IDEA GROUP PUBLISHING
701 E. Chocolate Avenue, Suite 200, Hershey PA 17033, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.idea-group.com
ITP4475

Coping with 'Requirements-Uncertainty': The theories-of-action of experienced IS project managers

Tony Moynihan
Dublin City University, Dublin 9, Ireland
Tel: 353 1 7045263; Fax: 353 1 7045442; tony.moynihan@compapp.dcu.ie

ABSTRACT

The notion of 'requirements-uncertainty' has received a lot of attention in the Information Systems and Software Engineering literature. As the level of uncertainty of user-requirements increases, the literature advises project managers to move away from the traditional waterfall life-cycle model and towards more 'experimental' approaches, such as incremental-delivery and prototyping. But there is evidence from empirical research to show that this advice is not always followed. So, it seems that managing requirements-uncertainty may be a more complicated matter.

In this paper, I identify the strategies that experienced IS project managers espouse for coping with requirements-uncertainty. I show that project managers espouse different strategies for coping with different aspects of requirements-uncertainty. I also show that project managers view prototyping and incremental development as 'broad-spectrum' strategies that are salient for coping with a wide range of project risk-drivers, including aspects of requirements-uncertainty.

INTRODUCTION

The notion of 'requirements-uncertainty' has received a lot of attention in the IS and software project management literature (see Davis(1982), Mathiassen and Stage(1990), Fazlollahi and Tanniru(1991) and Saarinen and Vepsalainen (1993)). At its core, the advice in this literature is that as the level of uncertainty around user-requirements increases, one should move away from the traditional waterfall life-cycle model and towards more 'experimental' approaches, such as evolutionary-delivery and/or prototyping.

But there is empirical evidence to suggest that this advice is not always followed. For example, Galleta and El Louadi (1995) collected data on 67 completed IS development projects across the United States. For each project, they chose one user and one systems analyst. From the user and the analyst, they gathered perceptions of the level of requirements-uncertainty in the project. From each analyst, they tried to identify the extent to which the project employed each of Davis's (1982) four generic requirements-determination strategies (ie 'asking users', 'deriving requirements from an existing system', 'synthesising requirements from user activities', and 'discovering requirements through experimenting, e.g. prototyping.') They found many projects in which the requirements-determination strategy most relied on was not congruent with the perceived level of requirements-uncertainty. Other empirical studies involving the notion of 'requirements-uncertainty' have identified similar discrepancies between theory and practice (see, for example, Naumann, Jenkins and Wetherbe (1983) and El Louadi, Galleta and Sampler (1999)).

So, it seems that managing requirements-uncertainty is not just a matter of applying a simple 'recipe.' It is this conjecture that motivates this paper. The aim behind the work described here is to learn more about how IS project managers experience the notion of requirements-uncertainty. Specifically, I try to locate requirements-uncertainty within the broader canvas of issues that project managers say they have to deal with. I then try to identify the strategies that project managers say they use to cope with requirements-uncertainty.

I begin by reviewing some of the mainstream formulations of 'requirements-uncertainty' in the literature. I then describe fieldwork which aimed to identify the strategies that project managers use to address a variety of project risk-drivers, including 'requirements-uncertainty.' I show that project managers **say** they would use **different** strategies to cope with **different** aspects of 'requirements-uncertainty.' I also show that project managers see prototyping and incremental development to be 'broad-spectrum' solutions for coping with a wide range of project risk-drivers, including aspects of requirements-uncertainty.

Hereinafter, I shall refer to 'requirements uncertainty' as RU and to project managers as PMs.

SOME MAJOR FORMULATIONS OF REQUIREMENTS-UNCERTAINTY

Davis (1982) introduced the concept of *overall requirements process uncertainty.* This concept is operationalised as the 'sum' of three variables : *the existence/stability of a set of usable requirements, the level of ability of users to specify requirements* and *the level of ability of analysts to elicit and evaluate requirements*. In turn, these latter three variables are described as being the 'sum' of :

- *uncertainty deriving from the utilising system* (eg the stability of the environment into which the new system is to be embedded, whether the activity the system is to support is structured or unstructured.)
- *uncertainty deriving from the application* (eg its complexity, the number of users, extent of change to structures/tasks.)
- *uncertainty deriving from the users* (eg extent of experience in using computers, level of understanding of the application, politics.)
- *uncertainty deriving from the systems analysts* (eg extent of experience with similar systems, knowledge of the business.)

With increasing levels of *overall requirements process uncertainty*, Davis advocates that one moves from a primary strategy of asking users what they want, through basing the solution on an existing system, through deriving requirements from an analysis of the user-tasks to be supported, to an experimental approach (eg prototyping).

Building on Davis (1982), Burns and Dennis (1985) introduced a distinction between *requirements-uncertainty* and *requirements-complexity*. They define *requirements-uncertainty* in terms of :

- *the degree of 'structuredness' of the user tasks to be supported*
- *the degree of understanding the users have about their tasks*

FIGURE 1 : Selection of approach based on complexity and uncertainty (Burns and Dennis(1985))

	Uncertainty Low	*Uncertainty High*
Complexity High	Waterfall Life Cycle Model	Mixed Model
Complexity Low	Prototyping	Prototyping

- *the degree of experience and training of the system developers.*

They defined *requirements-complexity* in terms of

- *relative project size*
- *the number of users*
- *the volume of new information required from the system*
- *the complexity of this new information*

They suggest the choice of basic project approach (i.e. traditional waterfall model v prototyping v a mixed model) be as shown in Figure 1.

As does Davis (1982) in relation to his definition of RU, Burns and Dennis aggregate ratings of the individual components of complexity and uncertainty to obtain overall complexity and uncertainty ratings.

Stork and Sapienza (1995) distinguish between RU and 'equivocality.' They define RU in much the same terms as other authors. They define equivocality as being the difference between the level of agreement and understanding between the people involved which is needed to accomplish the goals of the project, and the existing level of agreement and understanding. Equivocality, in their view, is a function of aspects of the project such as the degree of innovativeness of the task, and aspects of the people involved, such as how diverse people are in terms of training and background. To reduce equivocality, these authors recommend that people must interact to communicate their different perspectives (not just factual data) and to resolve their conflicting views.

Based on Principle Components Analysis of questionnaire-items completed by project managers, Nidumolu (1996) identifies three dimensions of RU :

Requirements Instability : The extent of changes in user requirements over the course of the project.

Requirements Diversity : The extent to which users differ among themselves in their requirements.

Requirements Analyzability : The extent to which the process for converting user needs to a set of requirements specifications can be reduced to mechanical steps or objective procedures.

Nidumolu found evidence to suggest that total project RU, defined as being the sum of the project's scores on these three dimensions, is negatively associated with ultimate project and 'product' performance. He also found that the use of appropriate software development standards reduced the negative effects of RU on both these outcome variables.

In summary of the literature, there appear to be common features across the different definitions of RU. All definitions treat RU as a multi-dimensional construct. Specifically, requirements-uncertainty is defined to be the aggregation of a number of requirements-uncertainty generating 'sources.' The level of requirements-uncertainty for a project is measured by rating the project separately on each of these 'uncertainty sources', and by then combining (e.g adding) the individual 'uncertainty' ratings to yield an overall requirements-uncertainty rating.

Also, there seems to be a fairly strong consensus amongst researchers on what constitute the main 'uncertainty-sources.' 'Sources' common to many definitions of RU include :

- Attributes of the application (complexity, stability, novelty, level of change involved);
- Attributes of the users (number, previous computer experience, diversity of their needs, their understanding of the application);
- Attributes of the analysts/developers (knowledge of the application, knowledge of the business);
- Wider aspects of the organisation (e.g. any unhelpful 'politics').

THE FIELD STUDY

In an earlier study (Moynihan 1996), I identified the situational variables which a sample of experienced project managers (PMs) claimed they took into account when planning and managing bespoke software development projects for new, external clients. I used the technique of personal construct elicitation for that purpose (Bannister and Fransella 1989). A personal construct is a bipolar distinction which a person uses when contrasting different people, objects, situations, and so on. For example, for me, an important distinction between dogs is the likelihood that a dog will bite me! So, when comparing dogs, or thinking about a particular dog, I am likely to think in terms of 'will he / won't he bite me?' People have multiple sets of many interacting constructs to help them to make sense of the world. The task of identifying the set of constructs used by a person in a particular context is called *personal construct elicitation.*

In that study, I asked each PM to make a list of the systems development projects he/she had worked on as project manager over the past year or two. I then selected three projects randomly from the list and asked the PM : *In what important ways are any two of these three projects the same, but different from the third, in terms of important situational factors you had to think about when planning the project?*

I asked the PM to repeat this task with different triads of projects, until no new situational constructs were being elicited. Using this process, I elicited 113 different constructs from the project managers. The constructs elicited reflect most of the situational variables which have identified as project 'risk-drivers' by IS and software project risk researchers, and reflect some additional situational variables not previously identified in the literature.

For the present study, I selected 34 constructs from the full set of 113 constructs. This sub-set of constructs was chosen to give coverage of all of the main themes identified in the full set of constructs. The constructs I selected are shown in Table 1. Table 1 includes constructs that seem to reflect most of the elements of 'requirements uncertainty', as described above. These 'requirements uncertainty' constructs are shown in italics. The purpose of Table 1 will become clear later.

I then constructed five hypothetical project profiles. A project profile consisted of 34 poles, one drawn randomly from each of the 34 chosen constructs. To construct a project profile, I used random numbers to decide whether the left-hand pole or the right-hand pole of the construct should appear in the profile. Within any one profile, I applied a fixed probability of selecting the right-hand pole. But I varied this probability across the profiles. In this way, I generated large variation in apparent 'riskiness' across the five hypothetical projects.

My subjects were twenty IS project managers (PMs), all located in Ireland. Many of these PMs had participated in the earlier study. All managed custom-built, software-intensive IS development projects for external clients. All twenty had at least six years experience of running projects. All were in the range 30-60 years of age. Almost all were owners or directors of their companies. The numbers of developers employed in their companies ranged from two to twenty persons. The scale of the projects they typically managed fell within the ranges:

Project Duration : 2-18 months
Project Effort : 2-36 man-months
Project Team Size : 1-6 people

All worked with mainstream, current technology. All were in the business of providing bespoke information systems 'solutions' to commercial clients.

I spent about one hour with each PM. What follows is a condensed description of the procedure I used. I explained that the purpose of the exercise was to learn more about the 'recipes' that experienced PMs use to cope with project risk, and that I would use a hypothetical project to help with this task. I then gave the PM a set of 34 index cards, one for each of the 34 constructs. The top, visible face, of the card showed both poles of the construct. The bottom face, **which I asked the PM not to look at**, showed the pole that belonged to the hypothetical project profile I had chosen for that PM.

I asked the PM to sort the constructs into three piles. The first pile to contain constructs that, depending on which of the two poles applied, could make a VERY BIG DIFFERENCE to the 'riskiness' of a project. The other two piles to contain constructs that could make a BIG DIFFERENCE and SOME/LITTLE/NO difference to 'riskiness', respectively.

Then I asked the PM to focus on the VERY BIG DIFFERENCE pile. I asked him/her to start by choosing the constructs, singly or in combination, which could make the biggest difference to project risk. I then explored with the PM why he/she had chosen as he/she had done. I then asked the PM to turn over the card(s) to reveal the pole(s) which applied to the hypothetical project, and to give me a reaction to this piece of information. I then explored the strategies, tactics, tricks, and so on, that the PM would use to cope with that pole. I repeated this procedure until all of the VERY BIG DIFFERENCE constructs had been turned over. At intervals, I asked the PM to sum up his/her feelings about the unfolding

TABLE 1 : Numbers of PMs putting each construct into each pile

		VBD	BD	LND
Someone on the customer's side has taken clear, committed ownership of the project	Nobody wants to 'own' the project	17	3	0
The customer has realistic expectations about time, cost and what's 'do-able'	The customer has unrealistic expectations about time, cost and what's 'do-able'	16	4	0
The customer's PM has the needed time/skill/authority	The customer's PM lacks the needed time/skill/authority	15	5	0
There seems to be no hidden agenda	The 'real' agenda seems to be hidden	14	6	0
They don't disagree amongst themselves about what's needed	They disagree amongst themselves about what's needed	14	6	0
We will be using familiar languages/tools	We will be using unfamiliar languages/tools	13	3	4
No major changes to the customer's workflow/ procedures	Major change to the customer's workflow/procedures	12	6	2
No tricky interfacing with existing applications	Some tricky interfacing with existing applications	12	5	3
The platform/environment is familiar to us	The platform/environment is new to us	12	5	3
The people most affected seem to genuinely want the new system	The people most affected don't seem to really want the new system	11	7	2
The new system isn't 'mission-critical' to the customer	The new system is 'mission-critical' to the customer	11	7	2
We'll be able to juggle a bit with time-scales	We'll have to work to tight customer-imposed time-scales	10	8	2
We've experience of this application	The application is new to us	10	8	2
We won't need to subcontract anything	We'll have to do significant sub-contracting	10	7	3
We can pilot the new system until we get it right	The new system has to go right first-time	9	9	2
We've only to satisfy a single group of similar users	We've to satisfy multiple groups of users with different needs	9	5	6
We won't have to change other developer's code	We'll have to change other developer's code	8	9	3
We won't have to share control of the project with a third-party	We'll have to share control of the project with a third-party (eg consultants)	8	8	4
We've a good knowledge of the customer's industry	We've little or no knowledge of the customer's industry	7	12	1
The new system doesn't have to be particularly adaptable to future needs	The new system must be adaptable enough to cope with unknown future needs	5	13	2
The customer is experienced in running computer projects	The customer is not experienced in running computer projects	5	11	4
The application logic is straightforward	The application logic is complex	5	9	6
The customer has the willingness and skills to manage implementation issues	We'll have to sort-out/drive implementation	4	14	2
We will need only one or two people on the project	We will need three or more people on the project	4	8	8
The project will take three months or less	The project will take more than three months	4	8	8
The customer is able to define the problem in IT addressable terms	We will have to work with them to define the problem	*4*	*8*	*8*
We'll be able to show the customer an early prototype	It won't be possible to show the customer an early prototype	3	12	5
We're dealing with experienced computer users	We're dealing with inexperienced computer users	3	10	7
We've no credible competitor for this project	We are up against a credible competitor for this project	3	4	13
We'll be dealing directly with the users	We'll be working through their IT department	2	12	6
The new system involves only a single functional area	The new system will span a number of functional areas	*2*	*10*	*8*
This is probably a 'one-off' project for this customer	This project could lead to other projects for this customer	1	8	11
The customer is a very small company	The customer is a larger company	1	5	14
It's a transaction processing system	It's an MIS/DSS type system	0	7	13

project, and to describe the advice he/she would give to a PM faced with managing that project. I concluded by asking the PM to rate the likely outcome of the project on a set of scales.

RESULTS

Putting requirements uncertainty into the larger context

In TABLE 1, for each construct, I show the numbers of PMs placing that construct in the VERY BIG DIFFERENCE, the BIG DIFFERENCE and the LITTLE/NO DIFFERENCE piles. The constructs are shown in decreasing order of the numbers of PMs placing that construct in the VERY BIG DIFFERENCE pile. The constructs which seem to most closely relate to aspects of RU, as formulated in the literature, have been italicised.

From TABLE 1, it seems that the 'requirements-uncertainty' constructs do not 'top the poll' in terms of risk-generation. Also, the RU constructs are not all seen to be of equal importance. Of the RU related constructs, the presence of hidden agendas and disagreement are seen by PMs to be by far the biggest risk generators. 'Non-political' aspects of requirements-uncertainty such as application complexity, the client's ability to define their 'problem', and the level of users' previous computer experience, are

FIGURE 2 : Some verbatim strategies espoused for addressing requirements-uncertainty

The 'real' agenda seems to be hidden	*Insist on sign-offs before doing anything; Document everything; Make sure people see precisely what's being contracted for; If power struggle, walk away; Sign-offs on everything by someone with clout; Be as rigorous as you can in specifying what the system is to do. Get sign-offs on everything…req spec…the lot; Agree detailed criteria for acceptance testing. Mock-up screens in advance so they know exactly what they are getting.*
They disagree amongst themselves about what's needed	*Sign offs every step of the way; Spec at a fine level of detail…nothing vague; First phase a document on their needs + rationale…to be agreed...then prototype; Summarise the issues from the various camps then pass the buck to someone with clout to decide; Get someone on the clients side to push things through; Think of a price for the contract, then treble it! Try for time and materials; First phase time and materials to define requirements; Hammer down the sope and the functional spec…treat this as a separate contract first phase; Have a heavy session with the heads of these people…explore why they can't agree…if it's about conflicting goals, tell them they need a consultant to help them sort it out…if they've similar goals, an IT person can usually resolve it; If they're signing-off at all stages, you can't go wrong; Escalate to their main project guy…tell him to bang their heads together; You have to facilitate reaching a consensus…if it's a big project, suggest they bring in an independent consultant…then we'd pick-up from there.*
Major change to the customer's workflow/procedures	*Spell out who is responsible for ancilliary activities like training, IR; Prototype; Build a good contingency into system definition stage; Make sure user training, union negotiations etc. are part of clients responsibility and are started early on; OD and BPR issues to be dealt with first; Sit down with users and prepare the way before you get stuck in; Break implementation into stages…if stage one works, then implement stage 2 etc; Point out that they are responsible to ensure organisation is ready for the change; Budget for more user-testing than usual; Roll out in phases; Be very explicit with client about what has to change; Need flexibility in time scales. Planning and handling change becomes a key part of the project.*
The application is new to us *We've to satisfy multiple groups of users with different needs*	*Work with Cecil on a day-to-day basis…or whoever is running the application…learn from him; Spend time at the client's site…at his expense…to learn the application…plus background reading; Do an initial req spec, check then if we had the needed skills in-house, decide then whether it's really our bread and butter... then the piloting.* *Put the right communication structures in place; Need a group of user representatives to work through; Don't try to satisfy everybody in the first implementation…take most straightforward or urgent group first; Put onus on the client to rank the priorities.*
We've little or no knowledge of the customer's industry	*Get them to document clearly what they want to be done…the more detail we can get from users in the spec. the better…learn from them; Get consultancy help or 'lessons' from someone in the industry; Put more into the specification stage…this would be a learning phase so budget for this…if you want to get into their industry, take the hit for this; Stretch the timescale of the specification stage to give yourself time to learn the industry.*
The new system must be adaptable enough to cope with unknown future needs	*Avoid hard coding things…have as much of the application parameterised as possible; Scope application into multiple phases…what you're building now must include interfaces that these things can plug into in the future.*
The application logic is complex	*Nail things down by getting agreement on budget for requirements that are visible up-front…get agreement that requirements that surface later will be treated as extras; We're into a big educational thing for us…maybe send people on courses…to speed-up the learning curve; If we don't know what we're doing, we'd walk away…or bring the expertise in-house or out-source it…don't chance learning it on the fly.*
We're dealing with inexperienced computer users	*Find out what sorts of people they are…decide on how to build them into the project…when they are going to be involved; Release early versions for them to play with and become familiar with the operating system and the new application. Intensify user-training; Involve users in generating and applying business test cases; Limit the increment of change to what users can cope with.*

seen by PMs to be relatively low risk generators.

Espoused Strategies

FIGURE 2 contains a 50% random sample of the phrases used by PMs to describe the strategies they say they would use to cope with RU. Apart from some minor editing, these interview extracts are verbatim.

An analysis of the complete set of interview transcripts suggests that there are common themes underlying the espoused strategies **within** each pole. There also appear to be differences in themes **across** the poles. *The 'real' agenda seems to be hidden* and *They disagree amongst themselves…* seem to share a common theme : 'go bureaucratic' (e.g. insist on sign-offs, 'put everything in writing'), presumably with the goal of self-protection. The strategies for coping with *Major change to the customer's workflow/ procedures,* relate to aspects of change management. In particular, to the need for the client to accept responsibility for readying the organisation for change, the importance of detailed implementation planning, the need to break major change into manageable stages, the need to follow an incremental-development life-cycle model, and the importance of prototyping.

The application is new to us, We've little or no knowledge of the client's industry and *The application logic is complex* share common themes. PMs emphasise strategies aimed at rapidly getting their team 'up the learning curve', ideally at the client's expense, through learning from users or by acquiring the needed knowledge through subcontracting or recruitment.

We've to satisfy multiple groups of users with different needs is addressed primarily by user-involvement mechanisms and, where needed, by resort to the client's power structure. *The new system must be adaptable…* is addressed primarily by using appropriate technical design principles, and by using an incremental development life-cycle model.

We're dealing with inexperienced computer-users is addressed in obvious ways : user-training through various forms of involvement, and , initially at least, by limiting any change to a level that inexperienced users can cope with.

Use of prototyping and incremental development

Most of the PMs saw prototyping, in some shape or form, to be an appropriate response to a wide range of project risk-drivers. The need for prototyping was believed to be **most** needed when one or more of the following applied :

- *The customer has unrealistic expectations ;*
- *The new system is mission-critical;*
- *The system must go right first time;*
- *Major change to customer's workflow;*
- *Inexperienced users.*

The terms *incremental delivery* and *evolutionary development* were not used at all by the PMs. But equivalent terms were frequently used :

"Let's get this bit done and in place first, then we'll see what's next."

"Put in the minimum requirements first."

"Do the easy things in phase 1, then see where we've got to."

"What's 'good enough' is what we should do first."

" Break it down into mini-projects."

The need for an incremental/evolutionary approach was felt to be most needed when one or more of the following applied :

- *The customer has unrealistic expectations ;*
- *Major change to customer's workflow…;*
- *The new system is mission-critical;*
- *Unfamiliar languages/tools;*
- *The application is complex.*

CONLUSION

The aim of this paper has been to learn some more about how IS/software project managers experience the notion of 'requirements-uncertainty', and to identify at least some of the 'recipes' they use to cope with it.

Requirements-uncertainty does not 'top the list' of IS project managers' concerns. For example, lack of real project 'ownership' and unrealistic client expectations are seen to be bigger risk-generators. Aspects of the 'political' side of requirements-uncertainty (hidden agendas, disagreement, and the like) , are seen to be far bigger risk-generators than are 'non-political' aspects of requirements-uncertainty (unfamiliar application, inexperienced users, and so on).

The data suggest that project managers use a rich mixture of strategies to address requirements-uncertainty. The data also suggest that project managers use very **different** strategies, or combinations of strategies, to address **different** components of requirements-uncertainty. It also seems that project managers see prototyping and incremental development as useful for addressing a wide variety of concerns, in addition to requirements-uncertainty.

To end on a provocative note. The findings prompt the question "Is the notion of 'requirements-uncertainty' **useful** in the real-world of IS project management? " If it is true that **different** strategies are salient for coping with **different** dimensions of 'requirements-uncertainty', does it make sense to combine the ratings for a project across a set of dimensions to obtain a single measure of requirements-uncertainty? In other words, has the literature over-abstracted the notion of requirements-uncertainty?

REFERENCES

Bannister D. and Fransella, F. (1989) Inquiring Man : The Psychology of Personal Constructs (third edition) (Routledge, London).

Baskerville, R. L. and Stage, J. (1996) Controlling Prototype Development through Risk Analysis. *MIS Quarterly*, **20**(4)**,** 481-504.

Burns, R.N. and Dennis, A.R. (1985) Selecting the Appropriate Application Development Methodology. *Data Base,* Fall, 1985, 19-23.

Davis, G.B. (1982) Strategies for information requirements determination. *IBM Systems Journal*, 21(1), 3-30.

El Louadi, M., Galleta, D.F. and Sampler J.L. (1999) An Empirical Validation of a Contingency Model for Information Requirements Determination. DATA BASE, **29**(3), 16-28.

Fazlollahi, B. and Tanniru, M.R. (1991) Selecting a requirement determination methodology- contingency approach revisited. *Information and Management*, **21 ,** 291-303.

Galleta, D.F. and El Louadi, M. (1995) L'effet de L'incertitude et des strategies de determination des besoins de l'utilisateur sur les projets d'informatisation. *Canadian Journal of Administrative Systems*, **12**(1), 56-76.

Keil, M., Cule, P., Lyytinen,K., and Schmidt, R. (1998) A framework for identifying software project risks. *Communications of the ACM*, **41**(11), 76-83.

Mathiassen, L. and Stage, J. (1990) Complexity and uncertainty in software design. In *Proceedings of the 1990 IEEE Conference on Computer Systems and Software Engineering,* (IEEE Computer Society Press), 482-489.

Moynihan, T. (1996) An Inventory of Personal Constructs for Risk Researchers. *Journal of Information Technology*, **11**, 359-371.

Naumann, J.D., Jenkins, A.M. and Wetherbe, J.C. (1983) *The information requirements determination contingency model : An empirical Investigation* (University of Minnesota, Minneapolis).

Nidumolu, S.R. (1996) Standardization, requirements uncertainty and software project performance. *Information and Management,* **31 ,** 135-150.

Redmill, F. (1997) *Software Projects : Evolutionary versus Big-Bang Delivery* (John Wiley and Sons, Chichester,England).

Saarinen, T. and Vepsalainen, A. (1993) Managing the Risks of Information Systems Implementation. *European Journal of Information Systems*, **2**(4), 283-295.

Stork, D. and Sapienza, A.M. (1995) Uncertainty and Equivocality in projects: Managing their implications for the project team. *Engineering Management Journal*, **7**(3), 33-38.