Fitzgerald, B. (2000) Systems Development Methodologies: The Problem of Tenses, *Information Technology & People*, Vol. 13, No. 3, pp. 13-22.

SYSTEMS DEVELOPMENT METHODOLOGIES: THE PROBLEM OF TENSES

Brian Fitzgerald
Executive Systems Research Centre
University College Cork
Ireland
Email: bf@ucc.ie

Abstract

This paper presents two fundamental arguments. Firstly, it is proposed that most of the currently-available systems development methodologies are founded on concepts which emerged in the period from about 1967 to 1977. This argument is presented through the use of literature references. The second argument is that the profile of the development environment now faced in organisations is very different from that which prevailed in the period 1967 to 1977. This is illustrated through original empirical research carried out by the author which supports this argument, and by contrasting these findings with those of previous studies in the literature. It is therefore argued that there is a need to update 'tenses' by deriving new methodological canons more appropriate to the needs of the current development environment. Some suggestions for new methodological canons appropriate to the current development environment are provided.

Keywords: IS development methodologies, IS development methods and tools, Systems development techniques, IS development strategies.

1. Introduction

Most of the systems development methodologies in use today have their origins in a set of concepts that came to prominence in the 10-year period from about 1967 to 1977. Thus, overarching concepts such as the systems development lifecycle, prototyping, and user participation can be traced to this period. Fundamental design strategies such as functional decomposition, information hiding, and techniques such as data flow diagramming and entity relationship modelling also stem from this era. Even the origins of object-orientation can be traced to this period. This paper provides some empirical evidence which supports the contention that the profile of development is very different from that faced in the past when these methodologies were first promoted. It is therefore argued that a 'problem of tenses' (cf. Friedman, 1989) also exists in relation to systems development methodologies, viz., there is a need to move from the past imperfect to the future by deriving new methodological canons more appropriate to the needs of the current development environment.

The paper is structured as follows. In the next section, the foundational concepts underpinning most methodologies are identified and traced to the 'golden decade', 1967 to 1977. Following this, empirical evidence is provided which suggests that the profile of the organisational development environment faced currently is vastly different from that of the period 1967 to 1977. Finally, some new canons to guide modern systems development are proposed and conclusions drawn.

2. Foundational Concepts of Systems Development Methodologies

As already mentioned, most of the systems development methodologies in use today have their origins in a set of concepts that came to prominence in the 10-year period from about 1967 to 1977.[1] The following is a brief summary of the concepts involved and the relevant supporting literature:

The *systems development lifecycle* (SDLC) was first applied in relation to systems development in this era (Royce, 1970). It has been reckoned to be the basis for most methodologies (Davis *et al*., 1988; Orr, 1989), and it has been described as the "cornerstone" and a "hallmark of every development effort" (Ahituv *et al*., 1984). A lifecycle concept is common in many branches of engineering, and its introduction into the systems development area in this period coincided with the desire to establish an engineering approach to systems development (cf. Friedman, 1989; Kraft, 1977). In the early years of systems development, developers were typically scientific researchers with a strong mathematical or engineering background who developed their own programs to address the particular areas in which they were carrying out research. Given this type of environment, there was little need for direct management control or for any methodological support for development. Thus, these early programmers operated in an environment which Friedman (1989) terms as loose responsible autonomy with little management control or project management focus.

The move to the SDLC approach represented a shift towards tighter control of the development process (Friedman, 1989). However, the traditional lifecycle proved to be problematic, and a common response to such problems was the adoption of *prototyping* amongst many practising developers (cf. Bally *et al*., 1977). Indeed, prototyping has become a common feature in many of the development methodologies which are now being marketed (cf. e.g. Boehm, 1988, Downs *et al*., 1992). Some researchers have identified a construct which forms a superset relationship with development methodologies which they have labelled a *process model* (Lyytinen, 1987; Wynekoop & Russo, 1993). The two most common process models which underpin most current methodologies are suggested to be the SDLC and prototyping ones (Wynekoop & Russo, 1993).

The concept of *user participation* in the development process also has its origins in this period (Emery & Trist, 1969; Herbst, 1974). During the 1970's it was becoming increasingly recognised that the Taylorist assumptions which guided systems development were problematic, often resulting in systems which were rejected by the end-users (Cherns, 1976). Hence there emerged a focus on sociotechnical systems, which considered the joint optimization of both technical and social aspects of systems design.

The *structured approach* which is documented in an early form in Yourdon (1967), has since been reckoned to be the most widely used methodology in North America and Europe (Yourdon, 1991). Its principal concepts such as *information hiding*—the idea that each module should hide exactly one design decision and reveal as little as possible about its inner workings or the data it uses (Parnas, 1972), *functional decomposition*—the progressive subdivision of primary functions into sub-functions until some primitive level is reached, *cohesion*, and *coupling* all emerged in this era (De Marco, 1978; Stevens, Myers & Constantine,

[1] It should be noted that some of these concepts emerged in an earlier period also. For example, the sytems development lifecycle (cf. Canning, 1956) and object orientation (cf. Dyke & Kunz, 1989) may be traced to the mid-1950s. However, the argument in this paper is that they became prominent in the decade, 1967-1977.

1974). Furthermore, techniques which are invariably present in many methodologies, such as *entity relationship diagrams* (Chen, 1976), *data-flow diagrams* and *data dictionaries* (De Marco, 1978) are also clearly from this era.

Also, SSADM, the most widely used methodology in the UK and Ireland (Downs *et al*., 1992), has its antecedents in the MODUS methodology which was in use between 1965 and 1977. Another popular methodology, Jackson Structured Programming (Jackson, 1973), is founded on the principles of Bohm and Jacopini (1966) which proved that all programming and data structures could be represented by three primitive constructs.

Some researchers have identified *object-orientation* as the new paradigm for systems development (Thomann, 1994), and, indeed, OO has recently been a major influence in the methodology literature. Evidence of the growing interest in OO can be found in the Ovum estimate that the OO market will have reached $2 billion by 1996 (Topper, 1992). However, the newness of the concepts underpinning the OO paradigm is questionable. Its origins can be traced directly to the Simula programming language in use in Norway from 1967, and also before this, OO principles such as encapsulation were used in the design of the Minuteman missile in the late 1950s (Dyke and Kunz, 1989). In addition, it seems to be the case that many of the recent methodologies which are based on the OO approach are often just evolutionary outgrowths of earlier approaches (cf. Berard, 1995; Firesmith, 1993; Iivari, 1994; Monarchi & Puhr, 1992).

3. Altered Profile of the Prevailing Development Environment

The previous section provides some significant evidence to argue that many current methodologies are founded upon concepts derived between about 1967 and 1977. However, in order to argue that it is now time to 'update tenses' in relation to methodologies, it is necessary to demonstrate that there are profound differences between the development environment currently faced and that which prevailed when these methodologies were first promoted. This is addressed in this section by drawing on recent literature, including the results of various empirical studies carried out by the author (Fitzgerald, 1994, 1997, 1998) which illustrate these differences. These issues 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.

The accelerating pace of change characteristic of the business environment facing organisations today is a common theme in contemporary research. Rockart and De Long (1988) refer to the "faster metabolism of business today" which requires organisations to act more effectively in shorter time-frames. Baskerville et al. (1992) also discuss the relevance of the nature of the prevailing business environment to the systems development issue. They argue that most 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.

Researchers in the area of rapid 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 concomitant change to an accelerated development approach. The main thrust of rapid delivery is to produce frequent tangible results, that is, every few months some functional capability is delivered. This concept is also central to Gilb's (1988) incremental engineering approach.

Recent empirical research carried out by the author (cf. Fitzgerald, 1994, 1997, 1998) also provides evidence of a change in the prevailing development profile. Firstly, a large-scale postal survey, involving a total of 776 named individuals in different organisations who were likely to be directly involved or responsible for systems development, was conducted. The findings of the survey are documented in detail in Fitzgerald (1998). The data in Table 1 is drawn from this survey and is reproduced here to illustrate the profile of the current development environment.

	Mean
Development Profile:	
% systems development in-house	47
% systems development outsourced	13
% use/customisation of packages	40
Development Project Profile:	
No. of developers	3.47
Duration (in months)	5.72
Hardware platform	
Mainframe	20%
Mini	26%
PC	33%
Mixture/Other	21%

Table 1 Profile of Current Development Environment

An interesting feature of Table 1 is the high-level of package customisation (average of 40 percent), and also the level of outsourcing (average of 13 percent). Given these figures, it would appear that in-house development is no longer predominant in companies. In addition, the fact that typical development projects comprised about three developers for less than six months, seems to reflect a profile of small-scale, rapid development, which contrasts greatly with findings from earlier studies. For example, Taylor and Standish (1982) report project durations of up to 5 years.

The second phase of the empirical research involved in-depth field interviews with 16 experienced developers in eight organisations. The findings are reported in detail in Fitzgerald (1997), and the salient aspects reported here. The emphasis on shortened development project duration was borne out in the interviews. Several interviewees expressed the view that development projects of long duration were not tolerable as the underlying business could have changed dramatically in the interim.

Further evidence of an altered development profile may perhaps be gained from the survey finding that only 20 per cent of development was on a mainframe platform, with the most common platform for development being the PC one (33 per cent), again not one typically associated with large-scale development projects in the past. This contrasts with the findings of an earlier study by Sumner and Sitek (1986) which reported 57 per cent of development on a mainframe platform, with only 2 per cent on a PC platform. Granted, PCs were in a state of relative infancy at that time. However, it is clear that the PC platform is becoming a more common one.

The survey also revealed that 60 percent of respondents did not use a development methodology, and that only 14 percent claimed to use a formalised commerical methodology[2] (see Table 2). Previous studies of systems development in practice have reported usage rates for methodologies of 87 percent (Jenkins *et* al., 1984) and 62 percent (Necco *et* al., 1987). The predominant reason for non-use cited by respondents was that currently available methodologies did not suit the profile of the development prevailing in the organisations studied. In the field interview research (Fitzgerald, 1997), it emerged that those organisations who were using methodologies had tailored them very precisely to meet the needs of their particular development environment. However, they were almost always framed at a higher level of granularity, in that they provided broad guidelines rather than a large number of low-level steps to be carried out in a prescribed sequence. Similar methodology adaptation has been reported in other studies (e.g. Russo *et al.*, 1995). Thus, a methodology derived in one software house emphasised issues to do with testing, version control, and telephone support, as these are critical issues in their business sector. Similarly, a large bureaucratic government department had constructed a methodology which focused on tender and request for proposal (RFP) issues, as these were critical given the level of outsourcing in the organisation. Also, a major bank had emphasised those methodology phases which had to do with strategic planning, as they considered it important that all systems development projects would be underpinned by a business case.

Organizations not using any methodology	60%
Organizations using a formalised commercial methodology	14%
Organizations using internal methodology based on a commercial one	12%
Organizations using internal methodology not based on a commercial one	14%

Table 2 Methodology Usage

Table 3 analyses the survey findings in relation to the average percentage of time

[2] The term 'formalised' is used here to denote formally-defined, brand-named or publshed 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.

spent on various development activities by methodology users and non-users. Although the findings do not reveal significant differences, it appears that there is a slightly more even distribution of time when a methodology is being followed. Thus, slightly more time appears to be allocated to the analysis and design activities. However, the differences really are very marginal, which serves to question the extent to which methodologies play a significant role in ensuring that specific activities take place, as has been suggested (Ahituv *et al*., 1984; Baskerville *et al*., 1992). Several researchers have recommended that more time be allocated to early development phases (e.g. Couger *et al*., 1982; Necco *et al*., 1987), with McKeen (1983) reporting a relationship between the amount of time allocated to the analysis phase and greater user satisfaction. However, the results of this study would suggest that methodologies *per se* do not ensure that this occurs, since there is so little difference in the proportion of time allocated to these activities by methodology users and non-users.

	Avg. percentage of dev. time allocated by those using a methodology	**Avg. percentage of dev. time allocated by those not using a methodology**
Activity	**%**	**%**
Systems Planning	10	10
Systems Analysis	17	14
Systems Design	15	12
Programming	28	31
Testing	17	17
Installation	8	10
Evaluation	3	4
Other	2	1

Table 3 Average Percentage of Development Time Allocated to Development Activities

Table 4 analyses the percentage of respondents citing the use of various development tools and techniques. As can be seen from the table, the most popular tools and techniques are prototyping, data flow diagramming, data dictionaries. and entity relationship models. The interesting finding here is that those using methodologies use **all** of these tools and techniques to a far greater extent than those not using methodologies. This lends support to the argument that methodologies provide a suitable framework to co-ordinate the purposeful application of tools and techniques (cf. Bantleman & Jones, 1984; Holloway, 1989). It has been suggested that methodologies generally assume some underlying philosophy and fundamental principles in relation to the phases and activities of systems development (Jayaratna, 1994). However, given that there is no real difference in emphasis on particular development phases, the extent to which methodology users assimilate the deeper underpinning principles of methodologies is questionable.

	Organizations using a methodology	**Organizations not using a methodology**
Tools/Techniques	%	%
Joint Application Design (JAD)	31	20
Prototyping	75	57
Data Flow Diagramming	71	37
Entity Relationship Modelling	63	19
Entity Life Histories	19	6
Flow Charting	55	35
Data Dictionary	74	34
Process Mini-Specifications	40	25
Structured Walkthrough	48	23
Other	9	5

Table 4 Percentage of Respondents Using Various Development Techniques

4. New Canons for Systems Development

The Chinese leader, Mao Tse Tung, is credited with the observation that revolutions are needed every 20 years or so as human beings tend to settle into numbing routines which actually prevent advances taking place (Patton, 1990). Certainly, there is support for the view that dramatic upheavals or paradigm shifts are necessary for scientific progress throughout the philosophy of science (Kuhn, 1970). Locating the argument in the software development area, Cox (1990) draws parallels with the craft of the gunsmith to argue cogently that there are usually limits to what can be achieved as one optimises any given process. In the traditional gunsmithing craft, gunsmiths were striving to make the process as efficient as possible. However, with the new technology of production afforded by the Industrial Revolution, a new process for gun-making emerged with much higher productivity levels. This new process represented a step change and there was no way the traditional gunsmith craftsmen could bridge the gap to compete. In calling for an "Industrial Revolution" in how software is developed, Cox argues that there is a need for radically new approaches which are more appropriate to the needs of the environment as it evolves. Also, as already mentioned, in the specific area of information systems development, Friedman (1989) has identified the "problem of tenses" whereby common-place practice often lags best practice by quite some time. Thus, there is much to be learned from best practice situations.

As illustrated in the previous section, the profile of the development environment is vastly different from that which prevailed when many of the currently available commercial methodologies were first proposed some 25 to 30 year ago. Thus, there is a need to 'update the clock' by deriving sensible methodological canons more suited to the needs of the current development climate. The following are offered as issues that should be taken account of in the derivation of new development methodologies:

1. As can be seen from the empirical evidence presented above, both

integration and customisation of packages and outsourcing are quite prevalent in today's environment, yet few methodologies cater for these phenomena. However, this mode of configuration development with higher-level building blocks facilitates initiatives such as timeboxing, frequent tangible returns etc., integral to rapid application development (RAD) approaches.

2. Business systems development is often algorithmically simple. Thus, methodologies which may be strong in the area of real-time engineering systems design may not be as appropriate for business systems development. This is possibly even more relevant given the emergence of OO methodologies as many of the latter have been derived from experiences with real-time applications, as object persistence—a fundamental feature of business applications where data storage is a key issue—is not necessary.

3. Neither the top-down SDLC approach which implies the elicitation and freezing of requirements in advance, nor the bottom-up prototyping and iterative development approach which views requirements as emerging as the development process takes place, are sufficient in isolation. Rather, a mix of strategies may be more appropriate. Thus, in an application of the Pareto principle, some system functions may be developed in a top-down fashion using the traditional lifecycle, perhaps even with an exaggerated absence of user involvement. This may represent about 60 to 70 percent of requirements, and the remaining system functions may be developed in a bottom-up fashion using a prototyping approach.

4. While methodologies can introduce rigour to the development process, productivity may necessarily suffer, and this trade-off is not tolerable given the current organisational climate. Thus, there may be a sense in which 'good enough' systems can be developed in an appropriate time-scale, rather than striving towards delivering optimum solutions in an unreasonable time-scale.

5. The development process can be over-intellectualised, and in some circumstances a methodology may prescribe an overly-complex approach whereas a simpler one may be more appropriate given the nature of development in many organisations. In these circumstances, the methodology becomes something of a 'mother hen'—overly cautious and conservative, thus leading to an inflexible and cumbersome development process.

6. A degree of responsible autonomy prevails in relation to the development process in many organisations, with much left to the discretion of developers. Methodologies cannot be inflicted upon developers; thus, departures from the prescribed steps of methodologies are common in practice; however, these departures are conscious and deliberate rather than arbitrary. This could be viewed as evidence, perhaps, of a maturity on the part of developers in relation to methodology usage.

7. Also, the level of granularity of methodological steps needs to be different, with the methodology specifying at a higher level of abstraction the outcome to be achieved, with the precise manner in which it is achieved left to the discretion of the developer. Thus, rather than prescribing the minutiae of steps which developers are expected to follow, the focus should be on higher-level goals and deliverables, and the precise manner in which these are actually achieved should be left to the developer.

8. The prevailing business climate requires that organisations act more effectively in shorter time-frames. There is a need for more rapid systems delivery than that which is currently being achieved with the monolithic development approaches inherent in traditional system development methodologies. In fact, the latter may impose a considerable degree of inertia on the development process. Also, given the strong arguments in favour of informating the workplace and empowering employees, the expectation implicit in many methodologies, that developers will plod robotically through standardised checklists, is not valid.

5. Conclusion

It is undoubtedly the case that practice should inform theory. This view is supported by the fact that practice has often preceded theory in the field. The systems development lifecycle and prototyping were both areas in which practice led theory (Agresti, 1986, Friedman, 1989), as were programming style, compiler writing, and user-interface design (Glass, 1991). Also, the Sage missile-defense system and the SABRE airline reservation system, developed in the 1950s and 1960s, were both examples of sophisticated interactive systems which far exceeded the maturity of the theory at the time (Shaw, 1990). Thus, it may be the case that empirical explorations of systems development may reveal the types of practice upon which the new generation of system development methodologies should be based. This has certainly been the case in the research reported here.

The importance of successful systems development persists as an issue of central significance and concern in the IS field, especially in light of the well-documented problems associated with system development. However, as has been argued in this paper, many methodologies in use today are derived from practices and concepts relevant to a very different organisational environment, and there is a need to reconsider their role in view of changes in organisational forms and work practices, and the increasingly-complex applications that need to be developed to suit the complexity of the current organisational environment. Given the significant 'push' factor that this environment represents, there is an urgent need to leverage new developments, both technological and in organisational work practices, which enable new development approaches more appropriate to this organisational climate. Further research is therefore needed which would investigate the true nature of the current systems development environment in real organisational situations, and on real development projects. Practitioners have in many cases assimilated good practices and techniques and may be rejecting methodologies for pragmatic reasons rather than due to ignorance as has been suggested (cf. Ward, 1991; Yourdon, 1991). The next generation of methodologies require a new set of foundational concepts. However, these should be drawn in large measure from the ‘best practice’ development situations which prevail at present.

References

Agresti, W. (1986) *New Paradigms for Software Development*. IEEE Computer Society Press, Washington DC.

Ahituv, N., Hadass, M. and Neumann, S. (1984) A flexible approach to information system development. *MIS Quarterly*, **8**, 2, 69-78.

Bally, M., Britton, J. and Wagner, K. (1977) A prototyping approach to information systems design and management, *Information & Management*, **1**, 1, 21-26.

Bantleman, J. and Jones, A. (1984) Systems analysis methodologies: a research project. In Bemelmans, T. (ed.) *Beyond Productivity: Information Systems Development for Organizational Effectiveness*, Elsevier, North Holland Press, 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., DeGross, J. and Lyytinen, K. (eds.) *The Impact of Computer Supported Technologies on Information Systems Development*, Elsevier., North Holland Press, 241-269.

Berard, E. (1995) Object-oriented methodologies, Online document available at http://www.toa.com.

Boehm, B. (1988) A spiral model of software development and maintenance. *IEEE Computer*, **21**, 5, 61-72.

Bohm, C. and Jacopini, G. (1966). Flow diagrams, Turing machines and languages with only two formation rules. *Communications of the ACM*, **May**, 366-371.

Canning, R. (1956). in Agresti, W. (1986) New Paradigms for Software Development. IEEE Computer Society Press, Washington DC.

Chen, P. (1976) The entity relationship model—towards a unified view of data, *ACM Transactions on Database Systems*, **1**, 1, 9-36.

Cherns, A. (1976) The principles of sociotechnical design. *Human Relations*, 29, 8, 783-792.

Couger, J., Colter, M. & Knapp, R. (1982) *Advanced System Development Feasibility Techniques*, Wiley & Sons, New York.

Cox, B. (1990) Planning the software industrial revolution, IEEE Software, November, pp.25-33.

Davis, A., Bersoff, E. and Comer, E. (1988) A strategy for comparing alternative software development life cycle models. *IEEE Transactions on Software Engineering*, **October**, 1453-1460.

DeMarco, T. (1978) *Structured Analysis and System Specification*, Yourdon Press, New Jersey.

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

Dyke, R. and Kunz, J. (1989) Object-oriented programming. *IBM Systems Journal*, **28**, 3.

Emery, F.E. and Trist, E. (1969) *Form and Content in Industrial Democracy*. Tavistock.

Firesmith, D. (1993) *Object-Oriented Requirements Analysis and Logical Design*, Wiley and Sons, New York.

Fitzgerald, B. (1994) Whither Systems Development: Time to Move the Lamppost, in Lissoni, C. *et* al (Eds) *Proceedings of Second Conference on Information Systems Methodologies*, BCS Publications, Swindon, pp. 371-380.

Fitzgerald, B. (1997) The Use of Systems Development Methodologies in Practice: A Field Study, *The Information Systems Journal*, Vol. 7, pp. 201-212.

Fitzgerald, B. (1998) An Empirical Investigation into the Adoption of ISD Methodologies, *Information & Management*, Vol. 21, No. 6.

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

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

Gilb, T. (1988) *Principles of Software Engineering Management*, Addison Wesley, UK.

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

Herbst, P. (1974) *Socio-technical Design: strategies in multi-disciplinary research.* Tavistock.

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

Iivari, J. (1994) Object-oriented information systems analysis: a comparison of six OOA methods. In Verrijn-Stuart and Olle (Eds.) *Methods and Associated Tools for the IS life cycle*, IFIP North Holland, 85-110.

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.

Kraft, P. (1977) *Programmers and Managers: The Routinization of Computer Programming in the United States*, Springer-Verlag, New York.

Kuhn, T. (1970) The Structure of Scientific Revolutions, University of Chicago Press, Chicago.

Jackson, M. (1973) *Principles of Program Design*, Academic Press, London. p.407-413.

Lyytinen, K. (1987) A taxonomic perspective on information systems development, in Boland, R and Hirschheim, R. (eds.) *Critical Issues in Information Systems Research*, John Wiley and Sons, Chichester.

McKeen, J. (1983) Successful development strategies for business application systems, *MIS Quarterly*, **7**, 3, 47-66.

Monarchi, D. and Puhr, G. (1992) A research typology for OO analysis and design. *Communications of the ACM*, **35**, 9, 35-47.

Necco, C., Gordon, C. and Tsai, N. (1987) Systems analysis and design: current practices. *MIS Quarterly*, **December**, 1987.

Orr, K. (1989) Methodology: the experts speak. *BYTE*, **April**, 221-233.

Parnas, D. (1972) On the criteria to be used in decomposing systems into modules. *Communications of the ACM*, **15**, 12, 1053-1058.

Patton, M. (1990) *Qualitative Evaluation and Research Methods* (2nd Edition), Sage Publications, London.

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

Royce, W. (1970) Managing the development of large software systems. *Proceedings of IEEE Wescon.*

Russo, N., Wynekoop, J, and Walz, D. (1995) The use and adaptation of systems development methodologies, in Khosrowpour, M. (Ed), *Managing Information & Communications in a Changing Global Environment*, Idea Group Publishing, PA.

Shaw, M. (1990) Prospects for an engineering discipline of software. *IEEE Software*, **November**, 15-24.

Stevens, W., Myers, G. and Constantine, L. (1974) Structured design. *IBM Systems Journal*, **13**, 2, 115-139.

Sumner, M. and Sitek, J. (1986) Are structured methods for systems analysis and design being used? *Journal of Systems Management*, **June**, 18-23.

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

Thomann, J. (1994) Data modelling in an OO world. *American Programmer*, **7**, 10, 44-53.

Topper, A. (1992) Building a case for object-oriented development, *American Programmer*, **5**, 8, 36-47.

Ward, P. (1991) The evolution of structured analysis: Part I--the early years. *American Programmer*, **4**, 11, 4-16.

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, 181-190.

Yourdon, E. (1967) The emergence of structured analysis, *Computer Decisions*, **8**, 4, 58-69.

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