigorous development procedures strongly voiced, some participants warned of the potential dangers of the rigid application of methods. Smith (quoted in Naur *et al.*, 1976, p.88) described his experience with the rigid adherence to methods in his organisation:

> They begin with planning specifications, go through functional specifications, implementation specifications etc. etc. This activity is represented by a PERT chart with many nodes. If you look down the PERT chart you discover that all the nodes on it up until the last one produce nothing but paper. *It is unfortunately true that in my organisation people confuse the menu with the meal*. (italics added)

Assumption that Methodologies are Universally Applicable

Researchers have criticised the tendency in Western society to view certain procedures as universally applicable (Kindler & Kiss, 1984). However, Wastell and Newman (1993), on the basis of a number of empirical case studies of systems development, conclude that methodologies should not be viewed as universally applicable. The latter has been labelled the "one size fits all" presumption (Chikofsky, 1989), and does not give due consideration to the contingencies of each development situation (Avison *et al*., 1988; Curtis *et al.,* 1988). In practice, developers frequently do not apply the methodologies in their complete form as specified (Chikofsky, 1989, Jenkins *et al*., 1984). Developers omit those aspects of the methodology that do not seem to suit the contingencies of the situation. For example, the US Department of Defence, whose strong advocacy of formalised methodologies has already been discussed, now recommend tailoring of their recommended methodological standard, DoD Std 2167, to suit the particular development situation (Coad & Yourdon, 1991; DeGrace & Stahl, 1990).

As discussed above, part of the rationale behind the use of development methodologies is to facilitate project management and control of the development process. Indeed, methodologies have an intuitive appeal for management by seeming to introduce rigour into the development process. However, Glass (1991) rejects this as illusory, comparing the use of a development methodology to the effect of the Maginot line—giving the "illusion of quality but hiding violations". Systems development in practice is actually an unstructured, evolutionary process. Yet, development methodologies attempt to impose complete solutions when the minimum are not yet well-defined. Indeed, an idealised approach to systems development as portrayed in a methodology may be seriously flawed since it omits the fact that failure is essential to human learning. This learning over time endows the developer with a repertoire of strategies to apply in different development situations—an issue discussed below.

Inadequate Recognition of Developer-Embodied Factors

The importance of individual differences in systems development has been acknowledged by several researchers. Boehm (1981) reports that people factors have more than six times greater effect on development productivity than the use of software tools. Brooks (1987) suggests that systems development is a creative process and that a methodology cannot "inspire the drudge" and he recommends that processes be put in place to nourish creative

people. He states that few fields have such a large gap between best current practice and average practice. However, the varied skill levels of different developers is not acknowledged in formalised development methodologies. For example, one of the explicit goals of the Jackson Systems Development (JSD) methodology is to eliminate personal creativity from the development process (King and Pardoe, 1985). Indeed, this is cited as a strength of the methodology. Likewise, Colter (1982) praises the contribution of development methodologies for moving the field "from the old artsy processes", and he laments the fact that "much is still left to the abilities of the designer". However, methodologies merely focus on the trappings of design rather than on its essence which is actually the cognitive activity in the mind of the developer. It is important, therefore, not to lose sight of the fact that it is *people*, and not *methodologies*, who actually develop systems. At best, methodologies provide only a useful organising framework.

Nor do methodologies allow for the learning experience and greater problem domain knowledge that developers gain over time. Each systems development project is generally undertaken as if it was unique. Yet, in a comparative study of successful and unsuccessful systems analysts, Vitalari and Dickson (1983) emphasise the importance of learning over time. They conclude that developers acquire a repertoire of strategies to apply in different systems development situations. This is in accord with Davis and Olson (1985) who suggest that developers gain more domain knowledge over time and that this is a vital factor in successful systems development. To view systems development as an orderly progression from requirements analysis to a solution designed purely around those requirements is to miss the critical synergy between developer and user. Curtis *et al.* (1988) have suggested that both the developer and user learn through a dialectic approach, in that by hearing about potential capabilities of the system, users envision new features which can then be incorporated into the system.

In calling for a paradigm change, Floyd (1987) criticises methodologies as static rule-systems operating in a stereotyped, standardised manner. She calls for a process-oriented perspective with a central role for the learning process, where methods

> appear as *second-order learning processes*, based on an ever-growing class of individual past learning processes in actual projects, using guidelines offered by method authors and tailored to the situation in hand, and preparing us for future learning processes in developing software.

Also, an interesting finding emerges from an empirical study by DeMarco and Lister (1989) which shows that even in organisations where methodologies are rigidly enforced, there is very poor convergence on design style among different developers. This again reinforces the point that development methodologies cannot be inflicted on developers.

An early, albeit cautious, questioning of the role of methodologies was that of Bubenko (1986 p.299) who put the case for prominence of soft, people issues in the systems development context:

> We should realise that a design will always have an artistic component and that not everything can be 'prescribed'...the quality of a design is totally dependent on the competence of the designer to the extent that one sometimes wonders about the utility of using a methodology at all.

3.2. Pressures for New Approaches to Systems Development

Just as there are a number of pressures in favour of formalised methodologies (as discussed above), there are also a number of pressures for new and radical approaches to systems development. These pressures have to do with the changing nature of the business environment in general, the changing profile of the systems development environment in particular, and the need for more rapid delivery of systems to meet short-term needs. These issues are discussed in turn in this section.

Changing Nature of Business Environment

The accelerating pace of change characteristic of the business environment facing organisations today is a common theme in the contemporary literature. The prevailing view is aptly summarised by Rockart and De Long (1988) in their reference to the "faster metabolism of business today" which requires organisations to act more effectively in shorter time-frames. Baskerville *et al*. (1992) discuss the relevance of this change in the nature of the business environment to the systems development issue. They argue that formalised methodologies are oriented towards large-scale development with a long development time, but the continuous change that organisations are now faced with, means that short-term needs dominate, and these in turn mean that the economics of formalised systems development is dwindling. Baskerville *et al*. further argue that the use of formalised methodologies are actually hindering essential organisational needs. Developers, thus, do not have the luxury of being able to patiently follow a comprehensive formalised methodology. This view is supported by other researchers who suggest that the truncation of some phases in the development process is seen as inevitable (Brown, 1985).

Again, this was a factor identified by interviewees in the author's field study (op. cit.) where the inertia that a large-scale formalised methodology can impose on the development process was perceived as a major stumbling block. The development process can thus be over-intellectualised, and in many circumstances a methodology may prescribe an overly-complex approach where a more parsimonious one is appropriate. In these circumstances, the methodology becomes something of a 'mother hen'—overly cautious and conservative, and leading to an inflexible and cumbersome development process.

Changing Profile of Systems Development Environment

In keeping with the changing nature of the business environment, there is a concomitant change in the profile of the systems development environment. A fundamental restructuring

is taking place in the software industry as companies are relying far less on in-house development of systems, but are pursuing alternative strategies such as buying package software or outsourcing systems development. Bansler and Havn (1994) characterise this as the "industrialisation of information systems development" and suggest that a similar process has taken place in other more mature branches of engineering. They discuss the practical effects of this altered profile of development, and suggest that users are constructing their own information systems out of several suppliers' modules, integrated together by specially-written software which incorporates local practice, a process which they label "configuration development". This altered development profile further serves to question the economics of formalised development as current methodologies do not address the process of configuration development.

Rapid Application Development

Researchers in the area of rapid application development make similar points in advocating a change in development approach. Folkes and Stubenvoll (1992) cite the change in the nature of the demand for systems, and they argue for a change to an accelerated development approach to reflect this. Researchers have estimated a need for a ten-fold increase in systems development productivity to meet the demand for systems (Verity, 1987), and, in keeping with this, Hough (1993) calls for a change in philosophy from large-scale monolithic approaches:

> In the past, this philosophy has framed application development as a process that follows a long, unbending path, resulting in software that is anything but "soft." Many applications are developed using a monolithic approach even though business needs and requirements change over time.

Hough cites evidence from an IBM study which discusses the problems that can arise from monolithic development approaches. These include a slowed pace of development; treatment of the project as a "career"; lost sight of the original business problem; loss of interest by the development staff; deterioration of morale among individuals participating in the project. Hough proposes a rapid delivery approach to development as a solution to these problems. His model draws on a Japanese model of product evolution, a model also used by De Grace and Stahl (1990) in their discussion of alternative approaches to systems development (cf. Takeuchi & Nonaka, 1986). The main thrust of rapid delivery is to produce frequent tangible results, that is, every few months some functional capability is delivered—a concept which is also central to Gilb's (1988) incremental engineering approach.

4. Conclusion: Time to Move the Lamppost!

Boehm (1988) has criticised the focus of much of the research in the software field as being directed towards certain well-understood areas while neglecting other areas which are less well-understood but equally important. He proposes the analogy of a drunk losing his

watch and looking for it under the light of a lamppost because the light is best there, even though it had been lost somewhere else. This analogy is quite appropriate given the vast body of research which has focused on evaluating and comparing formalised methodologies (cf. Fitzgerald, 1994b), in that it may be case of looking under the lamppost. Certainly, Lewis and Oman (1990) claim that in 20 years, systems development has "evolved to little more than a black art", and Wasserman's contention (1981) that the greatest boost to systems development productivity would be to teach programmers the skills of touch-typing has yet to be refuted.

While the rationale behind the use of formalised development methodologies is persuasive, the arguments against the use of formalised methodologies are also compelling. These dichotomous arguments thus bring about a dilemma for systems developers as to whether to adopt a formalised methodology or not (Fitzgerald, 1994b). However, researchers have pointed out that in practice, many practitioners do not use a formalised development methodology (Avison & Fitzgerald, 1988; Page-Jones, 1991; Ward, 1992). A number of reasons have been put forward to explain this. For example, it has been suggested that the failure to use a formalised methodology is due to a "wealth of ignorance" among the "great unwashed masses" (Ward, 1991), and the failure of practitioners to use development methodologies is seen as a weakness on their part (Page-Jones, 1991). Also, it has been suggested that it takes about 15-20 years for technology transfer to achieve sufficient maturity for general use (Chikofsky, 1989), and that this is what has delayed the adoption of methodologies. However, as discussed above, there is an implicit assumption that the failure of practitioners to use formalised methodologies has been to the detriment of systems development.

Non-use of a methodology is not a licence to conduct development in a sloppy or careless manner. Those who suggest that the failure of practitioners to use a formalised methodology is due to ignorance or a lack of awareness on their part may not be presenting a totally-accurate picture. An appropriate analogy might be that of Picasso dispensing with conventional artistic perspective, but from a position of superior knowledge. Likewise, many practitioners may be well aware of the limitations of formalised methodologies and may have rejected them for pragmatic reasons. As a practitioner with over 12 years experience of systems development in different organisations on many different applications, the author has yet to witness a development project where a formalised development methodology was faithfully adhered to. In practice, situations will inevitably arise where the developer needs to step outside the methodology, but formalised methodologies often serve to impose a considerable inertia on the development process. Indeed, the degree of inertia may be proportional to the degree of formality of the methodology.

The next stage of this research will be involve further investigation of the issues raised here. By addressing these issues, the research should thus help to ascertain whether practitioners are indeed moving towards more formalised development methodologies as has been suggested and, indeed, recommended; or whether there is a sense in which systems development is perhaps beyond method in some circumstances.

Acknowledgements

The author would like to acknowledge the insightful suggestions and support of Frédéric Adam, Guy Fitzgerald, Nimal Jayaratna, Trevor Wood-Harper in the evolution of this work, and also the useful and constructive comments of a number of anonymous reviewers.

References

Ahituv, N., Hadass, M. and Neumann, S. (1984) A flexible approach to information system development. MIS Quarterly, June, 69-78.

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, 379-380.

Bach, J. (1994) The immaturity of the CMM, *American Programmer*, **7**, 9, 13-18.

Bansler, J. and Bodker, K. (1993) A reappraisal of structured analysis: design in an organisational context, *ACM Transactions on Information Systems*, **11**, 2, 165-193.

Bansler, J. and Havn, E. (1994) Information systems development with generic systems, in Baets, W. (Ed), *Proceedings of Second European Conference on Information Systems*, Nijenrode University Press, Breukelen, 707-718.

Bantleman, J. and Jones, A. (1984) Systems analysis methodologies: a research project. In Bemelmans, T. (ed.), 213-227.

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.), 241-269.

Bemelmans, T. (ed.) (1984) *Beyond Productivity: Information Systems Development for Organisational Effectiveness*, Elsevier Science Publishers B.V., North Holland Press.

Beynon-Davies, P. (1989) *Information Systems Development*, Macmillan, London.

Boehm, B. (1976) Software engineering. *IEEE Transactions on Computers*, **25**, 12, 1226-1241.

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**, 259-275.

Bollinger, T. and McGowan, C. (1991) A critical look at software capability evaluations. *IEEE Software*, **July**, 25-41.

Bostrom, R. and Heinen, J. (1977) MIS problems and failures: a socio-technical perspective--Part I: the causes. *MIS Quarterly*, **September**, 17-32.

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

Brown, P. (1985) Managing software development. *Datamation*, April 15, 133-136.

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

Chapin, N. (1981) *Flowcharts*, Auerbach, New Jersey.

Checkland, P. (1984) Systems theory and information systems. In Bemelmans, T. (ed.), 9-21.

Chikofsky, E. (1989) How to lose productivity with productivity tools. *Proceedings of 3rd IFAC/IFIP Workshop*, Indiana, US, 1-4.

Coad, P. and Yourdon, E. (1991) *Object-Oriented Analysis*, (2nd Edition), Yourdon Press, New Jersey.

Colter, M. (1982) Evolution of the structured methodologies. In Couger *et al*., (1982) *Advanced System Development Feasibility Techniques*, Wiley & Sons, New York, 73-96.

Constantine, L. (1989) The structured design approach. *Byte*, **April**, 232-233.

Couger, J. (1973) Evolution of business systems analysis techniques. *Computing Surveys*, **5**, 3, 167-198.

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

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

DeGrace, P and Stahl, L. (1990) *Wicked Problems, Righteous Solutions: A Catalogue of Modern Software Engineering Paradigms*. Yourdon Press, Prentice Hall, Englewood Cliffs, New Jersey.

DeGrace, P. and Stahl, L. (1993) *The Olduvai Imperative: CASE and the State of Software Engineering Practice*, Yourdon Press, Prentice Hall, Englewood Cliffs, New Jersey.

DeMarco, T. and Lister T. (1989) Software development: state of the art v. state of the practice. *11th International Conference on Software Engineering*, 271-275.

Dijkstra, E. (1972) The humble programmer. In Yourdon, E. (1979) *Classics in Software Engineering*. Yourdon Press,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, 393-420.

Fitzgerald, B. (1994a) Whither systems development: time to move the lamppost, in Lissoni, C., Richardson, T., Miles, R., Wood-Harper, T. and Jayaratna, N. (Eds) *Proceedings of Second Conference on Information Systems Methodologies*, Edinburgh, 373-380.

Fitzgerald, B. (1994b) 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, 691-706.

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.

Folkes, S. and Stubenvoll, S. (1992) *Accelerated Systems Development*, Prentice Hall, London.

Friedman, A. (1989) *Computer Systems Development: History, Organisation and Implementation*, Wiley & Sons, Chichester.

Giddings, R. (1984) Accommodating uncertainty in software design. *Communications of the ACM*, **May**, 428-434.

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.*et al*., 79-94.

Griffiths, S. (1978) Design methodologies - a comparison, *Infotech State of the Art Report*, **41**, 2. Cited in Maddison *et al*., (1984).

Hackathorn, R. and Karimi, J. (1988) A framework for comparing information engineering methods. *MIS Quarterly*, **June**, 202-220.

Henson, K. and Hughes, C. (1991) A two-dimensional approach to systems development. *Journal of Information Systems Management*, **Winter**, 35-43.

Hirschheim, R. and Newman, M. (1991) Symbolism and information systems development: myth, metaphor and magic. *Information Systems Research*, **2**, 1, 29-62.

Holloway, S. (1989) *Methodology Handbook for Information Managers*, Gower Technical, Aldershot.

Hough, D. (1993) Rapid delivery: an evolutionary approach for application development, *IBM Systems Journal*, **32**, 3, 397-419.

Humphrey, W., Snyder, T. and Willis, R. (1991) Software process improvement at Hughes Aircraft. *IEEE Software*, **July**, 11-23.

Jayaratna, N. (1994) *Understanding and Evaluating Methodologies*, McGraw-Hill, London

Jenkins, A., Naumann, J. and Wetherbe, J. (1984) Empirical investigation of systems development practices and results. *Information & Management*, **7**, 73-82.

Jones, M. and Walsham, G. (1992) The limits of the knowable: organizational and design knowledge in system development. In Kendall, K. *et al*., (eds), 195-213.

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

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, 1-17.

King, M. and Pardoe, J. (1985) *Program Design using JSP: A Practical Introduction*, Macmillan, London.

Land, F. and Hirschheim, R. (1983) Participative systems design: rationale, tools, techniques, *Journal of Applied Systems Analysis*, **10**, 91-107.

Langefors, B. (1973) *Theoretical Analysis of Information Systems*, Auerbach, Philadelphia.

Lewis, T. and Oman, P. (1990) The challenge of software development. *IEEE Software*, **November**, 9-12.

Longworth, G. (1985) *Designing Systems for Change*. NCC, Manchester.

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, 173-234.

Lyytinen, K. (1986) *Information Systems Development as Social Action*, University of Jyvasykla.

Lyytinen, K. (1987) Different perspectives on information systems: problems and solutions, *ACM Computing Surveys*,

McCracken, D. and Jackson, M. (1981) A minority dissenting position. In Agresti, W. (1986) *New Paradigms for Software Development*. IEEE Computer Society Press, Washington DC.

Maddison, R., Baker, G., Bhabuta, L., Fitzgerald, G., Hindle, K., Song, J.,Stokes, N. and Wood, J. (1984) Feature analysis of five information system methodologies. In Bemelmans, T. (ed.), 277-306.

Mumford, E. (1984) Participation- from Aristotle to today. In Bemelmans, T. (ed.), 95-104.

Naur, P. Randell, B. and Buxton, J. (1976) *Software Engineering: Concepts and Techniques*, Charter Publishers, New York.

Olerup, A. (1991) Design approaches: a comparative study of information system design and architectural design. *The Computer Journal*, **34**, 3, 215-224.

Olle, T., Hagelstein, J., Macdonald, I., Rolland, C., Sol, H., Van Assche, F. and Verrijn-Stuart, A. (2nd Edition) (1991) *Information Systems Methodologies: A Framework for Understanding*. Addison-Wesley.

Page-Jones, M. (1991). Structured methods are dead: long live structured methods. *American Programmer*, **November**, 31-37.

Palvia, P. and Nosek, J. (1993) A field examination of system life cycle techniques and methodologies, *Information & Management*, **25**, 73-84.

Plat, N., Katwijk, J. and Pronk, K. (1991) A case for structured analysis/formal design. In *VDM '91: Formal Software Development Methods*, Prehn, S. and Toetenel, W. (Eds), Vol. 1, Springer-Verlag.

Pressman, R. (1987) *Software Engineering: A Practitioner's Approach*, McGraw-Hill, New York.

Ramamoorthy, C., Garg, V. and Prakash, A. (1986) Programming in the large. *IEEE Transactions on Software Engineering*, **July**, 769-783.

Robey, D. and Markus, M. (1984) Rituals in information system design. *MIS Quarterly*, **March**, 5-15.

Rockart, J. and De Long, D. (1988) *Executive Support Systems*, Dow Jones-Irwin, Homewood, Illinois.

Sakthivel, S. (1992) Methodological requirements for information systems development. *Journal of Information Technology*, **7**, 141-148.

Schon, D. (1987) *Educating the Reflective Practitioner*, Jossey-Bass, San Franscisco.

Shemer, I. (1987) Systems analysis: a systematic analysis of a conceptual model. *Communications of the ACM*, **June**, 506-512.

Stage, J. (1991) The use of descriptions in the analysis and design of information systems, In Stamper *et al*. (1991) *Collaborative Work, Social Communications and Information Systems*, North Holland, 237-260.

Stamper, R. (1988) Analysing the cultural impact of a system, *International Journal of Information Management*, **8**, 3.

Stolterman, E. (1994) The 'transfer of rationality', acceptability, adaptability and transparency of methods, In Baets, W. (Ed), *Proceedings of Second European Conference on Information Systems*, Nijenrode University Press, Breukelen, 533-540.

Swartout, W. and Balzer, R. (1982) On the inevitable intertwining of specification and implementation. *Communications of the ACM*, **July**, 438-440.

Tagg, R. (1983) Too many methodologies, in Baker, G (Ed) *Data Analysis Update*, BCS Database Specialist Group.

Taggart, W. and Tharp, M. (1977) A survey of information requirements analysis techniques. *Computing Surveys*, **9**, 4, 273-290.

Takeuchi, T. and Nonaka, I. (1986) The new new product development game, *Harvard Business Review*, **January-February**.

Taylor, T, and Standish, T. (1982) Initial thoughts on rapid prototyping techniques, *ACM SIGSOFT Software Engineering Notes*, **7**, 5, 160-166.

Verity, J. (1987) The OOPS revolution. *Datamation*, May 1, 73-78.

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, 4-16.

Ward, P. (1992) The evolution of structured analysis: Part II--maturity and its problems. *American Programmer*, **5**, 4, 18-29.

Wasserman, A. (1981) In Chikofsky, E. (1989) How to lose productivity with productivity tools. *Proceedings of 3rd IFAC/IFIP Workshop*, Indiana, US, 1-4.

Wastell, D. and Newman, M. (1993) The behavioral dynamics of information systems development: a stress perspective, *Accounting, Management & Information Technology*, **3**, 2, 121-148.

Yeh, R. (1991) System development as a wicked problem. *International Journal of Software Engineering and Knowledge Engineering*, **1**, 2, 117-130.

Yourdon, E. (1991) Sayonara, once again, structured stuff. *American Programmer*, **4**, 8, 31-38.

Yourdon, E. and Constantine, L. (1977) *Structured Design*, Yourdon Press, New York.

Zelkowitz, M. (1988) Resource utilisation during software development. *Journal of Systems and Software*, **8**, 331-336.

Zolnowski, J. and Ting, P. (1982) An insider's survey of software development, *Proc. 6th Int'l Conf. Software Engineering*, Tokyo, Japan, 178-187.