Risk Management - Implicit and Explicit

D. Greer and D.W. Bustard
Faculty of Informatics, University of Ulster, Cromore Road, Coleraine, BT52 1SA, Northern Ireland. Email: D.Greer@ulst.ac.uk

Abstract

Experience has shown that software development involves substantial risk. It follows therefore that risk must be addressed effectively within the software development process. This can be achieved by identifying risks explicitly at various stages in the process and then determining how best to deal with those risks. Alternatively, some (perhaps most) of the significant risks can be handled covertly by building risk reduction techniques into the development process. This paper argues that both forms of risk management are needed, with initial emphasis placed on the implicit approach as that is where the greatest return for effort is likely to be obtained. This point is illustrated by considering the earliest stages of requirements engineering for software in which the business context for change is examined. Two particular techniques are discussed: (i) Checkland and Wilson's Soft Systems Methodology (SSM); and (ii) Gilb's Evolutionary Delivery. In each case the implicit risk reduction value of the technique is identified and then the further advantage of an explicit risk treatment explored. This work is contributing to the development of RACE (Requirements Acquisition and Controlled Evolution), a new requirements engineering method.

Keywords: Risk Management, Business Analysis, Evolutionary Delivery

1. Introduction

Risk management research has been carried out in many diverse fields, including environmental science, civil engineering, and, of course, software engineering. Risk analysis seems particularly relevant to computing projects because of the relatively high instance of failure in the industry [20]. It is perhaps surprising, therefore, that in practice few organisations currently use formal risk analysis [19]. Part of the reason may be that software engineers are by nature optimists [4], tending to overestimate the likelihood of success and, by implication, underestimate the likelihood of failure [24]. Also project managers tend to play down risk to some extent in an attempt to develop a climate of optimism and enthusiasm that helps drive a project forward [15].

Another factor inhibiting the uptake of risk management is that often there are so many risks in a software development project that their investigation can become overwhelming. Standard advice is to address only the high impact risks [16] but in practice it can be difficult to decide where to draw the line - indeed it can be uncomfortable to ignore any risk once it has been identified and documented. The situation is made worse by the presence of general risks that are acknowledged to be important but extremely difficult to handle because they affect so much of the development life cycle. One example is the risk of software not meeting client requirements adequately.

This paper suggests that a useful strategy for introducing risk management into a software development process is to consider it in two parts:

1. traditional, *explicit* risk management, in which risks are addressed directly by a process of identification, assessment, prioritisation, planning, resolution and monitoring [2]; and
2. *implicit* risk management, in which the software development process is designed to reduce risk.

Further, the paper suggests that emphasis should be placed on implicit risk management in the first instance because it deals with generic risks that apply across all projects, thus giving a greater return for effort. Avoiding an explicit treatment of generic risks in individual projects also makes it easier to concentrate on those risks that are specific to a particular situation, thereby helping to ensure that they are handled adequately.

Implicit risk management can be defined as any technique that deals with risk inherently. In many respects, all process improvement activities can be interpreted as designing for risk reduction. For example, in applying the key practices of the Capability Maturity Model [23] an organisation is reducing its exposure to risk by introducing activities that help to avoid problems that can have a detrimental effect on a project. Another earlier, and particularly well known example of implicit risk management, is Boehm's COCOMO cost estimation model [1], where his use of cost drivers is effectively a means of assessing a project against standard risks and introducing a contingency adjustment accordingly.

To illustrate the roles of implicit and explicit risk management in a software development process the remainder of this paper examines two techniques that have been used in the RACE requirements engineering method [5] currently under development at the University of Ulster. RACE (Requirements Acquisition and Controlled Evolution) concentrates on the business context for software development, and it is particularly beneficial to deal with risk at this early stage of analysis [9].

The two techniques examined are Checkland and Wilson's Soft Systems Methodology (SSM) [13, 26] and Gilb's Evolutionary Delivery [17]. In each case the implicit risk reduction value of the technique is discussed and then the further advantage of explicit risk treatment considered. In the next two sections of the paper, SSM and Evolutionary Delivery are summarised, illustrated and their risk reduction contributions identified. A following section then demonstrates how these implicit risk reduction techniques may be integrated and extended with explicit risk management techniques into an evolutionary, business-driven, software development strategy.

2. Implicit Risk Management in Soft Systems Methodology

Soft Systems Methodology (SSM) has been developed as an engineering approach to management problems [13, 26]. It is, however, a general problem solving technique and can be used for any system where a process is involved. Examples of its range of application are given in [22]. The methodology is based on an assumption that while some computing problems are 'hard', i.e. they are well defined with exact solutions,

more commonly they are ‘soft’, suffering from a lack of agreement about the nature of the problem, its cause, or its solution.

SSM supports a goal driven approach to computing systems analysis and ensures that the investigation starts with a business analysis. This is achieved by, initially, identifying the intended purpose or purposes of the system in which the computing facilities are to be used. Each such purpose is developed into a behavioural model of the activities necessary to achieve that purpose. Classically, the phases of SSM are summarised by the diagram shown in Figure 1. This divides analysis into *Real World* activities, dealing with tangible objects, and *Systems Thinking* activities, which involve the building and analysis of exploratory abstract models of the ‘problem situation’. The abstract models are subsequently used for comparison with the real world to identify desirable changes.

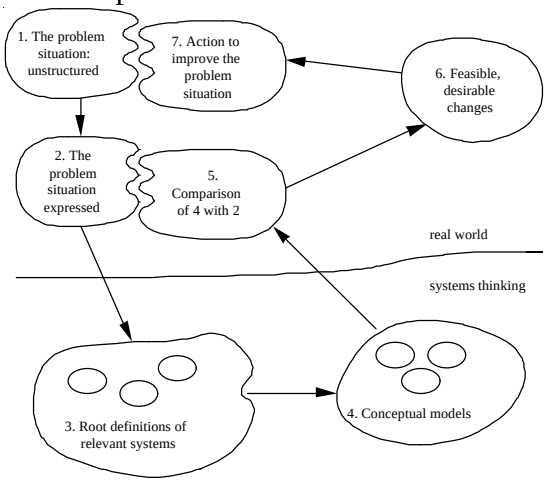

Figure 1: Soft Systems Methodology Summary Diagram

To illustrate the use of SSM, consider the case of a manufacturing division of a company, PrecisionParts. The division manufactures aircraft parts for the parent company and for outside customers, on a competitive basis. The analysis has been started by a request for an investigation into how the division might achieve a more effective organisational structure making full use of information systems.

The first step of the analysis is to generate *root definitions* for the system under investigation. For example, one root definition for the manufacturing division might be: *A manufacturing division owned system which responds to PrecisionParts for the effective manufacture of a range of aircraft parts and which seeks to be a major supplier for both PrecisionParts and the market, within the performance constraints defined by PrecisionParts.*

A root definition identifies a central goal or purpose of a system and the transformation it performs (seeks to be a major supplier). It may also, optionally, include the customer for the transformation (PrecisionParts), the actors of the transformation (not stated here), the owner of the system (the manufacturing division) and any environmental constraints (within the performance constraints defined by PrecisionParts). A *conceptual model* can be constructed (Figure 2), identifying the activities stated or implied in the root definition, and indicating dependencies among them.

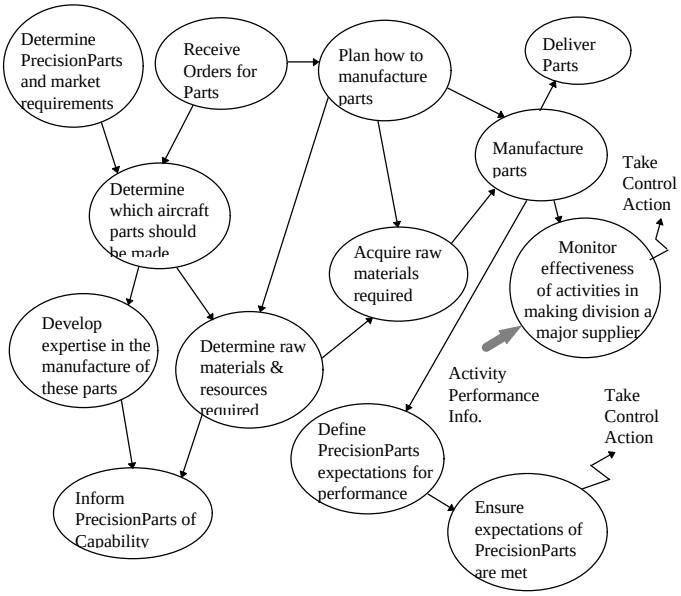

Figure 2: Sample Soft Systems Activity Model for a Manufacturing Division

Systems typically have several root definitions. For example, from the parent company's point of view, the purpose of the manufacturing division is to make parts for it as inexpensively as possible. This perspective then requires another root definition to be produced, together with a matching conceptual model. Conceptual models are later merged to form a single system model, introducing activities to resolve conflicts where they occur.

In phase 5 (Figure 1) a comparison is made between the system models and the real world to test the adequacy of the models and identify real world improvements. One systematic approach is to build a table (Table 1), examining each activity in turn.

Table 1: Sample Activity Table

(1) Activity	**(2) Exists ?**	**(3) Current Mechanism**	**(4) Measure of Performance**	**(5) Proposed Change**	**(6) Comments**
Determine which aircraft parts should be made	Yes	Decision on which parts to make is based on manager experience	% orders filled	Build costing system and inventory of tools to aid decision	May be possible to implement expert system
Determine raw materials & resources required	Yes	Assumption that raw materials can be ordered and resources are available	None	Bill of Materials is drawn up for every part ordered and resources and materials identified	In the past, delays have often resulted while materials were ordered
etc.					

The first column of the table identifies the activities concerned, the second indicates whether or not that activity is currently performed; if so, then the third column describes the mechanism involved. The fourth column defines how the performance of

the activity is measured; the fifth column summarises any proposed changes and the final sixth column can be used to make clarifying comments or add notes.

2.1 Implicit Risk Reduction in SSM

In terms of risk reduction, the SSM approach offers a number of distinct advantages over other systems analysis techniques:

1. *Risk: Undue reliance placed on a computing system to bring about improvement*
   There is a tendency in traditional systems analysis to assume at the outset that computing facilities are needed and that they will largely be responsible for bringing about business improvement. SSM instead focuses on business needs, without any initial assumptions about how those needs will be met. In this way the business can be improved in whatever way is are considered appropriate and may not involve computing development at all.
2. *Risk: System changes do not achieve potential benefit*
   Another problem with traditional analysis techniques is that they tend to focus on improving the current situation and so miss opportunities for innovation. SSM also starts with an initial investigation of the current situation but only to help identify its purpose. The first models developed (root definitions and conceptual models) describe an 'ideal' world and consider the activities that *should* be carried out rather than those that are currently performed. This approach eliminates a common fault of merely implementing a computerised version of the current system.
3. *Risk: System changes fail to satisfy those in the problem situation*
   The SSM approach, because it considers multiple perspectives on the purpose of the system, tends to take more viewpoints into account and so reduce the risk of ignoring the needs of any group or individual in the system. The multiple viewpoint approach also helps to expose differing views and encourage the resolution of conflict (detected when separate models are merged).
4. *Risk: System changes do not have the effect predicted or the system deteriorates over time.*
   It is an established practice in soft systems analysis to include activities that relate to monitoring and control. Monitoring activities measure the effectiveness, efficacy and economy of other activities and take control action as necessary. Hence, these activities reduce the risk of a system change failing to bring about improvement or problems arising later because of changing circumstances.

As well as reducing risks, development techniques can also *introduce* risks so these need to be identified to ensure that there is a net gain. In the case of SSM, the associated risks are relatively small, a common problem that occurs in practice is that the analyst fails to escape from the influence of the current system ('reality seepage'!) and ends up largely following a traditional analysis approach in an SSM style. SSM analysts need good creative skill.

Another risk concern is that the SSM approach typically identifies a large number of potential system improvements and implementing these at the same time could affect the stability of the system or simply be impractical because of the associated cost. One way to reduce this risk is to use a variation of Gilb's evolutionary approach to software delivery [17] in phases 6 and 7 of SSM, where beneficial changes are defined,

implemented and evaluated. The effect of Evolutionary Delivery on risk is discussed in the next section.

3. Implicit Risk Management in Evolutionary Delivery

The Evolutionary Delivery technique for software production is simple, at least in concept. Instead of implementing and installing a software system in one step, parts are developed and delivered in phases. Each delivery is a complete system that is of value to the client. The delivered system is evaluated by the client and the results fed back to the developers who then take that information into account when implementing subsequent phases.

The same approach can be taken when making any type of system change. An evolutionary plan is produced initially based on the objectives for the system being developed. The plan is executed in stages, each with its own objectives, deliverables and system changes. After each stage the plan is reassessed based on an evaluation of the stage delivered. Figure 3 [17] illustrates the evolutionary delivery cycle, implying that there can be large or small adjustments to objectives after each delivery.

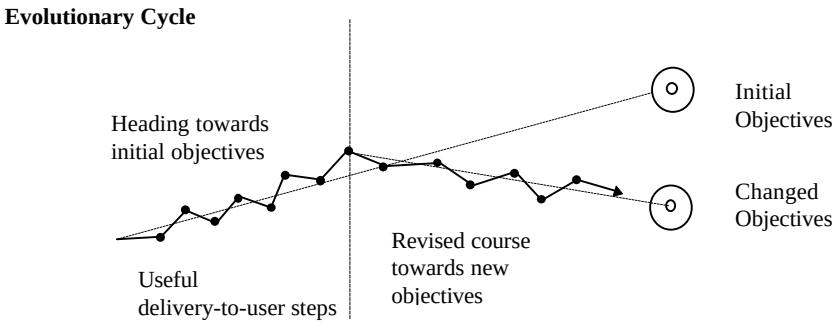

Figure 3. Evolutionary Cycle.

3.1 Implicit Risk Reduction in Evolutionary Delivery

The Evolutionary Delivery technique reduces risk in a number of ways:

1. *Risk: System change fails to satisfy those in the problem situation*
   Proposed changes are prioritised according to their user-value to development cost ratio [17] so the initial changes at least should provide a good return for the costs involved. Also, if the criteria for delivery are accurately established, the risk of losing user support is reduced; it should also help that users are directly involved in the development process and have a chance to comment at each stage. Finally, as changes are made in relatively small steps, any unsatisfactory changes can be repaired relatively quickly anyway.
2. *Risk: Late delivery*
   Usually there is some flexibility in the changes implemented at each stage so if an overrun seems likely some aspects of the change can be deferred to a later release. Thus late delivery need never occur.
3. *Risk: Budget overruns*
   Budget risk is reduced because the stages are relatively small and consequently easier to analyse and cost.
4. *Risk: Environmental changes invalidate the system changes planned*

Since staged changes are scheduled relatively frequently, environmental changes can be detected and responded to relatively quickly.

These potential advantages are offset to some extent by a matching number of risks associated with using the technique:

1. *Risk: Overhead costs are too high*
   Each change step is a project carrying with it various overhead costs. The steps need to be reasonably large to ensure that sufficient benefit is gained for the costs expended.
2. *Risk: Funding is difficult to obtain for a succession of changes*
   Because of the way that many organisations manage their budgets it may be much easier to obtain support for one major change than a succession of small changes.
3. Risk: *The change process deteriorates into 'code & fix'*
   Without sufficient controls there is a danger that changes will not be sufficiently well worked out because corrections can be made relatively quickly.
4. *Risk: People tire of frequent changes*
   It is likely that almost any change to a system will cause disruption and therefore lead to some dissatisfaction, regardless of the eventual benefit. A promise of on-going frequent changes may, therefore, be considered unacceptable.

Despite these threats, the Evolutionary Delivery technique is an appealing concept and represents a helpful way of prioritising and planning change even if a monolithic implementation is attempted. When using Evolutionary Delivery its associated potential risks should be examined explicitly as part of the development process, possibly as part of a formal plan review.

In general, having defined a development process to reduce risk inherently, an explicit risk treatment is still necessary in each project to deal with issues directly related to the techniques being used and the particular type of change being proposed. The next section illustrates one use of explicit risk management in connection with the combined application of Soft Systems Methodology and Evolutionary Delivery, summarising and illustrating earlier work in this area [6, 18]. In doing so it effectively illustrates a combined use of implicit and explicit risk management.

4. Combining Implicit and Explicit Risk Management

When applying SSM with its Evolutionary Delivery extension, the associated risks that need to be considered explicitly are as follows:

1. *Risks in the application of the SSM process*.
   The quality of the models depends on the experience of the consultant/analyst. There may be a risk that the situation under investigation, its scope and its boundaries may not have been properly identified that the models are not validated or accepted by all of the parties concerned. The Evolutionary Delivery risks, listed earlier, should also be examined.
2. *Risks in the system proposed as a result of the SSM study*.
   Given that SSM may result in the recommendation of an innovative, 'revolutionary' system, the proposals need to be carefully examined for operating risks.
3. *Risks in the implementation of the system resulting from the SSM study*.

These are basically risks of acceptance, risks to budget or schedule, or technical risks where the system uses new technology, or uses technology in novel ways.

These risks can be handled as part of the development of an evolutionary delivery plan. The approach is based on a combined cost-benefit and risk analysis of the current system, the proposed system and the change process. The basis of the resulting method is that the order in which a proposed system should be delivered may, initially, be considered a function of five variables.

$$\textit{Implementation Priority} = f(\ \textit{costs, benefits, risk}_{\textit{current system}}\ ,\ \textit{risk}_{\textit{proposed system}},,\ \textit{risk}_{\textit{development}}\ )$$

The derived priority values can then be used to rank the changes, taking account of inter-dependencies among them. The steps of the method, which we have named SERUM (Software Engineering Risk: Understanding and Management) are as follows.

Step 1: Identify the risks for each activity in the required system model
Risks are identified explicitly for each activity in the required system by considering how each activity could produce an unfavourable outcome or adversely affect another activity. Thus at the end of this step, for each activity there is a list of risks and their consequences (Table 2).

Table 2: Identification of risks for activities

(1) **Activity**	**(7)** **Risk**	**(8)** **Consequence**
Determine which aircraft parts should be made	Division undertakes to make parts which it cannot produce	Cost of subcontracting/ loss of reputation/ delay
Determine raw materials & resources required	Raw materials incorrectly identified/ ordered Insufficient resources	Possible penalty for delay in fulfilling orders. Waste of resources.
etc...		

This risk identification process may itself reveal weaknesses in the models and identify the need for a separate activity to be added to deal with that risk. For example, for the activity “Determine raw materials & resources required”, there is a risk of assuming that there are resources available to complete a job. This may lead to a refinement to the SSM model to include an activity: “Check availability of resources”. Hence the process is iterative and ensures that the models are adequate.

Step 2: Identify and assess risks in the current system.
The table generated in step 1 may now be used to derive a list of risks in the current system. Column (9) of Table 3 illustrates how the risks in the current system are documented.

Table 3: Identification of risks in the current system

(1) Activity	(3) Current Mechanism	(9) $Risk_{current}$	(10) Risk Assessment $_{current}$
Determine which aircraft parts should be made	Decision on which parts to make is based on manager experience	Management may make a misjudgment and be overoptimistic	Medium
		Part is incorrectly specified	Low
etc.			

The method of assessing the risks is open to the analyst. Either a risk exposure calculation [2] may be carried or risks classified as in the SEI's SRE method [25] where risks are assessed by a magnitude (high, medium or low) derived from a description of the severity or consequence of the risk and its probability. The latter approach has been used in this example.

Step 3: Refine proposed changes to current system to reduce risks in the current system.
Traditionally, the application of SSM leads to a consideration of problems with some current system. Given that a problem can be defined as a risk that has already occurred and consequently that risks are future problems [12], a consideration of risk is a more comprehensive approach. Having identified risks in the current system, the original changes proposed may now have to be refined and risks identified for those refinements. This is an iterative process which should lead to an improved understanding of what needs to be done.

Step 4: Identify and assess risks in the proposed system.
For each proposed change the risks are identified and assessed in a similar manner to that for the current system (Table 4).

Table 4: Identification of operational risks in the proposed system

(1) Activity	(5) Proposed Change	(11) Operational $Risk_{proposed}$	(12) Operational Risk Assessment $_{proposed}$
Determine which aircraft parts should be made	Build costing system and inventory of tools to aid decision	Estimating skills are lost	Low
		Costing system data is incorrect	Low
etc.			

Specific development risks must also be assessed. Table 5 shows a small part of this risk assessment for the PrecisionParts manufacturing division.

Table 5: Identification of development risks

(1) Activity	**(5) Proposed Change**	**(13) Development Risk$_{proposed}$**	**(14) Development Risk Assessment$_{proposed}$**
Determine which aircraft parts should be made	Build costing system and inventory of tools to aid decision	Inexperience in this type of application may lead to slippage	Medium
etc..		etc.	

Step 5: Refine proposed changes to reduce risk in the proposed system.
The risk assessment of the previous step may now necessitate a refinement to the proposed changes so as to reduce the risk. For example, the risk of the development team being "inexperienced in this type of application" could be reduced by providing the necessary training. This will have a cost and so must be added to the proposed changes column and the risk assessment adjusted accordingly.

Step 6: Carry out cost-benefit analysis on the proposed changes.
Cost-Benefit analysis is a well established technique [e.g. 21] for determining the feasibility of a system. Before accepting any changes their economic feasibility must be determined. However, a full cost-benefit analysis may only be considered worthwhile for changes that have actually been agreed in principle. The recommendations arising out of an SSM analysis are merely proposals at this stage so a non-rigorous cost-benefit analysis is more appropriate. Since a categorisation method has been used for the risk assessment, it is be appropriate to carry out a similar exercise in relation to cost and benefit. The following matrix has been developed [18] to illustrate how a relative measure of cost-benefit ratio may be obtained. For example, a cost considered "Low" and with a "Medium" benefit is classified as II.

Table 6: Matrix for obtaining the cost-benefit assessment of a proposed change

Cost	**Benefit**		
	Low	*Medium*	*High*
Low	III	II	I
Medium	IV	III	II
High	V	IV	III

This process is carried out for each activity (Table 7).

Table 7: Assessment of cost benefit of proposed changes

(1) Activity	**(5) Proposed Change**	**(15) Cost-benefit assessment**
Determine which aircraft parts should be made	Build costing system and inventory of tools to aid decision	II
etc.		

Step 7: Prioritise the proposed changes using the cost-benefit assessment.
From step 6 it is then possible to group the proposed changes into, in this case, five classes. Gilb [17] expresses the view that in planning a project the potential steps with the highest "user-value to development-cost ratio" should be selected for earliest implementation. An approximation to this would be to implement those with the lowest cost-benefit ratio first. Using the assessment technique described those with an assessment of I would be given top priority.

Step 8: Refine priority of the proposed changes using risk assessment.
Having prioritised the proposed changes according to cost-benefit ratio, the analyst may then review the priority in the light of the risk assessments carried out previously. If the current system is found to have a "High" risk exposure associated with one of its activities, then it would be preferable to implement the change that reduces this first before the associated problem occurs. If a change is given a high priority in step 7, but has a high risk associated with it, it may not be advisable to perform it first.

The tediousness of this review process may be reduced by use of an automated tool, the analyst may examine the relevant columns of the table and easily adjust the order.

Step 9: Produce risk control plan for accepted risks.
The process up to now will have produced a prioritised list of system changes which hopefully will reduce the risks in the current system. Inevitably, some changes will have been rejected leaving risks in the current system unanswered. Other risks will have been introduced in the new system and there will be risks associated with the development of the system. Risk control consists of monitoring these risks and preparing contingency plans. Thus for each risk not resolved, a risk control plan with a means of monitoring and taking possible corrective action is produced.

Conclusion

This paper has introduced the notion that risk management may be carried out in an implicit or explicit manner. Building on this notion, it has advocated that any systems development process should be designed in a way that reduces risk inherently thereby reducing the amount of explicit risk management that needs to be handled within the process. The relationship between implicit and explicit risk management has been illustrated by its use in a business systems analysis and development method. The combination of (i) a method that implicitly reduces risk at the business analysis stage with (ii) a method that reduces risks in implementing changes by using an evolutionary approach and (iii) the employment of explicit risk management techniques in planning implementation, should greatly reduce the overall exposure to risk in a project. Tool support for the explicit risk handling proposed is essential and currently under development. In the meantime, however, parts of the technique are being refined through their trial application in the Northern Ireland Civil Service, where SSM is a standard business analysis technique and where explicit risk management is performed in many projects.

REFERENCES

1. Boehm, B.W., *Software Engineering Economics*, Prentice Hall,, NJ, 1981.
2. Boehm B.W., *Software Risk Management*, IEEE Computer Society Press, 1989.

3. Boehm, B.W., *Software Risk Management: Principles and Practices*, IEEE Software, January 1991, pp 32-41.
4. Brooks, F. *The Mythical Man-Month*, Addison-Wesley, MA, 1975.
5. Bustard, D.W., Progress Towards RACE, a 'Soft-Centred' Requirements Definition Method, 1st IFIP/SQI International Conference on Software Quality and Productivity, Hong Kong, pp 29-36, Chapman & Hall, Dec. 1994.
6. Bustard, D.W., Greer, D. and Tate, G., *Enhancing Soft Systems Methodology with Risk Management Techniques*, in Software Quality Management II, vol. 2, Ch. 53, pp. 145-147, Elsevier, Southampton, 1994.
7. Bustard, D.W., Oakes, R. And Heslin, E., *Support for Integrated Use of Conceptual and DataFlow Models in Requirements Specification*, Colloquium on Requirements for Software Intensive Systems, DRA Malvern, May 1993, pp 37-44.
8. Carr, M.L., Konda, S.L., Monarch, I., Ulrich, F.C, and Walker, C.F., *Taxonomy-Based Risk Identification*, Technical Report CMU/SEI-93-TR-6, Software Engineering Institute, Pittsburg, PA 1993.
9. Carter, B., Hancock, T., Morin, J., Robins, N., *Introducing Riskman Methodology*, NCC Blackwell, Oxford, England, 1994.
10. CCTA (Central Communications and telecommunications Agency - UK Civil Service), *Introduction to the Management of Risk*, HMSO Books, London, 1993.
11. Charette, R., *Software Engineering Risk Analysis and Management*, McGraw-Hill, NY, 1989.
12. Charette, R., *Essential Risk Management: Notes from the Front*, Proceedings of The second SEI conference on Software Risk, SEI Pittsburg, PA, 1993.
13. Checkland P.B. & Scholes, J: *Soft Systems Methodology in Action*, Wiley, 1990.
14. Chittister, C. and Haimes, Y.Y., *Assessment and Management of Software Technical Risk*, IEEE Trans. On Systems, Man and Cybernetics, vol. 24, No. 2, Feb. 1994. Pp 187-202.
15. DeMarco, T. And Lister, T., *Peopleware : productive projects and teams*, Dorset House , New York, NY, 1987.
16. Down, A., Coleman, M, Absalon,P., *Risk Management for Software Projects*, McGraw-Hill, London, 1994.
17. Gilb, T., *Principles of Software Engineering Management*, Addison-Wesley, 1988.
18. Greer, D. And Bustard, D.W. *Towards an Evolutionary Delivery Strategy based on Soft Systems and Risk Analysis*, Proceedings of the IEEE Symposium and Workshop on Engineering of Computer Based Systems, 1996, pp. 126-133.
19. Griffiths, C. And Willcocks, L., *Are Major Information Technology Projects Worth the Risk?*, Oxford Institute of Information Management/IC-PARC, Imperial College, 1994.
20. Keen, P., *Shaping the Future: Business Design Through Information Technology*, Harvard Business School Press, Boston, 1991.
21. Kendall, K.E. and Kendall, J.E., *Systems analysis and design*, Prentice Hall, NJ, 1988.
22. Mingers, J. and Taylor. S., *The Use of Soft Systems Methodology in Practice*, Journal of the Operational Research Society, 43(4), 1992, pp 321-332.
23. Paulk, M.C., Weber, C.V., Curtis, W., Chrissis, *The Capability Maturity Model, Guidelines for Improving the Software Process*, Addison-Wesley, 1995.
24. Sage, A, *Behavioural and Organization Considerations in the Design of Information Systems and Processes for Planning and Decision Support*, IEEE Trans. On Man , Systems and Cybernetics, vol. SMC-11, No. 9, September/October, 1981.
25. Sisti, F. and Joseph, S., *Software Risk Evaluation method v 1.0*, Software Engineering Institute Technical Report, CMU/SEI-94-TR-19, Dec. 1994.
26. Wilson B: *Systems: Concepts, Methodologies and Applications*, 2nd Edition, Wiley, 1990.