

An Empirical Study of System Development Method Tailoring in Practice

Brian Fitzgerald
University College Cork,
Ireland
bf@ucc.ie

Nancy Russo
Northern Illinois University,
USA
M10NLR1@wpo.cso.niu.edu

Tom O’Kane
Motorola, Cork
Ireland
qcor067@email.mot.com

Abstract Little research has been conducted to date on the specific topic of the tailoring of systems development methods. Two related research areas—contingency factors research and method engineering—have exhibited a primarily deductive research focus. In contrast, this paper presents an inductive study into method tailoring in practice within the Motorola organisation. The findings illustrate the sophisticated multi-level tailoring process at industry, organisational and project level. The multi-level tailoring process depicted here overcomes the problem of trying to comprehensively tailor a method in a development environment in which time is not available for a lengthy tailoring process on each project. The paper builds on both the contingency factors and method engineering streams, and also contains useful practical guidelines for practitioners.

I. INTRODUCTION

Information systems development (ISD) continues to be an issue of central significance and concern in the IS field. Given the well-documented problems in ISD, it is not surprising that ISD methods have been the focus of much research. It is reckoned that more than a thousand ISD methods exist [1], and while this figure may be excessive, it is certainly the case that an abundance of ISD methods are available. These methods are generally touted by their advocates as universally applicable in all situations; yet, some methods differ fundamentally—Soft Systems Method [2] and Information Engineering [3], for example, while other methods differ in relatively trivial aspects—the many variants of the structured approach, for example (e.g., [4] [5] [6]). Also, despite the fact that many of these methods are scrupulously documented, prescribing in minute detail the exact sequence of steps to be followed, several researchers have reported that, in practice, developers rarely follow the sequence of steps as prescribed in the method [7] [8] [9]. As a consequence, newer versions of existing methods now almost routinely recommend some contingent tailoring [10]. Notwithstanding this, there is very little by way of practical guidance to inform developers as to what steps to modify or omit. This paper is an attempt to contribute to an increased understanding of this issue. The approach taken is slightly different from the conventional one, in which tenets for method tailoring might be derived deductively from some *a priori* stated set of axioms. Rather, in this paper, we focus in-depth on method tailoring in a very formalised development environment in a particular high-profile software

development organisation, the Motorola organisation in Cork, Ireland.¹ From this case, we use an inductive strategy to draw lessons on tailoring which could be applicable to other organisations.

The remainder of the paper is laid out as follows: The next section briefly discusses the literature relevant to method tailoring, including contingency research and method engineering. Following this the research approach adopted for this study is discussed. Then the case study is presented, and following this the method in use in Motorola, Cork and the manner in which it is tailored is discussed. Finally, some conclusions and lessons from the study are presented.

II. PREVIOUS RESEARCH ON METHOD TAILORING

Little research appears to have been conducted on the specific topic of method tailoring. However, two closely-related areas are contingency factors research and method engineering. These are briefly summarised here and their relationship to method tailoring discussed.

A. Contingency Factors Research

Research on contingency factors in ISD methods has been a long and continuing research stream in IS (e.g. [11] [12] [13] [14] [15] [16] [17]). Contingency research is typically premised on the notion that specific features of the development context are mapped to the selection of an appropriate ISD method from a portfolio of methods.

Davis [13] represents an early and widely-cited contribution. He considered the area of information requirements determination and evaluated alternative strategies for this. He proposes a contingency approach based on an assessment of different levels of uncertainty through which an appropriate strategy is selected from among several available alternatives. Thus, in Davis’ model, an organisation would be expected to have a range of methods available to developers, who would presumably be fully *au fait* with each method, and the most appropriate one would be chosen depending on the contingencies of the situation.

¹ Given the size of the Motorola organisation, it is appropriate to more precisely identify the software organisation as the Cellular Switching and Operations and Maintenance Centre departments of Motorola’s Network Solutions Sector (NSS) group in Cork, Ireland.

In a similar fashion, Gremillion and Pyburn [14] also recommend a contingency approach proposing that development projects be evaluated according to the criteria of commonality, impact and structure before deciding on a development approach, which could range from the traditional approach, prototyping, or an application package solution.

Iivari [15] argues for a pragmatic contingency approach and provides a framework which illustrates how newer methods have been incorporating a contingency approach. However, his approach differs from that of Davis [13] in that he argues for built-in contingency as a feature of the method itself. Thus, he is not arguing for a repertoire of methods; rather, the encompassing framework of the method is expected to cover all situations.

Avison and Wood-Harper [11] review various ISD methods and conclude that none can be appropriate in all situations. Adhering more to the Iivari model than that of Davis, they propose their favoured method, Multiview, as a contingency framework which can incorporate the tools and techniques relevant to a particular context. They also acknowledge the difficulties that can arise in using a contingency approach.

A similar approach which has been proposed by Benyon and Skidmore [12] is to create a single “tool kit” which combines the essential features of five methods which they suggest are complementary. The methods include the soft systems method (SSM) [2], Structured Systems Analysis and Design [4], the traditional approach [18], the data-centered approach [19], and the participative approach [20]. The expectation is that developers would be skilled enough to choose the appropriate method or tool depending on the situation. While the five different approaches are indeed complementary, ranging from ‘soft’ to ‘hard’, and from process-driven to data-driven, the tool-kit approach has been criticised as being inadequate [21].

Shomenta *et al.*, [17] developed their Application Approach Worksheet as a tool which may be used to help developers and users make the transition from using one development approach to using a variety of tools and approaches. Based on various characteristics of an application, the worksheet guides the selection of approach, either traditional, end-user computing via 4GL on the mainframe, or end-user development using a microcomputer-based package.

Naumann and Palvia [16] describe a quantitative model for selecting a system development method. The model, which uses a Delphi method to achieve consensus regarding relative importance of functions, is suggested for selecting the organizational standard method, not necessarily on a project-by-project basis.

The contingency approach in general has been the subject of criticism by Kumar and Welke [22]. They contend that existing methods do not adequately cover all contingencies, and further, that the cost of sourcing and training for each method that is required by the contingencies of development

would be prohibitive. This situation would be further exacerbated by the rapid and fundamental changes in the prevailing development environment in organisations. In addition to this, the contingency literature contains little by way of practical guidelines as to how methods or tools may be mapped to development contingencies. The tool-kit and Multiview approaches provide no guarantor of either the adequacy or necessity of the particular tools or stages included in the method. The solution proposed by Kumar and Welke is that of method engineering. This stream of research is considered next.

B. Method Engineering Research

Harmesen *et al.* [23] trace the origins of the concept of method engineering to mechanical engineering in the 1930s [24]. They acknowledge the advantage of ISD methods in their provision of a disciplined standard for development, but recognise that flexibility is necessary in order that methods be ‘tuned’ to meet specific project needs. Their objective is the “harmonisation of methods” through the provision of a strategy for constructing situational methods out of existing proven method fragments. To operationalise this they suggest a method base repository which contains suitable method fragments. This repository is part of a computer-aided method engineering (CAME) tool, and they identify two new personnel roles—a method engineer who configures the project-specific situational method, and a method administrator who assumes responsibility for the method base. Harmesen *et al.* provide quite a detailed description of the application of situational method engineering; however, the case they use to illustrate their approach is drawn from a literature example [25] rather than a real-life organisation.

Kumar and Welke [22] argue that design methods must themselves be designed, and propose a recursive step whereby the method would itself be designed using a meta-method. They suggest that methods be represented as discrete pre-defined and pretested components. These can then be drawn upon to construct a method quickly, cheaply and efficiently. They identify and discuss a number of strategies which need to be accommodated in such a meta-method. The component base should be based on stakeholder values and supported by automated computer-based support as well as the organizational structure. The automated support should allow for seamless integration of the modular components. Tutorials and training aids should also be part of the package.

This recursivity feature of method engineering is similar to an older argument about programming languages [26]. In research on method engineering, Tolvanen proposes the use of metamodeling which is defined as “a modeling process which takes place one level of abstraction and logic higher than the standard modeling process”. The metamodel captures information about the concepts, representation forms, and uses of a method. Metamodels are divided into two categories, meta-data models which describe the static aspects of a method, and meta-process models which describe

the dynamics of a method. The author suggests that metamodeling provides advantages in terms of representing, systematising and comparing methods. He argues that many current methods are defined quite vaguely, and suggest that method users, already familiar with modeling techniques, would be very competent at analysing methods which have been metamodelled.

One marked feature of both the contingency and method engineering research is that they are largely deductive in nature as they employ theoretical and conceptual arguments to support how methods should be tailored or constructed. Very little is available in terms of practical applications of these ideas in real life. This gap is addressed specifically in this study which is grounded on method tailoring in practice.

III. THE RESEARCH APPROACH

As already mentioned, this study was not concerned with deductively testing some *a priori* defined hypotheses. Rather, the emphasis was on inductively deriving some lessons on method tailoring from a practically-grounded example of method tailoring. Researchers have long called for research on ISD in real-life organisational situations [27] [28] [29] [30] [31] and they continue to do so [32] [33]. The dearth of such real-life studies of ISD has been noted by several researchers [32] [30] [34]. There have been calls for a "clearer understanding of the realities of systems development" [35]. As McLean [36] aptly put it: "the proper place to study elephants is the jungle, not the zoo". More research is therefore needed into the actual practice of ISD in organisations, justifiable even solely on the basis that practice has often preceded theory in the field. Few computing theorists are former practitioners; yet, in the early stages of a discipline, theory can best progress by examining good practice [37]. Also, given the wide gap between the best and average practice in the field (cf. [38] [39] [37]), it is important to discover the essentially good practices of capable systems developers, so that these can be transferred to other developers.

A. Research Objective

In the specific case of method tailoring the absence of real-life studies is even more stark. The objective of this research was therefore to investigate the nature of method tailoring in practice. The particular site selected for study allowed us to investigate method tailoring in a large-scale formalised development environment. The strategy adopted by Motorola, Cork contains many lessons which can be of benefit to both academic researchers and practitioners alike.

B. Matching the Research Objective with Research Method and Data Capture Techniques

Given that the method tailoring area is one in which much research of an exploratory and descriptive nature is needed,

any research method chosen should reflect this. Marshall and Rossman [40] consider this issue in great detail, and propose a framework for matching research purpose with research methods and data capture techniques. In the case of research which has a descriptive and exploratory focus, a combination of case study and in-depth interviewing is deemed appropriate according to their framework.

1) *The Case Study Method*: The case study is not viewed in a similar fashion by all researchers (cf. [41]). However, according to one of the more common interpretations, it describes a single situation, and usually involves the collection of a large amount of qualitative information (cf. [42] [43] [44]). Case studies can be very valuable in generating an understanding of the reality of a particular situation, and can provide a good basis for discussion. There is no attempt at experimental design nor any control of variables. As much data as possible is gathered on the presumption that it might prove useful, and also because it is difficult to go back for more information later. However, since the information collected is often specific to the particular situation at a particular point in time, results may not be generalisable. Scott [45] describes the central problem with the case study method very well:

The sustained researcher who burrows deeper and deeper into a single situation is faced with the danger of emerging so impressed with the complexity and uniqueness of 'the one dear case' that he may have difficulty in thinking abstractly about his materials or in attempting to generalise from them. Notwithstanding this limitation, the case study was chosen as the research method for this study, as its advantage in providing 'thick description' was seen as outweighing its limitations.

2) *In-depth Personal Interviews*: The purpose of the personal interview is to encourage the interviewee to relate experiences and attitudes relevant to the research problem [46]. It is a very flexible technique in that the interviewer can probe any interesting details that emerge during the interview, and concentrate in detail on particular aspects.

It should be noted that a *reflexive* approach was deliberately allowed in the interview phase. This has been identified as important in exploratory research [47] as it allows for refocusing as the research progresses, in that responses to certain questions can stimulate new awareness and interest in particular issues which may then require additional probing. Eisenhardt [48] also recommends such a strategy, labelling it "controlled opportunism".

In this study, a series of formal and informal interviews were conducted over a two-year period with the manager responsible for the ISD process at Motorola, Cork. Interviews were generally of a one to two-hour duration. The more formal interviews were taped and transcribed. Informal interviews were used to clarify and refine issues as they emerged. This manager subsequently became a co-author of the paper. Thus, the findings are further strengthened through

the direct validation of those responsible for the process being studied.

A number of problems have been identified in relation to the use of interviewing as a research technique. A frequently-cited problem is that of researcher bias, that is, the researcher may have expectations as to what the research is going to uncover and may ask questions that elicit the answers he or she wants to hear. This may be subconscious, but the way questions are phrased may lead to particular answers being given. The problem of researcher bias is further compounded by another associated problem, demand characteristics. This refers to the phenomenon whereby subjects give answers that they think the researcher wants. Critics of the interviewing technique suggest that the researcher "acts like a sieve which selectively collects and analyses non representative data" [49]. However, almost all research methods are 'guilty' of bias to varying degrees. In the interviews, open-ended questions were used as often as possible to allow more freedom for answers. Also, as the primary source of information became a co-author of the paper, the correctness of the researchers' interpretation was less of an issue than in the traditional model whereby exclusively-external authors interpret the research findings.

IV. THE CASE STUDY: MOTOROLA, CORK

A. *The ISD Environment at Motorola, Cork*

Motorola is a major systems provider in the mobile telecommunications sector. These systems are very large and expensive switching and communications infrastructure systems, and Motorola have over 300 engineers working on systems development in Cork. Clients are typically very large telecommunications providers who purchase Motorola systems to support their mobile phone networks. The nature of the environment in which these systems operate is one in which users (individuals who make telephone calls) take the underlying system completely for granted and expect total reliability. Given the fact that there are several very significant and reputable competitors in the market-place, the reliability of the Motorola systems is very important. Also, the telecommunications technology area is one that is constantly evolving, with new products and services continually on offer. Thus, systems are constantly being adapted to incorporate interfaces to these new developments. The systems themselves are developed using common languages such as C and C++. Technical personnel tend to have a background in engineering or computer science. Large numbers of them work on each development project and the development environment is very formalised. There is clear differentiation between the phases of design, implementation and testing. In the case of the latter, special test laboratory facilities are available for rigorously testing each system function. The development process is also very formalised. The organisation has explicitly documented their fundamental software process, the Cork Organisational Standard Software

Process (OSSP). This is tailored precisely to the development process for each project and is then followed rigorously on all projects. New employees are made aware of the Cork OSSP via induction training sessions. It is evolving as the company follows their program for continuous process improvement, which will ultimately lead to an improved rating on the Capability Maturity Model (CMM) [50]. Satisfying the concepts of the CMM is very important to Motorola, and a specialist group—the Process Engineering Group—exists within the Cork facility to ensure that the CMM criteria are complied with. A large amount of metric data on the development process is collected and analysed, and this information is later displayed on notice boards for the attention of developers.

Given the nature of the environment in which the Motorola systems operate, and the competitive nature of the marketplace, it is vital that errors and downtime are kept to a minimum. When errors do occur in systems, a very precise process for handling the situation is mandated. All fixes undergo rigorous testing before they are released to the customer.

V. METHOD TAILORING AT MOTOROLA

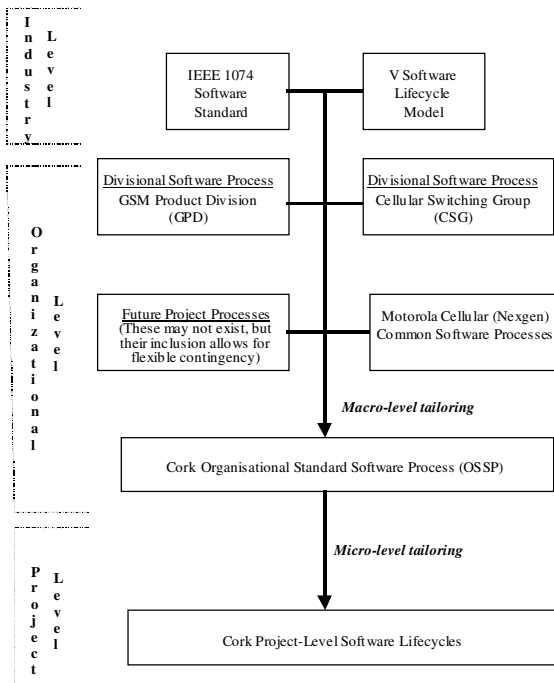
The software development process at Motorola involves a number of discrete elements (see Figure 1). These elements, their interrelationships, and the tailoring process are discussed next.

The components in Figure 1 may be categorised into three different levels, viz., the Industry level, the Organisational level, and the Project level. At what we have termed the Industry level, the elements or components are available more or less universally to any organisation developing software in that they are part of the public domain. Here, the two basic elements on which Motorola, Cork ground their development method are the IEEE 1074 software standard [51] and the V software lifecycle model (V-SLCM) (cf. [52]).

The IEEE 1074 standard is a very detailed one which prescribes a set of activities that are deemed "mandatory for the development and maintenance of software" [51 p. 1]. It comprises six high-level stages, seventeen process steps and sixty-five activities within these process steps. A detailed description of the standard is beyond the scope of this paper; thus, an overview is presented in Table I.

Motorola perceive a number of significant benefits in adopting the IEEE 1074 standard. Firstly, it represents an internationally-recognised standard for development which is evolving, but in a controlled and rigorous manner. Also, the standard is complementary to the CMM, which is very important in Motorola as a means of assessing the maturity of their development process and also as a mechanism to introduce improvements to that process.

Fig. 1: Tailoring the Software Process at Motorola



model (SLCM) be chosen, to which the activities can be mapped. Also, the IEEE 1074 standard states that it merely prescribes the *processes* for the lifecycle. The *products* of the lifecycle in terms of specific documents and deliverables must subsequently be mapped to the method. Thus, tailoring is very much an inherent requirement of the IEEE 1074 standard.

A software lifecycle model is a time-ordered sequence of activities for development. A number of such models exist, including the Waterfall [53], the Spiral [54], the Sashimi [55]), and the V-model [52]). The latter has been chosen by Motorola to complement the IEEE 1074 standard. However, by constructing their ISD method from discrete components, Motorola could introduce an alternate lifecycle model if they wished; indeed, the Spiral model is currently being investigated by Motorola.

At the Organisational level, a number of software processes exist which are specific to the various parent divisions within Motorola and naturally they influence the Cork process. These divisions include the GPD (GSM Product Division) and the CSG (Cellular Switching Group), which are both US-based. Each of these divisions has configured their software process to suit the exigencies of their particular development environment. For example, sub-contractor management is relevant to the GPD division but not to the other. Also, Motorola found that some of their common software processes were not covered in sufficient depth by the IEEE 1074 standard, testing and software maintenance issues being two examples. Thus, these needed to be factored into the organisational development process.

Finally, at the organisational level, it is recognised that some development projects in the future might require processes which are not accommodated by the current method. One possible example might be that a customer would seek some intermediate delivery of a product after design and prior to system testing. This would require a change to the existing processes. Thus, the existence of the Future Project Processes component ensures flexibility to cater for the contingencies that may arise in future development scenarios.

Based on these considerations, the overall Cork Organisational Standard Software Process (OSSP) is constructed. As can be seen from the discussion above, the process is already characterised by a good deal of tailoring. However, the tailoring is at a macro-level in that the specifics of the individual projects have not yet been factored in. At the organisational level, the main emphasis is on creating a trusted, rigorous and reliable software process which has already absorbed the sequencing aspects of a software lifecycle model (in this case, the V model). Also, the CMM key process areas (KPA)s² are explicitly factored into the method at this level.

TABLE I
OVERVIEW OF IEEE 1074 STANDARD

STAGE	PROCESS
Software Lifecycle Model	Software Lifecycle Model
Project Management	Project Initiation Project Monitoring and Control Software Quality Management
Pre-Development	Concept Exploration System Allocation
Development	Requirements Design Implementation
Post-Development	Installation Operation and Support Maintenance Retirement
Integral Processes	Verification and Validation Software Configuration Management Documentation Development Training

While the IEEE 1074 standard is detailed and comprehensive, it is recognised that it will need to be tailored in context. The standard explicitly specifies that an actual software lifecycle

² A full discussion of the CMM is beyond the scope of this paper. Briefly summarising however, the CMM specifies 18 key process areas (KPA)s which are central to a mature software process. These are requirements

The OSSP is reasonably stable although it is expected to evolve, and, indeed, the capability to evolve is built into the model. It represents the general process that each project is expected to follow, being the operational definition of the fundamental process elements and their inter-relationships. This strategy overcomes a problem identified with both method engineering and contingency approaches, namely, that organisations in practice clearly cannot afford to wait while a lengthy tailoring process takes place. In the Motorola case, much of the broad macro-level tailoring is done in advance.

Following the construction of the OSSP, a phase of micro-level tailoring of the method takes place at the Project level. This is where the project-specific characteristics are factored in. In essence, certain elements of the OSSP are chosen depending on the operational needs of the project. Since the OSSP process elements cover all aspects of the software process including those practices which are project specific; e.g., sub-contractor management or project planning, and those practices which are non-project specific, e.g., training or process improvement, the project specific elements of the OSSP must be selected to address the operational needs of the project. The project manager is responsible for this level of tailoring. Following this, specific characteristics or features of the actual project under development are considered and further refinements to the project lifecycle are duly made. Some of these tailoring decisions will be made at the start of the project and recorded in the project plan. For example, it may be decided to produce a High Level and a Low Level Design specification, as opposed to a simpler Detailed Design specification, for a particular software feature if that feature is judged to be particularly complex. Other tailoring decisions will be made dynamically in the course of the project. For example, if commitments change significantly after a project has started, then, based on an impact assessment of the change, it may be decided to invoke the Project Re-Planning process or absorb the impact in the current schedule.

Tailoring at this level also applies to areas that are non-project specific. For example a change to the test process may or may not require piloting based on an impact assessment of the process change. Another example might be to grant a developer a waiver from a particular training course if they satisfy certain criteria.

VI. DISCUSSION

We see in this case an organization that has recognized both the advantages to be gained from using a standardized

management, software project planning, software project tracking and oversight, software subcontract management, software quality assurance, software configuration management, organisation process focus, organisation process definition, training program, integrated software management, software product engineering, intergroup coordination, peer reviews, quantitative process management, software quality management, defect prevention, technology change management, and process change management.

ISD method and the need to provide a method that is tailored to fit the specific requirements of each development project. The desire to adhere to CMM criteria, which requires a standard, measurable ISD process, motivated the development of the OSSP. The institutionalization of the method has involved the creation of an internally-developed notation scheme (PROMPT) to document the method (cf. [56]). The method is available on the web, and is used for all development projects. The development process is measured and monitored, in an extremely public manner. Throughout the Cork offices of Motorola are charts and graphs which indicate progress on various measures. All of these things enforce the ISD method culture.

The Motorola development environment is unique in that the types of development projects undertaken are very predictable and very similar to one another. Because all development projects are modifications to the existing cellular support structure, the broad characteristics of the projects can be defined *a priori*.

Obviously, this type of pre-defined tailoring is not possible in all organizations. However, this case does illustrate that tailoring is necessary, and possible, even in rigidly controlled development environments. Once an organization reaches the point where it can identify the various characteristics or contingencies which occur in its development projects, then it is possible to build in flexibility, along with the rules to allow developers to identify appropriate choices, into the method. This type of tailoring can provide the option of a number of tools and approaches along with the framework to guide the appropriate selection, thus avoiding many of the criticisms of the tool-kit and contingency approaches. It is in fact a very specialized type of method engineering.

The modular division of the ISD method into discrete prime components also has significant advantages in that individual components can be upgraded or replaced as new ideas and concepts emerge which may be worthy of investigation. Yet, the introduction of these new concepts can be implemented in a controlled and rigorous fashion. This allows the method to adapt over time to fit new project requirements. Thus the method provides both the advantages of standardization and the flexibility to cater for changes in the development environment.

This case study has elements relevant to both the contingency factors and method engineering literature discussed above. Firstly, the initial macro-level tailoring which results in the OSSP is analogous to the overarching framework within which methods may be tailored, as recommended by [11] and [15]. Also, the formulation of the basic elements which comprise the method (see Fig. 1) are redolent of the method engineering constructionist strategy. The study extends the literature in both areas in that it details the mechanics of the tailoring process. Thus, it is not the case that individual developers are expected to be familiar with a range of ISD methods from which the appropriate one may be chosen, as recommended by [13] and [14]. Rather, the first macro-level of tailoring is intended to provide a method

which is broad ranging enough to cater for the range of development projects that will be faced. This facilitates a speedy transition to the further ‘fine tuning’ type tailoring that is necessary at the project level. Nevertheless, the Future Project Processes component (see Fig. 1) allows for the incorporation of features that may be deemed relevant. The subsequent micro-level tailoring allows for a precise fit to the contingencies of each specific project. This dual tailoring level allows the valuable CMM elements to be incorporated, but without sacrificing any local strengths of the development process.

REFERENCES

- [1] Jayaratna, N. (1994) *Understanding and Evaluating Methodologies*, McGraw-Hill, London
- [2] Checkland, P. (1981) *Systems Thinking, Systems Practice*, Wiley, Chichester.
- [3] Martin, J. and Finkelstein, C. (1982) *Information Engineering*, Savant Institute, UK.
- [4] DeMarco, T. (1978) *Structured Analysis and System Specification*, Yourdon Press, New Jersey.
- [5] Ross, D. and Brackett, J. (1976) An approach to structured analysis. *Computer Decisions*, September, 40-44.
- [6] Yourdon, E. and Constantine, L. (1979) *Structured Design*, Yourdon Press, New York.
- [7] Edwards, H.M., Thompson, J.B., Smith, P. (1989). Results of survey of use of SSADM in commercial and government sectors in the United Kingdom. *Information and Software Technology*, Vol. 31, No. 1, 420-28.
- [8] Fitzgerald, B. (1994), Whither systems development: time to move the lamppost?, in Lissoni et al., (eds), *Proceedings of Second Conference on Information Systems Methodologies*, pp.371-380.
- [9] 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.
- [10] Eva, M. (1996) *SSADM Version 4: A User's Guide* (2nd Ed.), McGraw-Hill, London.
- [11] Avison, D. and Wood-Harper, A. (1991) Information systems development research: an exploration of ideas in practice. *The Computer Journal*, 34, 2, 98-112.
- [12] Benyon, D. and Skidmore, S. (1987) Towards a toolkit for the systems analyst. *The Computer Journal*, 30, 1, 2-7.
- [13] Davis, G. (1982) Strategies for information requirements determination. *IBM Systems Journal*, 21, 1, 4-30.
- [14] Gremillion, L. and Pyburn, P. (1983) Breaking the systems development bottleneck. *Harvard Business Review*, March/April, 130-137.
- [15] Iivari, J. (1989) A methodology for IS development as organisational change. In Klein, H. and Kumar, K. (1989) *Systems Development for Human Progress*, North-Holland, Amsterdam, 197-217.
- [16] Naumann, J. and Palvia, S. (1982) A selection model for systems development tools, *MIS Quarterly*, March, pp. 39-48.
- [17] Shomonta, J., Kamp, G., Hanso, B. and Simpson, B. (1983) Application approach worksheet: an evaluative tool for matching new development methods with appropriate applications, *MIS Quarterly*, December, pp. 1-10.
- [18] Lee, B. (1979) *Introducing Systems Analysis and Design*, NCC Publications.
- [19] Howe, D. (1984) *Data Analysis for Data Base Design*, Edward Arnold, London.
- [20] Mumford, E., Land, F. and Hawgood, J. (1978) A participative approach to the design of computer systems, *Impact on Society*, Vol. 25, No. 3, pp. 235-253.
- [21] 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.
- [22] JKumar, K. and Welke, R. (1992) *Methodology engineering: a proposal for situation-specific methodology construction*, in Cotterman, W. and Senn, J. (Eds.) *Challenges and Strategies for Research in Systems Development*, Wiley & Sons, Chichester.
- [23] Harmesen, F., Brinkkemper, S. and Oei, H. (1994) Situational method engineering for IS project approaches, in Verrijn-Stuart, A. & Olle, T. (Eds) *Methods and Associated Tools for the IS Life Cycle*, Elsevier Science, North-Holland, pp. 169-194.
- [24] Maynard, H. and Stegemerten, G. *Operational Analysis*, McGraw-Hill, New York.
- [25] Olle, T. (1988) Business analysis and system design specification for an inventory control and purchasing system, in Olle, T., Verrijn-Stuart, A. and Bhabuta, L. (Eds) *Computerized Assistance During the Information Life Cycle*, North Holland.
- [26] Tolvanen, Juha-Pekka (1998). *Incremental Method Engineering with Modeling Tools*, University of Jyväskylä, Finland.
- [27] Aaen, I. (1986) Systems development and theory—in search of identification. In Nissen, H. and Sandstrom, G. (Eds) *Quality of Work versus Quality of Information Systems*, Lund University, Sweden, 202-223.
- [28] Banbury, J. (1987) Towards a framework for systems analysis practice. In Boland, R and Hirschheim, R. (eds.) *Critical Issues in Information Systems Research*, John Wiley and Sons, 79-96.
- [29] Boehm, B. (1976) Software engineering. *IEEE Transactions on Computers*, 25, 12, 1226-1241.
- [30] Jenkins, A., Naumann, J. and Wetherbe, J. (1984) Empirical investigation of systems development practices and results. *Information & Management*, 7, 73-82.
- [31] Westrup, C. (1993) Information system methodologies in use, *Journal of Information Technology*, 8, 267-275
- [32] Iivari, J. and Maansaari, J. (1998) The usage of systems development methods: are we stuck to old practices?, *Information & Software Technology*, Vol. 40, No. 9, pp. 501-510.
- [33] JZmud, R. (1997) Editor's Comments, *MIS Quarterly*, Vol. 21, No. 1, v-vii.
- [34] Sauer, C. and Lau, C. (1994) The adoption of IS methodologies. *Proceedings of 5th Australasian Conference on Information Systems*, Monash University, Australia, 27-29 September 1994.
- [35] CSTB. (1990) *Computer Science and Technology Board* (1990) *Communications of the ACM*, March, 281-293.
- [36] McLean, E. (1973) In Van Horn, R. *Empirical studies of management information systems*. *DataBase*, Winter, 172-180.
- [37] Glass, R. (1991) *Software Conflict: Essays on the Art and Science of Software Engineering*. Yourdon Press, Prentice Hall, Englewood Cliffs, New Jersey.
- [38] Boehm, B. (1981) *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, New Jersey.
- [39] Brooks, F. (1987) No silver bullet: essence and accidents of software engineering. *IEEE Computer Magazine*, April, 10-19.
- [40] Marshall, C. and Rossman, G. (1989) *Designing Qualitative Research*, Sage Publications, California.
- [41] Smith, N. (1990) The case study: a useful research method for information management, *Journal of Information Technology*, 5, 123-133.
- [42] Benbasat, I., Goldstein, D. and Mead, M. (1987) The case research strategy in studies of information systems, *MIS Quarterly*, September, 369-386.
- [43] Lee, A. (1989) A scientific methodology for MIS case studies, *MIS Quarterly*, 13, 1, 33-50.
- [44] Yin, R. (1989) *Case Study Research: Design and Methods*, Sage Publications, California.
- [45] Scott, W. (1965) Field methods in the study of organisations, in March, J. (Ed.) (1965) *Handbook of Organisations*, Rand McNally, Chicago, 261-304.

- [46] Walker, R. (1988) Applied Qualitative Research, Gower, Hampshire.
- [47] Trauth, E. and O'Connor, B. (1991) A study of the interaction between information, technology and society. in Nissen, H., Klein, H. and Hirschheim, R. (eds) (1991) Information Systems Research: Contemporary Approaches and Emergent Traditions, Elsevier Publishers, North Holland,131-144.
- [48] Eisenhardt, K. (1989) Building theory from case study research, Academy of Management Review, 14, 4, 532-550.
- [49] Bogdan, R. and Taylor, S. (1975) Introduction to Qualitative Research Methods, Wiley & Sons, New York.
- [50] Paulk, M., Curtis, B., Chrissis, M. and Weber C. (1993) Capability Maturity Model for Software, version 1.1, IEEE Software, Vol. 10, No. 4, pp.18-27.
- [51] IEEE (1991) Standard for Developing Software Life Cycle Processes, IEEE Computer Society, 345 East 47th St., New York., P1074/D6.1
- [52] Sommerville, I (1992), Software Engineering, Addison-Wesley Ltd, UK
- [53] Royce, W. (1970) Managing the development of large software systems. Proceedings of IEEE Wescon.
- [54] Boehm, B. (1988) A spiral model of software development and maintenance. IEEE Computer, 21, 5, 61-72.
- [55] 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.
- [56] Fitzgerald, B. and O'Kane, T. (1999) A Longitudinal Study of Software Process Improvement, IEEE Software, May/June, pp. 37-45.