A Field Study of Requirements Engineering Practices in Information Systems Development[¶]

Khaled El Emam[*]
Nazim H. Madhavji[*][†]

[*]School of Computer Science, McGill University
[†]Centre de Recherche Informatique de Montreal (CRIM)

Abstract

To make recommendations for improving requirements engineering processes, it is critical to understand the problems faced in contemporary practice. In this paper, we describe a field study whose general objectives were to formulate recommendations to practitioners for improving requirements engineering processes, and to provide directions for future research on methods and tools. The results indicate that there are seven key issues of greatest concern in requirements engineering practice. These issues are discussed in terms of the problems they represent, how these problems are addressed successfully in practice, and impediments to the implementation of such good practices.

1 Introduction

The requirements engineering process is arguably one of the most important processes in software development. Unfortunately, however, the record is poor. For example, it has been found in a number of previous studies that the requirements engineering phase is the source of the majority of detected software code errors [3][9][14][19]. It is thus imperative to focus specifically on this process and find ways to improve it.

As part of our research efforts to make recommendations for improving requirements engineering processes, this paper describes a field investigation of contemporary requirements engineering practices. The general objectives of the investigation were to formulate recommendations to practitioners for improving requirements engineering processes, and to provide directions for future research on methods and tools that would result in process improvements. The specific objectives of the investigation were to identify the most pressing problems faced by practitioners, how these problems are addressed successfully in practice, and the major impediments to the successful implementation of such good practices.

While there have been previous field investigations of software engineering practice, ours differs in a number of aspects. For instance, Curtis et al. [6] conducted a field study of software engineering practices. Their findings covered the requirements engineering process as well as other software engineering processes. Our study focuses only on the requirements engineering process, and is therefore closer in focus to the field study reported by Lubars et al. [15]. Moreover, the study described in this paper differs from the above two in that: we used two different data gathering methods (interviews and inspection of documentation); we collected data on 60 cases (as opposed to 23 and 17 in the previous studies respectively); the data we collected constituted two types of cases: descriptive and prescriptive (as opposed to primarily descriptive cases in both of the previous studies); we included an explicit validation step with a subset of the original informants; our domain of analysis was business Information Systems that are fully customized for individual user organizations; and we explicitly attempted to identify impediments to the implementation of good requirements engineering practices. These differences were intended to provide a better understanding of requirements engineering practices in the chosen domain of analysis.

Briefly, the results of the study indicate that, in

¶ This research has been supported, in part, by the IT Macroscope Project and NSERC Canada.

0-8186-7017-7/95 $04.00 © 1995 IEEE

addition to technical issues, non-technical issues are of major importance during the requirements engineering process in terms of problems, solutions, and impediments to implementing the solutions. Furthermore, many of these issues are of primary concern at the planning stage of the requirements engineering process. The significance of these findings are that they provide direction for future research targeted at assisting requirements engineering process planning, and they include recommendations for practitioners.

In section 2 the research method is described. Section 3 is a detailed description of the study's findings. Section 4 concludes the paper with a summary of the key points and directions for future research.

2 Research Method

The empirical research method followed in this investigation is the multiple case study method as described by Yin [26], and recommended for research on software development projects [6]. Furthermore, it has been asserted that this method is more suitable for research on organizational, as opposed to individual, problem-solving (i.e., multiple individuals) [21].

The unit of analysis of this study was the requirements engineering process. The context of our

Business of Organization	
Pharmaceutical and Health Services	13%
Financial/Insurance	14%
Govemment	33%
Retail and Distribution	27%
Other	14%
Functional Area of Information System	
Purchasing/Sales/Marketing	47%
Finance	33%
Other	20%

Figure 1(a): Characteristics of interview requirements engineering cases (15 cases).

Business of Organization	
Financial/Insurance	14%
Govemment	40%
Transportation	7%
Engineering	14%
Other	27%
Functional Area of Information System	
Purchasing/Sales/Marketing	7%
Finance	27%
Production/Manufacturing/Operations	33%
Transportation/Logistics	14%
Telecommunications	7%
Other	14%

Figure 1(b): Characteristics of document requirements engineering cases (15 cases).

research study was the requirements engineering phase of a software system development method (henceforth Method X). Method X has been developed and is marketed by an Information Systems consultancy firm with clients worldwide. The ultimate objective of the requirements engineering phase in this method is to determine the cost-effectiveness of the Information System to be developed and make a *go/no-go* decision based on it. However, prior to the performance of such an analysis, the business requirements must be elicited and a *recommended solution* formulated. The recommended solution consists of a requirements specification, a high level architecture, and an analysis of how the proposed system will fit into, and have an impact on, the existing user organization.

The research method consisted of two main activities: a data gathering step using multiple sources, and a synthesis and validation step.

2.1 Data Gathering Step

The objective of this step was to formulate a set of cases. In total there were 60 cases. Two types of cases were sought. The first type of case is descriptive, and the second type is prescriptive. The descriptive cases represent requirements engineering processes performed at client sites of the company that developed Method X. The prescriptive cases represent rules of good practice that were solicited from practitioners using Method X.

For requirements engineering process (descriptive) cases, seventeen individuals were interviewed. The interviewees provided a description of the problems faced during the requirements engineering process and how these problems were handled, if they were. Of those interviewed, 35% had a technical background, and 94% had a project management background[1]. These interviews resulted in fifteen cases[2] summarized in Figure 1(a)[3].

1 It should be noted that an interviewee can be characterized as having an intersection of backgrounds. Therefore, the total does not add up to 100%. For example, some senior analysts take on the role of project manager.

2 For 2 of the descriptive cases, 2 interviewees provided information.

3 No information on the actual sizes of the Information Systems is available. For some cases the system was not actually developed yet, and for some other cases the project was terminated at the end of the requirements engineering phase or at some later phase. For the remaining cases where a system was actually developed, no data that can be compared meaningfully across organizations was available. For example, a summary of system sizes using LOC measures would be misleading given that different organizations use different programming languages and use different counting standards.

Furthermore, the documentation resulting from fifteen other requirements engineering processes was inspected. The documentation included requirements engineering deliverables (e.g., process and data models), as well as some project management documentation and post-mortem reviews. These fifteen cases are summarized in Figure 1(b).

The second type of case (prescriptive) is a set of "rules" and their rationale. Each set of rules reflected the aggregate experiences of a single practitioner. The experience of the practitioners ranged from 2 requirements engineering processes and up to 35. These rules concerned problems that are commonly faced during the requirements engineering process, how they *should* be addressed, and why they are sometimes not addressed in practice. Example rules are *"users participating in the requirements engineering process should always be at the supervisory level"* and *"if the analysts are unfamiliar with the application domain then consider building a prototype, consider software packages, and consider letting the analysts construct models of the current system (the one being replaced) as a means for them to learn about the application domain"*. The rule sets were solicited from thirty practitioners. Of these, 43% had a technical background, and 70% had a project management background.

A two-pronged strategy was utilized in collecting the data for these cases. The first was to collect information about best practices and why they are considered best practices. The second was to collect information about infidelity or deviations from the standard requirements engineering process defined in Method X and the rationale behind such deviations. Such infidelity captured the variations across the requirements engineering processes. Both of the above approaches have been recommended as ways for capturing information that can be employed in process improvements [17]. The high level questions that were posed to gather this data are shown in Figure 2. It should be noted that these questions were not posed in the form of a questionnaire, but represent the general reasoning behind the specific questions.

- What changes were made to the standard Method X and why? If no changes were made, then why not?
- For each activity in Method X, under what circumstances would this activity be performed and why? Under what circumstances would you not perform this activity and why? What level of detail is appropriate for the output of this activity and why?
- For each activity in Method X, what individual, project and/or organizational characteristics would have an impact on its performance, how, and why?
- What individual, project and/or organizational characteristics would have an impact on the overall success of the requirements engineering process, how, and why?
- In retrospect, what changes should have been made to the requirements engineering process and why?

Figure 2: High level questions used in the data gathering step.

2.2 Synthesis and Validation Step

For the purposes of the results presented in this paper, the cases (60 in total) had to be synthesized into a number of issues that were of greatest concern to practitioners. An issue is denoted by a label given to a set of related problems, the approaches used to address them successfully in practice, and difficulties encountered implementing such good practices.

The synthesis task proceeded iteratively. Each iteration synthesized the information from one more case. The approach was to identify the issues in each case. On subsequent iterations the synthesized issues were refined and elaborated upon. This was repeated until all the cases were exhausted. The authors subsequently decided which issues were of greatest concern by the number of times they were repeated and the importance attached to them in the original cases.

For the validation task, six of the original interviewees were interviewed again. They were presented with the set of issues that were identified and asked to agree/disagree with the issues, and comment on them in terms of their correctness and importance. Furthermore, a version of the discussion presented in the results section of this paper was sent to three of the original participants for comment. As well, two formal presentations were made to three of the original interviewees, and the ensuing discussion resulted in feedback about the correctness and importance of the identified issues. This validation exercise culminated in a refinement of the original set of issues[4], and these are presented in the following section.

4 Refinements concerned largely the grouping of results into common issues and the clarification of terminology. To a lesser extent, the relative importance of some problems was also affected.

Issue	Concern	Recommendations	Difficulties Implementing Recommendations Successfully
Package Consideration	Should packages be considered in the requirements engineering process?	• packages should, in general, be considered in the requirements engineering process except in the case of unprecedented systems	• organizational politics and the 'Not Invented Here' syndrome may hinder package consideration • when resource constraints exist, packages are considered with a view to compromising user requirements to avoid package adaptation
Managing the Level of Detail of Functional Process Models	How much functional process modeling is necessary in the requirements engineering phase?	• consider the extent of package adaptation necessary and the level of uncertainty when deciding when to stop the modeling activity	• inexperienced analysts tend to produce models that are too detailed • the greater the level of control in the organization, the greater the pressure to produce too much detail • when there are resource constraints, there is pressure to cut down on the modeling activity (i.e., too little detail)
Examining the Current System	Should the current system be examined? If so, how detailed should this examination be?	• if there is no current system, then there is nothing to examine, otherwise examination of the current system should be done • consider the following 4 criteria when determining when to stop: models of the existing system should be detailed enough to allow a diagnosis, more effort is necessary for larger legacy systems, it may be possible to reuse models of the current system, junior analysts may leam about the application domain if they model the current system	• inexperienced analysts may model the current system in too much detail • some organizations are reluctant to spend more money on the current system and its examination • the higher the level of control of the organization, the greater the pressure to examine the current system in too much detail
User Participation	Should users participate in the requirements engineering process? If so, how could they participate and which users should participate?	• users should always participate in the requirements engineering process • the mechanisms that can be used to promote user participation include face-to-face meetings, workshops with users, communication with users through electronic mail and/or conference calls, liaison groups (either based in the user organization or IS organization), off-site visits by senior analysts to user departments, users sitting on a project steering committee, users reviewing and approving documentation items, users performing some of the requirements engineering activities themselves (e.g., benefits analysis), users paying for the project out of their own budgets and users being evaluated on the overall success of the systems development effort • the appropriate skills of participating users (especially the principal user) include the list presented in Section 3.4	• the most desirable users may not be available to participate • there is a history of lack of cooperation between user departments and IS • there is open hostility between the user and IS organizations
Managing Uncertainty	How can uncertainty be alleviated or dealt with in the requirements engineering process?	• assign appropriately skilled people to analyst and architect positions, in particular the lead architect. The necessary skill set includes the list presented in Section 3.5 • assign appropriately skilled users to participate in the requirements engineering process, in particular the principal user. The necessary skill set includes the list presented in Section 3.4 • if uncertainty is very high, then consider constructing a prototype	• inappropriately skilled analysts and users are assigned to these positions due to the common practice of assigning people based on availability rather than capability • a prototyping effort is not managed properly (e.g., not managing user expectations and assigning staff inexperienced with the prototyping tool to develop the prototype)
Benefits of CASE Tools	How can benefits be gained from CASE tools in the RE process?	• an infrastructure must be in place to support the implementation of a CASE tool (e.g., training on methods and tool, and a CASE support group) • an organization must be willing and able to invest in putting such an infrastructure in place	• high and unrealistic expectations (e.g., immediate pay-off) by management about the benefits of CASE tools • lack of funding for the implementation of CASE tools
Project Management Capability	What are the necessary skills of a project manager?	• always assign a project manager of high capability to the requirements engineering phase. The skill set of a project manager includes the list presented in Section 3.7	• people are assigned to project management positions based on their availability rather than on their project management capabilities • the career track leads inappropriately skilled people to project management positions

Figure 3: Summary of the seven requirements engineering process issues.

3 Results

The results of this study indicate that there are seven issues of greatest concern in requirements engineering practice. These issues are discussed below in terms of the problems they represent, how these problems are addressed successfully in practice, and impediments to the implementation of such good practices. An overall summary of the issues is shown in Figure 3. This figure accompanies the discussion below.

3.1 Package Consideration

Package solutions (these are also sometimes referred to as Components Off The Shelf or COTS) may be considered during the requirements engineering process for one or a combination of three reasons. The first is that packages may serve as existing system surrogates; the second is that a package solution may form the whole or part of the recommended solution; and the third is that package consideration may boost the users' confidence with the recommended solution.

While serving as existing system surrogates, packages aid in the definition of user needs. Through their interactions with a package, users can more easily identify the functions they need and desire, and define the functions that are not available in the package. Davis [8] has also previously suggested that a package may be used as an anchor and the new system requirements can be adjusted from it. Furthermore, when used as existing system surrogates, packages have acted as a source of ideas to analysts in terms of user-interface design, and in terms of how to structure the functions and data in their models. These models are ultimately used as part of the architecture if a software system is to be developed.

When considered for their potential as the whole of or part of the solution, packages are perceived to be less costly business solutions in terms of funds and time to implement. This is particularly attractive when project funding is low or when the delivery schedule is tight.

There was strong consensus that packages should also be considered so as to convince the users that alternative solutions (alternative to building a system from scratch) were considered, even if they do not serve as existing system surrogates or become (part of) the solution. Furthermore, packages have been used by analysts to convince users of the feasibility of a particular solution, and hence making them more comfortable with it. For example, in one case where the users were not convinced about the viability of expert system technology, seeing a package that utilizes an expert system convinced them that the technology was mature enough for their purposes.

The above paragraphs indicate that some benefits would probably be gained through package consideration. Thus, the general recommendation has been that packages should be considered in the requirements engineering process. In terms of costs, the requirements engineering phase typically consumes 10-15% of total system development effort (this is also supported in the literature [7][20]), and package consideration is only a small fraction of the requirements engineering phase. Thus the total effort that is consumed in package consideration is relatively small.

In the case of unprecedented systems (with respect to functionality and/or technology), however, appropriate packages may not exist in the market. Therefore, if it is evident that no packages exist, or that an extensive package adaptation effort is necessary to make use of existing packages, then package consideration would not necessarily be cost-effective. For example, in one case packages were not considered because it was very obvious that the software system was very close to the state-of-the-art in Information Technology.

Two problems were identified that frequently inhibited package consideration. The first problem is organizational politics. One project manager with extensive experience in government environments elaborates:

> *In government it is less likely [that packages are considered]. They are not so concerned about cost, so they will not accept a package to save money; there are political reasons for budgeting. The basic government attitude is that they don't look at the value of things, it's not their money.*

Another example is when an organization just purchased a mainframe and there existed a PC-based package worthy of consideration:

> *Someone up top told them that because they spent $20M on a mainframe, they had to use it. They were reluctant to look at any PC-based solutions. No one wants to rock the boat!*

The second problem is the "not invented here" syndrome, whereby analysts believe that their situation is unique and they *want* to construct a system from scratch.

3.2 Managing the Level of Detail of Functional Process Models

The main purposes of functional process modeling are to gain an understanding of what the software system will do, what data it will consume and produce, and its interactions with other manual and automated systems. Such an understanding is employed in performing the cost/benefits analysis and in formulating and comparing alternative solutions.

Modeling is still necessary even when package solutions are considered. In such a situation, one would want to do three things. First, to identify a suitable set of packages for consideration. Second, to determine the extent to which package adaptation is necessary. Third, to compare development costs with package adaptation costs in order to make a "build" or "buy" decision. This implies that, at least, a minimal amount of modeling is essential to identify packages. Also, if a package is destined to become the whole of or part of the solution, then a minimal amount of modeling is necessary to determine the extent to which package adaptation is required. This is consistent with both, the prescriptive and empirical literature, which promotes modeling as a prerequisite to package consideration [4][24][25].

Therefore, it is always necessary to construct functional process models in the requirements engineering process. The question, of course, is how much modeling is necessary (or how much detail) in the requirements engineering phase? An exiguous level of detail inhibits the ability to perform an adequate cost/benefits analysis, and to formulate and compare alternative solutions. An excessive level of detail is not cost-effective.

Considerable variation in the level of detail of functional process models was observed in the cases studied. This was evident from document inspections, where some requirements engineering phases only produced a one page context diagram, and others produced a few hundred pages of data flow diagrams. In this study, the level of appropriate detail was found to be contingent upon the extent to which package adaptation is necessary and on the amount of uncertainty.

Package adaptation only becomes important when a package is considered as part of or the whole of the recommended solution. Package adaptation was found to be driven primarily by two factors: the need for interfaces with other systems and the extent to which the package covers the system requirements. The greater the extent of adaptation, the more detailed the models should be. For example, detailed models will facilitate a better "build" or "buy" decision since they allow for better estimates of (the package adaptation and implementation) costs and (package functionality) benefits, as well as giving the users a better understanding and awareness of how the package will fit in with their business needs. A good example of the benefits of models is described below by the project manager:

> *The users sometimes focused too much on the look of the packages and not on whether the functionality is covered. In the [company's name] case, after the first demo, the users were very happy with the package, they wanted it right now, and bought it the day after. We just showed them that the functionality that was covered in the package represents only 20% of the cost of developing the system. What do you do with the 80% that is left?*

In the above example, the model allowed the analysts to estimate how much of the desired functionality was covered by the package. This particular package was later abandoned.

Furthermore, when the context is characterized as *relatively* uncertain, then detailed modeling is one mechanism that would facilitate learning about the business process and the technology. However, there are dangers in the employment of modeling as the primary method for understanding within a *highly* uncertain context. For example, if the users are not familiar with computerized systems or they do not understand models, then they will be unable to provide adequate feedback and validation of the models, and therefore there is no point in presenting them with models. Under such conditions, resorting to prototyping as the primary method for determining requirements is preferable.

Functional process modeling is one of the most effort consuming activities in the requirements engineering process. Hence, doing excessive modeling (where excessive means that the added detail provides little value for understanding the problem and the solution, comparing alternative solutions, and for performing a cost/benefits analysis) leads to a lower likelihood of a highly cost-effective requirements engineering process. It should also be recalled that, just because the requirements engineering phase was started, it is not predestined that a system will be developed or a package purchased and adapted. One of the main points of the requirements engineering process is to make a go/no-go business decision as to whether to continue or stop the project after the requirements engineering phase; thus the project may be stopped at the end of the requirements engineering phase. This provides further fuel to the argument that excessive detail is not cost-effective since the project may be terminated at the end of the requirements engineering phase.

Two factors were frequently identified as leading to highly detailed functional process models. First, inexperienced analysts tend to spend a disproportionate effort on model construction. Judging by the interviewee comments on this issue, it seems to be one of the most critical problems in the requirements engineering process. One project manager elaborates:

They fall in love with modeling. They lack experience and are too junior in a business sense. They produce clever models that are as close to code as they can get them. They have too narrow a vision of what the models are or what they are intended to do. [...] Most of the time you have to hold back [junior analysts] from going into details. You have to convince them that the extra effort to put details will not bring much benefits. You have to make a business decision and make [junior analysts] move from one deliverable to another, even if they think their baby is not beautiful enough.

The level of detail of models is a project management issue, and it is up to the project manager to terminate modeling. To achieve this, an experienced project manager is necessary to know when to stop.

Second, in some organizations, the level of control is relatively high. This implies that they want much detail in their models. In some cases, it is the way the organization operates, the organization is a bureaucracy. For example, in one case the requirements engineering team had to build detailed models of all the alternative solutions, to the extent that they could actually implement all of them if they so wished. Unfortunately, no particular recommendations for alleviating detail in such a situation could be identified.

3.3 Examining the Current System

The main purpose of examining the current system is to understand it and then to assess it. The former includes understanding what the existing system does, who are the parties involved with the current system (e.g., users, suppliers, clients etc.), and why it operates in the current way. This understanding would always be translated into an assessment of the current system. The assessment includes the identification of the strengths and weaknesses of the current system, and why the current system is not adequate and requires replacement or upgrading.

Understanding and assessing the current system provides the basis for a business case to construct a new system. Furthermore, where some of the analysts are not experienced in the application domain, modeling the existing system is considered a good way to gain such experience. Moreover, it was observed that models of the existing system can be reused in developing the new system where there is a similarity in functionality. However, some users and/or Information Systems (IS) organizations prefer not to examine the current system due to the attitude that *"they have already paid for the current system and do not want to spend any more money on it".*

In the cases studied, the existing systems were inadequate and were being replaced or upgraded because they were inefficient or ineffective, or because they did not support evolving organizational strategies and objectives. These inadequacies usually concerned both the IS organization, as well as the user organization. Example IS organization inadequacies are: the existing system is difficult to maintain due to unpredictable ripple effects in making changes, the existing system is not well documented and turnover or cutbacks are eliminating IS knowledge about the system, and the IS organization wants to take advantage of new technology. Example user organization inadequacies are: the existing system supports/automates an inefficient business process, the existing system has unreliable or inaccurate outputs which results in extra manual effort to check and correct its outputs before they can be used, and the existing system requires the input of too much manually collected data to be useful.

In an ideal world, existing systems would be well documented, and functional and data models for these systems should be readily available at the beginning of the requirements engineering process. However, in many of the cases studied, existing system documentation was lacking. In the remaining cases, models of the existing system were available. This availability was due to the IS organization keeping documentation current, or because the organization was going through a business process reengineering exercise. In the latter situation, existing business processes and their supporting Information Systems are modeled as part of the reengineering approach.

In cases where existing systems had to be modeled during the requirements engineering phase, the same problems with the level of detail of models mentioned above arose. These are the lack of experience of analysts and the high level of control of some organizations.

3.4 User Participation

The primary benefits of user participation were perceived to be: less rework of the documentation items since the users took part in their generation and approval; greater fit of the recommended solution to the organization; participation helps reduce political conflicts among users; participation ensures early user "buy-in" into the system; and participation reduces the likelihood of sabotage or users *"trying to defeat the [recommended solution] or the system"*. Furthermore, since a critical decision has to be made after the cost/benefits analysis, user participation ensures that they are convinced of the estimated costs and benefits of the recommended solution.

While it was evident that it is one of the most important factors that contribute to the success of the requirements engineering phase, a number of factors inhibited user participation. The most common ones were a history of lack of cooperation between IS groups and users, open hostility between project members and users, and unavailability of users.

When there is a history of lack of cooperation between users and IS, it is surprisingly difficult to bring about a change towards greater user participation. In one case the project manager explains the situation:

> *The IS people have the policy of building products they feel the users will need and then try to market them to the users. Business people are not happy about that. [...] The IS people think they understand the business so well, better than the user, and so they try to force their ideas on the users. IS may understand the business, but their perceptions of the business objectives may be different from the users.*

Open hostility occurred when there were severe personality clashes between a senior IS person involved in the requirements engineering process and a senior user. In one case the personality clash was severe enough that the user refused to validate the documentation, with excuses akin to *"don't you know what you are doing, why do you want me to validate it?"* A lack of trust between the two functions also inhibits user participation.

Furthermore, the most desirable users are usually the least available users. Hence, it is frequently difficult to get their time commitment to the requirements engineering phase. This is a particularly arduous issue to resolve in small companies, and with users in organizations experiencing rapid growth. In addition, participation in the requirements engineering phase entails an increase in the users' workload and responsibility; if they are not appropriately compensated, then this may lead to resentment and eventually lack of cooperation.

The most desirable users have a skill set that includes the following:

- familiar and experienced with computerized systems;
- good knowledge of the application domain, the business processes, and the needs of the user organization;
- good knowledge of systems development processes (especially the requirements engineering process and modeling techniques) and their deliverables;
- good knowledge of Information System implementation planning and management, and change management concepts; and
- able to deal with people and have good communication skills.

In addition, it is desirable that participating users have influence by authority within the user organization. This is critical because, in most situations, the participating users must subsequently go back to their respective departments and ensure the commitment and buy-in of the remaining end-users. Moreover, if it is necessary to pull user resources, especially from other departments, the participating users must be high enough in the organizational hierarchy to do it. If the recommended solution encompasses a restructuring of the business, low level managers usually do not have the power or capacity to decide at that level, especially if other departments are concerned. A senior participating user also has the authority (and one of the few in an organization) to implement an Information System that requires organizational changes in the user organization and to secure the resources to do so; getting early participation and commitment from such a user is a prerequisite to a successful implementation.

3.5 Managing Uncertainty

Uncertainty is one of the most acknowledged situational variables in the software engineering literature. It has been recommended that uncertainty should be considered when deciding whether to prototype [13], when deciding what requirements engineering activities to perform [8], and when deciding which general systems development life cycle models to follow [2].

Uncertainty denotes *the difference between the amount of knowledge that is required and that is available about the problem and solution domains.* The greater the uncertainty, the greater the amount of changes to the requirements engineering documentation.

Four dimensions to uncertainty have been identified: user uncertainty, analyst uncertainty, application uncertainty, and utilizing system uncertainty. User uncertainty is concerned with the ability of the users participating in the requirements engineering phase to specify requirements. Analyst uncertainty is concerned with the analysts' ability to elicit requirements and use these to formulate a recommended solution. Application uncertainty concerns the software system to be developed and its associated technology. Utilizing system uncertainty concerns the business process to be automated or supported by the software system.

Users who contribute toward uncertainty will generally not understand models and will not be able to get the right information. The requirements engineering team will not be building the right system. This situation is more common in small and rapidly growing organizations where the most capable users are not available. One project manager describes the situation in an organization experiencing rapid growth:

> *There was very little discipline in the management structure because the organization was built in no time, and before you can build [discipline] the company doubled in size. Everybody in there has been promoted about three levels beyond their level of competence as managers. Nobody knows the business.*

Lack of knowledge of the business process by the user cannot be overcome except by either replacing the user, or if the senior analyst has extensive knowledge of the business.

Capable analysts are able to elicit requirements from the users and express these as models reflecting a solution. A capable analyst will have a skill set including the following:

- good knowledge of the application domain and the business processes of the user organization;
- good knowledge of the hardware and software technology to be used in the Information System;
- able to deal with people, and has good interviewing and verbal communication skills;
- good knowledge of and experience with conceptual and functional modeling;
- good knowledge of and experience with the requirements engineering process and its deliverables;
- good knowledge of and experience in conducting a cost/benefits analysis;
- good knowledge and experience of issues related to system implementation and organizational change; and
- good knowledge of and experience with software package analysis and evaluation.

Inexperienced analysts are considered not to be able to ask the right questions or the "tough" questions to executives, because they never did. One commonly encountered problem that leads to incapable analysts taking part in the requirement engineering phase has been careless assignment and hiring practices. It is common for roles in the requirements engineering phase to be instantiated by individuals who are available rather than those who are capable of doing the job. The following comments exemplify the problem of hiring practices:

> *You have companies who feel that Information Systems can be built without any specific skills. Many organizations do not understand [the need for specific roles]. They feel that IS people can do any role. [...] The company does not necessarily recruit from the domain. In fact, the senior members complain that the recruiting habits are "really bad" and that they were hiring people that were not qualified. This was because Human Resources usually did the hiring. The problems were manifested in that there was a high turnover rate at the junior level.*

Also, uncertainty was increased when organizations were rapidly adopting new technologies in an attempt to gain a competitive advantage. Such new technologies included object oriented analysis and design techniques, client/server architectures, and sophisticated GUI's. Furthermore, unstable businesses and users that are unable to commit to the decisions they make provide the analysts with a moving target, and hence contribute to uncertainty.

One solution recommended for managing *high* uncertainty is prototyping. The primary motivations for building a prototype were: eliciting requirements, validating requirements, and determining the feasibility of particular solutions. This is concordant with the recommendations in the literature [1][13]. Different kinds of prototypes were also constructed. These ranged from the simple screen mock-ups, internal prototypes used only by analysts to test new technologies, to full scale prototypes including a model office and actors to play the roles for demonstrating the manual and automated processes and their interactions.

When uncertainty is low, *"prototyping never makes things bad, it only costs more"*. However, even when uncertainty is low, prototyping has the advantage of making the users more comfortable with the solution; it provides them with greater visibility. This, of course, is even more beneficial when uncertainty is high. In addition, as Tozer [23] and Alavi [1] note, prototyping can lead to improved relationships between the users and the analysts.

A classic example where prototyping was critical is summarized below by one of the team members:

> *Prototyping was started early because the system was too much state-of-the-art, not like designing a payroll system, it was something that does not exist, with very high ergonomic requirements. It was very multidisciplinary, with many different types of users. Each of these disciplines has their own way of working, so it is not just mechanizing something that was never mechanized before, but it is also to standardize the way the information is presented. The system is multi-media, it has voice, text, and image. [...] The practical users on the day-to-day job wanted to criticize every button on the screens, they're saying there's too many screens, the navigation is too long, can you combine this screen with that? They're really in the middle of what's going to change their way of working. They don't even like unreal data, it has to make sense.*

This example highlights two facts. First, when technology is new and too fragmented, when the user interface is complex, and when there are many different kinds of users, a prototype is useful for assessing the feasibility of the solution, and ensures greater user acceptance of the emerging solution. It also enables the analysts to learn about the different implementation alternatives. The second fact is the importance of constructing prototypes of acceptable quality using real data, a point previously made by Tate [22].

When users are not familiar with Information Technology, there is the contention among lead architects that it is easier to prototype than to teach the users about data and process modeling (for the users to understand the system and to perform validation). Furthermore, some project managers assert that in a number of cases the users were too afraid to admit that they do not understand the models, and hence may provide erroneous feedback. When in doubt, it is therefore more prudent to present to the users a prototype which they can more easily relate to. In situations where there are users representing multiple departments in the organization, a prototype may also help in attaining a consensus on the recommended solution.

Three prototype dangers were also highlighted. The first one concerns users with little Information Technology experience. A typical comment is:

> *Users can build misconceptions about software development [if you prototype]. For example, software is easy to build and development has quick turnaround, or systems don't work and are full of bugs.*

While some authors discount such fears about users' misconceptions as something that does not occur in practice [23], others recognize it as a real danger [22][1]. The issue here is that of managing user expectations during prototyping efforts.

The second danger is that of the capability of the person building the prototype. If such a person is not an expert in the prototyping tool, then prototyping may not achieve its objectives. Finally, there is the danger of the prototyping tool itself. One lead architect elaborates:

> *Prototyping tools have limitations. They are good with simple applications. When you exceed the boundaries of the tool and you have to modify the generated code, then this results in much greater effort. It really depends on the tool and its limitations.*

This corroborates the dangers identified by Tozer [23] who states that *"No 4GL meets 100% of the requirements for building systems such as complex corporate systems which may have to fit in with existing mainframe systems"*. However, she goes on to add that, from her experiences, modifying the generated code is not a problem since the tool that was used generated readable and structured code, and the personnel constructing the prototype have extensive experience with the tool.

3.6 Benefits of CASE Tools

A prerequisite to attaining significant benefits from a CASE tool is to implement the tool. In the requirements engineering processes we studied, the implementation of CASE tools was only effective in organizations that had put in place an infrastructure to support implementation. This implies that at least the requirements engineering process was well defined, the analysts received training on the CASE tool, and there exists some form of CASE tool support organization. The literature provides evidence that to implement a CASE tool, at a minimum a methodology must be in place [5][27]. In organizations that did not have an adequate infrastructure, CASE tools were bought but were not used, CASE tools were purchased and were being used as drawing tools, other non-CASE tools were used solely for the purpose of drawing models, and/or the analysts prepared the models by hand.

There was no particular trend noticed in terms of funding sources for the implementation of CASE. In some organizations these came out of project budgets, while in others funding was centralized and CASE was considered an IS function asset for use by all projects.

However, where the source of funding imposed resource constraints, there was no incident where CASE was implemented. Huff [12] and Hayley and Layman [11] report on the high dollar and time costs associated with CASE implementation. Hence, resource constrained projects or IS organizations simply cannot afford the expense and/or time to implement CASE. A typical situation is the organization being unable to purchase enough copies of the tool or purchase enough workstations for all the participating analysts. In such requirements engineering phases, the CASE tool was not used. Zagorsky [27] also pinpointed the lack of availability of the CASE tool to all team members as an obstacle to its implementation.

Interestingly, many project managers in small IS organizations, usually characterized by low funding, emphasized the need for CASE tools (or better CASE tools) and the need for more automation. This emphasis is based on the perception that more automation, by itself, will increase productivity. This perception has been further reinforced by the marketing efforts of tool vendors and (independent and corporate) consultants touting the productivity gains of CASE.

These predicted gains not only seem attractive to small IS organizations with low funding, but also to other larger organizations. It was repeatedly stated by the interviewees that IS organizations within companies facing strong competition in their traditional markets, where there is a large backlog of systems to be developed, where a large percentage of effort is expended evolving legacy systems, where there is a lack of job security, and/or where top IS management is under pressure to deliver results quickly, there is a strong perception that CASE is a critical key to solving productivity problems rapidly. Unfortunately, the literature does not support such expectations. Productivity gains from CASE are one of the *long-term* benefits of CASE *implementation* [11][10][16].

3.7 Project Management Capability

The project manager's role in the requirements engineering process includes planning and control, and managing the people in the requirements engineering team. The project manager also has to manage relationships with the users. This is important since, as one interviewee expressed it, *"if you have a problem with a user group, they'll remember it for a long time"*. Furthermore, the project manager should have a good knowledge of the requirements engineering process, because s/he has to tell team members what to do and to make sure that they are doing what they are supposed to be doing *"so as not to be taken advantage of"*.

One of the more important activities that a project manager must perform is estimating and obtaining adequate resources (time and funding) for the requirements engineering process. The majority of estimates were based on the experience of the project manager and estimation grids for typical requirements engineering phases.

A few of the project managers interviewed conceded that there exists pressure to underestimate resources, especially when the IS and/or user organization is not benefits oriented (i.e., decisions are based mainly on costs rather than on a cost/benefits ratio). This leads to resource constraints in the requirements engineering process. Example sources of pressure to underestimate are: political reasons (for example the politics of budgeting), and some IS organizations are simply not accustomed to making large investments in IS. Schedule constraints are also created by hard deadlines that are decided outside the IS function. For example, when one utility had to change its billing process, the system that supports billing had to be in place at a specific date because commitments were already made to customers. Also, in government agencies, when there is new legislation, for example new taxes, the system that supports it must be operational in time for when the legislation comes into effect.

The consequences of resource constraints (unrealistic budgets and schedules) is that they force the requirements engineering team to be *"innovative"*. This can take the form of cutting down on process and data modeling activities, to the detriment of the team's ability to formulate a realistic business case. Also, package solutions are favored given the resource constraints, but the user requirements are compromised to avoid the costs of adaptation. Furthermore, project management activities are curtailed (having only part-time project managers and reduce progress reporting).

Experienced project managers understand the need to schedule time for decision making during the requirements engineering phase. Failure to do so may result in gross underestimates of resources. Organizations vary widely in the speed with which they can make decisions. In some organizations the decision making process is consensual, or every decision requires multiple layers of approval. One project manager commented about a government agency where no one actually makes a decision, it just emerges. This way no one can be held responsible for a particular decision. It is thus important for the project manager to estimate accurately the delays due to decision making lags and include these in the project plan.

Project managers lacking capability are assigned to this role for two commonly cited reasons. The first is the general practice of assigning people to roles based on their availability not capability. The second is the career track that leads people to project management positions:

> *In many cases, people are promoted to the project manager role based on years with the company, not based on capability. They don't understand the importance of project management, so they assign anyone to project management, usually an analyst. They do not understand that a good analyst does not [always] make a good project manager.*

The skill set of a capable project manager should include the following:

- able to use automated project management tools;
- has good estimation capabilities;
- able to decompose and monitor activities;
- able to deal with people, and has good negotiation and verbal communication skills;
- good knowledge of the system development process (in particular the requirements engineering process) and its deliverables; and
- has a good understanding of emerging requirements engineering technologies and tools.

4 Conclusions

It is evident from the foregoing discussion that, as well as technical issues, non-technical issues are of importance in the requirements engineering process in IS development. Specifically, the problems faced concern performing the appropriate activities (e.g., whether to consider packages and whether to prototype), deciding when to stop particular activities (e.g, functional process modeling and examination of the current system), securing an adequate level of user participation, and selecting capable personnel to instantiate key roles in the requirements engineering phase (e.g, the project manager, lead architect, and principal user). Furthermore, the main problem with attaining benefits from CASE tools were implementation problems, like training the analysts on the tool and the underlying methodology.

Finding that non-technical issues are important supports the beliefs of other researchers in software process improvement [18], and is concordant with the results of an earlier study of requirements engineering practices where the authors state [15] : *"It was strikingly obvious that many of the problems faced by the projects we interviewed were organizational and non-technical. Moreover, projects tend to adopt organizational solutions to most of their problems, whether technical or non-technical"*. In addition, the results of this study complement existing works by identifying impediments to the implementation of what are considered to be good requirements engineering practices.

Further confidence in our results would be obtained if other researchers replicate this study or conduct similar studies using a different sample of projects and informants. Moreover, the employment of alternative empirical research methods (e.g., surveys and quasi-experiments) in investigating requirements engineering practices would ascertain the extent to which our results are affected by the case study approach that we followed. Such investigations could, for instance, utilize our findings as hypotheses to be tested.

Many of the identified problems are of concern mainly during the planning of the requirements engineering process (e.g., deciding whether to prototype and whether to consider package solutions). Future research should therefore also focus on providing automated and non-automated tools and methods that support decision making during the planning activities. For example, a tool that would assist in deciding what activities to perform and for how long. This would be a requirements engineering process design or customization assistance tool. When the prospect of such a tool was proposed to the interviewees, there was high interest and strong agreement that it would be beneficial, not only to assist decision making, but also for training project managers on how to plan requirements engineering processes.

Another high leverage area of research that is identifiable from this study is the development of a method or instrument to assess the capability of key requirements engineering personnel. Given that previous studies of requirements engineering practices have found that personnel capability and knowledge have a nontrivial impact on the success of the requirements engineering process [6][15], such assessments would be invaluable for *staffing* and *training* purposes.

Acknowledgements

The authors wish to thank Lionel Briand for his comments on an earlier version of this paper.

References

[1] M. Alavi. "An assessment of the prototyping approach to information systems development". In *Communications of the ACM*, 27(6):556-563, June 1984.

[2] L. Alexander and A. Davis. "Criteria for selecting software process models". In *Proceedings of the 15th International IEEE COMPSAC*, pages 521-528, 1991.

[3] V. Basili and B. Perricone. "Software errors and complexity: An empirical investigation". In *Communications of the ACM*, 27(1):42-52, January 1984.

[4] M. Bryce and T. Bryce. "Make or buy software?". In *Journal of Systems Management*, pages 6-11, August 1987.

[5] K. Culver-Lozo and V. Glezman. "Experiences applying methods supported by CASE tools". In *AT&T Technical Journal*, pages 20-26, November/December 1992.

[6] B. Curtis, H. Krasner, and N. Iscoe. "A field study of the software design process for large systems". In *Communications of the ACM*, 31(11):1268-1286, November 1988.

[7] E. Daly. "Management of software development". In *IEEE Transactions on Software Engineering*, SE-3(3):229-242, May 1977.

[8] G. Davis. "Strategies for information requirements determination". In *IBM Systems Journal*, 21(1):4-31, 1982.

[9] A. Endres. "An analysis of errors and their causes in system programs". In *IEEE Transactions on Software Engineering*, SE-1(2):140-149, June 1975.

[10] M. Gibson, C. Snyder and K. Rainer Jr.. "CASE: Clarifying common misconceptions". In *Journal of Systems Management*, pages 12-19, May 1989.

[11] K. Hayley and T. Lyman. "The realities of CASE". In *Journal of Information Systems Management*, pages 18-23, Summer 1990.

[12] C. Huff. "Elements of a realistic CASE tool adoption budget". In *Communications of the ACM*, 35(4):45-54, April 1992.

[13] M. Janson and L. Smith. "Prototyping for systems development: A critical appraisal". In *MIS Quarterly*, pages 305-316, December 1985.

[14] C. Jones. "Assessment and control of software risks". Prentice Hall, 1994.

[15] M. Lubars, C. Potts, and C. Richter. "A review of the state of the practice in requirements modeling". In *Proceedings of the IEEE International Symposium on Requirements Engineering*, pages 2-14, January 1993.

[16] C. Kemerer. "How the learning curve affects CASE tool adoption". In *IEEE Software*, pages 23-28, May 1992.

[17] D. Perry. "Human prospects of process design". In *Proceedings of the Seventh International Software Process Workshop*, pages 22-27, 1992.

[18] D. Perry, N. Staudenmayer, and L. Votta. "People, organizations, and process improvement". In *IEEE Software*, pages 36-45, July 1994.

[19] R. Rubey, J. Dana and P. Biche. "Quantitative aspects of software validation". In *IEEE Transactions on Software Engineering*, SE-1(2):150-155, June 1975.

[20] Software Engineering Laboratory. "Software Engineering Laboratory (SEL) relationships, models, and measurement rules". *Technical Report SEL-91-001*, NASA Goddard Space Flight Center, February 1991.

[21] E. Swanson and C. Beath. "The use of case study data in software management research". In *Journal of Systems and Software*, 8(1):63-71, January 1988.

[22] G. Tate. "Prototyping: Helping to build the right software". In *Information and Software Technology*, 32(4):237-244, May 1990.

[23] J. Tozer. "Prototyping as a system development methodology: Opportunities and pitfalls". In *Information and Software Technology*, 29(5):265-269, June 1987.

[24] S. Vallabhaneni. "Auditing software development: A manual with case studies". John Wiley & Sons, 1990.

[25] M. Wu. "Selecting the right software application package". In *Journal of Systems Management*, pages 28-32, September 1990.

[26] R. Yin. "Case study research, design and methods". Sage Publications, 1984.

[27] C. Zagorsky. "Case study: Managing the change to CASE". In *Journal of Information Systems Management*, pages 24-32, Summer 1990.