2008

A Software Release Planning Methodology for Developers

Jennifer Jewer
University of Waterloo, jljewer@uwaterloo.ca

Kenneth N. McKay
University of Waterloo, kmckay@ist.uwaterloo.ca

A Software Release Planning Methodology for Developers

Jennifer Jewer
Department of Management Sciences
University of Waterloo
jljewer@uwaterloo.ca

Kenneth N. McKay
Department of Management Sciences
University of Waterloo
kmckay@ist.uwaterloo.ca

ABSTRACT

The process of assigning requirements to releases is difficult and release planning methodologies are becoming increasingly complex in an attempt to take into account different stakeholder perspectives and criteria. However, there is a need to focus on understanding the criteria used in release planning in order for the methodologies to adequately support this process. This paper specifies the criteria for software developers to use when planning releases by operationalizing the risk criterion and enhancing the interdependency criterion. A controlled experiment was conducted to evaluate release plans created with this new versus an informal methodology. The results imply that the criteria specified in this new methodology are in fact used by developers in the creation of releases. After further testing, this methodology could prove beneficial in helping developers create release plans for large numbers of requirements.

Keywords

Incremental software development, software release, requirements prioritization, decision support, risk management.

INTRODUCTION

Many researchers have investigated ways of minimizing software project risks through systems development methodologies. One such methodology has been incremental development because research suggests that delivering systems incrementally with smaller functioning releases is less risky than implementing a complete system after a long development time (Beck, 1999; Greer and Ruhe, 2004). In fact, small project size has been identified as one of the most significant factors in project success (Standish Group, 1995).

In order to develop software incrementally the requirements are assigned to different increments and each increment is released in stages. This assignment of requirements to different increments is known as release planning (RP) and researchers agree that it is crucial to incremental software development (e.g. Carlshamre, Sandahl, Lindvall, Regnell, and Natt och Dag, 2001; Karlsson, Thelin, Regnell, Berander and Wohlin, 2007). However, RP is a difficult process (Aurum and Wohlin, 2003; Karlsson and Ryan, 1997; Lubars, Potts and Richter, 1993) and often done informally in organizations based upon individual experience and tacit knowledge (Karlsson and Ryan, 1997; Lehtola, Kauppinen and Kujala, 2004). This informal RP could cause misunderstandings (Lehtola et al., 2004), and if planning is done badly it could increase risks such as budget overruns and a loss of market share (Wiegers, 1999). However, "[a]lthough requirements prioritization is recognized as an important area, few research papers aim at finding superior prioritization techniques that are accurate and usable," (Karlsson et al., 2007, p.8). In a study comparing two different RP methodologies it was found that the resulting priority order was incorrect and that users sometimes did not trust or have confidence in the results of these methods (Lehtola and Kauppinen, 2004).

Given this gap in the literature and the apparent challenges with the use of these methodologies in practice, this paper focuses on specifying criteria that software developers may use when RP. The RP methodologies in the literature acknowledge different knowledge areas of stakeholders and suggest that one group of stakeholders consisting of users and/or customers should rank the requirements on criteria such as value and business relevance, and that another group of stakeholders consisting of developers should rank the requirements on other, more technical, criteria. This paper introduces further guidance on what these technical criteria are in order to facilitate RP. Specifically, the methodology introduced this paper operationalizes the risk criterion and enhances the interdependency criterion. Risk is part of many of the other RP methodologies; however, this paper introduces an operationalization of risk that is grounded in literature to provide guidance to developers when assessing the risk of a requirement in order to create release plans. The methodology enhances the interdependency criterion by including guidance on the assessment of functional interdependencies between requirements.

There is a need to focus on understanding the criteria used in RP, rather than relying on the developers to make their own assessments of criteria such as risk and interdependencies, in order for the methodologies to adequately support this process (Wohlin and Aurum, 2005). A review of the literature indicated that no RP methodologies have specifically looked at the criteria that developers use when assigning requirements to releases, therefore, the methodology presented in this paper is a

first step towards filling this gap. This study offers some preliminary findings based on data from a controlled experiment with developers using an informal versus this new methodology.

Following this introduction, an evaluation of the RP methodologies is presented, and then the new RP methodology for developers is outlined and explained. Finally, the experimental design and findings are discussed and the paper concludes with a summary of the significance of the research findings, and identification of the limitations and areas for future research.

EVALUATION OF RELEASE PLANNING METHODOLOGIES

The RP methodologies are becoming increasingly complex as they attempt to take into account different stakeholder perspectives and criteria. These methodologies are briefly discussed below according to the (1) stakeholders, (2) criteria, and (3) processes to assign requirements to releases.

First, all the methodologies stress the importance of including input from multiple stakeholders in the planning of releases. Most methodologies include at least representatives from business and software development; however, other methodologies also include stakeholders, such as: different types of users of the system, customers, investors and shareholders.

Second, the methodologies also suggest multiple criteria upon which to base the assignment of requirements to releases; however, no study providing conclusive evidence that one set of criteria is superior to the others was found in the literature. In an empirical evaluation of two methodologies it was found that participants found the resulting priority order incorrect and some participants did not trust the results (Lehtola and Kauppinen, 2004). They also found that the practitioners did not know on what information they should base their evaluations of value or cost. This is an important finding because "prioritization results are never better than the raw data inserted," (Lehtola and Kauppinen, 2004, p.168). Therefore, it is essential that the user of the RP methodology understand the criteria and know how to assign values to requirements based on these criteria. However, it is difficult to determine what criteria should be included in a RP model because the criteria used without the aid of a RP methodology are often not stated and are instead implicitly used by the decision-makers (Wohlin and Aurum, 2005) and decision-makers may not even be explicitly aware of which criteria they take into account (Lehtola et al., 2004). Some studies provide no guidance and rely on developers to know how to rank the requirements. For example, the Incremental Funding Method (Denne and Cleland-Huang, 2004) proposes that developers estimate the cost and effort involved in developing each release; however, it offers no guidance on how these estimates are to be developed. Finally, it is also important to consider interdependencies between the requirements in addition to their priorities when planning releases (Carlshamrea et al, 2001; Karlsson and Ryan, 1997). Carlshamrea et al. (2001) found that only 20 percent of the requirements have no interdependencies) and suggested the need for research on incorporating interdependencies in RP. Greer and Ruhe's EVOLVE (2003) is the only methodology found in the literature that specifically considers interdependencies between the requirements in the ranking process.

Third, the methodologies have also suggested a number of processes by which to assign the requirements to releases, relying on analytical tools to identify and integrate multiple stakeholders' rankings and multiple criteria per requirement. Some methodologies suggest relatively simple processes, such as Wiegers (1999) Value-Cost-Risk method where the requirements are ranked from 1 to 9; whereas, other methodologies, such as the EasyWinWin (Boehm et al., 2001) groupware system to capture individual ratings and to calculate and display level of consensus, are slightly more complex. However, there have been some unsatisfactory performance results of these methodologies with findings such as tedious and resulting in a loss of control (Karlsson and Ryan, 1997), difficult and pointless (Lehtola and Kauppinen, 2004), and untrustworthy results (Du, McElroy and Ruhe, 2006). This is troubling because it has been found that unclear prioritization methods affected the priority order of the requirements (Lehtola and Kauppinen, 2004). Recognizing the importance of clear and easy to use methods, some researchers have created methodologies that concentrate exclusively on reducing the complexity of the process (Bagnall, Rayward-Smith and Whitlley, 2001; Jung, 1998).

A SOFTWARE RELEASE PLANNING METHODOLOGY FOR DEVELOPERS

The aim of the new methodology is that after further testing and refinement it will be effective in helping software developers make 'better' decisions and consequently create release plans with less risk. This paper presents the first stage of this testing and refinement process by introducing the methodology and presenting an exploratory examination of the effect it has on developers' release plans.

Rather than focusing on the process of RP (i.e. use of absolute versus relative rankings, or use of genetic algorithms versus linear programming to calculate the increments), as is the focus of the majority of the literature, this methodology seeks to establish a soundly based set of criteria against which a single developer or multiple developers may evaluate software requirements thereby facilitating the assignment of these requirements to release plans. It is not the intention of this

methodology to include the criteria that all stakeholders may use to create releases, but rather to include those criteria that software developers may be best qualified to assess. A review of the literature indicated that it is suggested specifically that developers should consider variables such as cost (Karlsson and Ryan 1997), technical issues (Boehm et al., 2001), cost and value (Denne and Cleland-Huang, 2004), and cost and risk (Wiegers, 1999). Also, Greer and Ruhe's EVOLVE (2003) doesn't specifically indicate what criteria developers should rank the requirements on versus other stakeholders, but some of their criteria, seem to be best addressed by users - user priorities - and other criteria by the developers - risk factors, development effort and precedence, coupling and resource constraints.

The criteria in this new methodology are Essence, Risk and Interdependency. These criteria encompass the criteria from the RP literature with the exception of cost and effort, which were left out of the methodology presented in this paper for practical reasons[1].

Essence

The Essence criterion is used to determine the relative benefit that each requirement offers to the end product and indicates the degree to which the success of the project depends on each requirement. The "essence" of the software represents the minimum requirements necessary to satisfy the basic functionality of the software. For example, a requirement that is mandatory for business functionality or legally required would be assigned a higher value than a requirement that provides business value but has a feasible alternative such as a manual work-around. Furthermore, a requirement that is "nice-to-have" would be assigned the lowest value. An important aspect of the Essence criterion is that the essence of each requirement is determined independently of the technological constraints of that requirement.

Risk

The Risk criterion is used to determine the relative risk of developing each requirement. The risk refers to risk exposure resulting from the development and implementation of each requirement. It is essential to consider the risk in the ranking of the requirements to assign the riskier requirements higher priorities. The higher risk requirements should not be left to implement last in the project when resources or time may be limited or when necessary changes cannot be made because of technical constraints. Additionally, multiple high-risk requirements should not be scheduled for development at the same time. Rather, their development should be distributed to reduce the risk at any one time.

This Risk criterion incorporates theories and practices from software risk management such as risk identification, risk metrics and risk assessment, and considers the risk of each requirement from both the project and the technical perspectives. One of the contributions of this new methodology is the operationalization of the risk criterion, which is part of many of the other RP methodologies. For example, in Greer and Ruhe's methodology (2003) it is specified that developers should rate the requirements on the risk; however, no further guidance on what constitutes requirement risk is offered. This is where this new methodology may prove useful - the operationalized risk criterion could be used by the developers to rate the risk of the requirements.

The Risk criterion, briefly outlined in Figure 1, is composed of three main sub-criteria – Project Risk, Technical Process Risk, and Technical Product Risk - that contain specific operationalized measures for each risk factor.

[1] Cost and effort were excluded from the methodology because this would be difficult for the developers to estimate in an experimental setting, and instead the experiment participants were not given any cost information and were told to allocate the requirements into 12 equal releases.

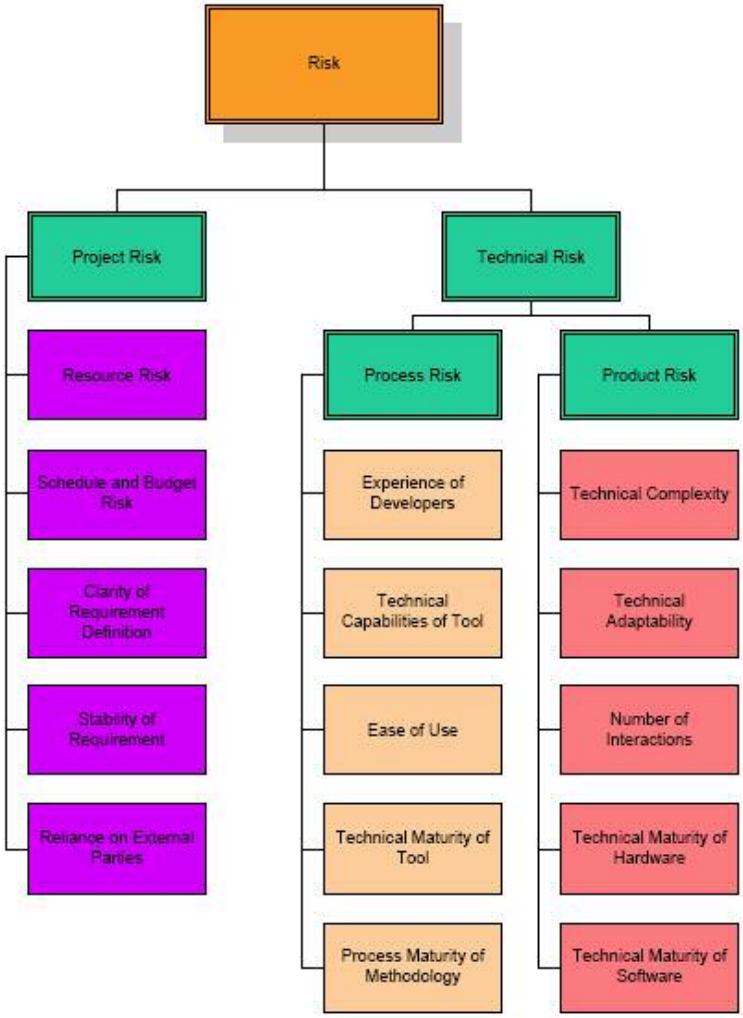

Figure 1. Overview of risk criterion

Interdependency

The Interdependency criterion addresses the relationship between the requirements and is used to determine the necessary implementation order since in some cases the inherent characteristics of the requirements may dictate their implementation order. The enhancement of the interdependency criterion to include functional interdependencies (referred to as Benefit Interdependencies in this methodology), in addition to the technical precedence that is included in Greer and Ruhe's EVOLVE (2003) is another contribution of this new methodology. The Benefit Interdependencies criterion goes one step further than the Essence criterion in that it considers whether the total functionality realized from implementing two requirements increase due to their synergistic effects. For example, requirements' interdependencies may stipulate their implementation order by identifying those complimentary requirements that, if developed in parallel, may offer additional business value, than if they were implemented independently. Technical Interdependencies, alternatively, incorporate the consideration of technical precedence into the RP process. Technical Interdependencies are characterized by "depends on" or "constrains" relationships. "Depends on" refers to whether the implementation of a requirement technically necessitates the implementation of a related requirement. A "Constrains" relationship between two requirements refers to whether implementing the first requirement limits the options of the second. If a requirement "constrains" another requirement(s) then it should be implemented before the other requirement(s).

A requirement that should be implemented "before" another either because of technical or benefit interdependencies would be assigned a higher priority than an "independent" requirement having no benefit or technical considerations constraining or promoting its implementation. For example, a requirement with the most dependants, and thus having the biggest effect on

the rest of the system if it changes (i.e. infrastructure) should be implemented before another requirement that does not affect the system functionality or require other aspects of the system in order to function. The next section discusses the experiment that was conducted to provide some initial testing of this methodology for software developers.

EXPERIMENT

This controlled experiment aims at comparing the release plans created with the use of the software RP methodology for developers (referred to as formal methodology) introduced above with plans created using an informal methodology. The consistency between the plans created formally versus informally will be calculated to compare the plans. Consistency refers to the degree of similarity between the plans. The premise is that if the release plans created formally versus informally are similar then the formal methodology includes criteria that the developers intrinsically use when informally developing the release plans. The following hypothesis is investigated:

H_0 : The release plans created with the use of the software RP methodology for developers will not be consistent with the plans created informally.

H_1 : The release plans created with the use of the software RP methodology for developers will be consistent with the plans created informally.

Participants

A non-probability judgment sampling procedure was used to select participants for this exploratory study. Ten undergraduate and graduate students in a university's Computer Science and Software Engineering departments were selected to participate in this study. These students all have one to two years work experience in software development and have studied similar courses. The use of students is suitable in this study because other studies evaluating the efficacy of RP methods have also used students as subjects in their experiments (e.g. Du et al, 2006; Karlsson et al., 2007) and because one of the situations in which students most often seem to be suitable is when evaluating if a new technique is better than a known technique (formal vs informal RP in this experiment) (Berander, 2004).

Experimental Design

Pre-tests were performed incrementally with three software developers. After each pre-test, any weaknesses were corrected and the newest version of the instrument was given to the next pre-test participant. The pre-tests were useful for eliminating errors in the experimental design and served as a trial for the full execution of this experiment.

For the experiment, a control group and an experimental group, each with five subjects, were selected to participate. All participants were asked to rank 27, randomly ordered, software requirements for a spreadsheet program (listed in Table 1) and assign the requirements to 12 releases.

Working independently, the participants in the control group created release plans for the software program informally. In fact, this group was not told about this new RP methodology or about the purpose of the study. Each participant was left up to his/her own to determine how he/she would rank the software requirements and create the release plan. Conversely, the experimental group was given an introduction to the methodology and some supporting documentation, and each participant was instructed to independently follow this new methodology to rank the software requirements and create a release plan.

The participants in the experimental group were instructed to rank the requirements against the Essence and operationalized Risk criteria on 7-point Likert scales. Then, the participants were instructed to assign one ranking to each requirement based upon the Essence and the Risk scores. This helps the participants to make trade-off decisions regarding Essence and Risk when assigning the rankings. Finally, the participants were asked to estimate the Interdependencies between requirements, and assign each requirement to a release.

Requirement Number	Requirement Name
1	Row, column – Read/Write/Select (e.g., group of cells).
2	Bottom text floating or fixed. (e.g. Freeze Frame)
3	Simple formulas in cell (parse, interpret) e.g.=sum(c1:c3)
4	All text cells and labels -font size, colour, style, alignment.
5	Multiple matrices per window.
6	Insert before selected row or column.
7	Rows, columns and cells showing default value on open.
8	Variable height rows.
9	Title above and bottom of matrix for description info.
10	Scroll bars optional when matrix fits in window.
11	Input in cell (type in cell).
12	Cells – Read/Write/Select (e.g., individual cell).
13	Error message area.
14	Insert row or column at end.
15	One matrix per window.
16	Row and column labels auto generated.
17	Row and column labels optional (e.g., no row headers).
18	Variable width columns.
19	Row and column labels top and left.
20	Input in input area (type in separate input area).
21	Click resize on columns and rows.
22	Cell data types – Real, Integer, Boolean, Date, String
23	Automatic resize view/ view on window resize.
24	Default row, column settings - width, formats, data types.
25	Row and column labels unique and having text for names.
26	Delete selected row or column.
27	Title above and below optional.

Table 1. Software requirements for spreadsheet program

Results

Refer to the Appendix for the release plans created by the participants. The data was analyzed using nonparametric measures of correlation for ordered data. Measures of correlation determine the probability associated with the occurrence of a correlation as large as the one observed in the sample under the null hypothesis that the variables are independent or unrelated in the population. Nonparametric measures are suitable for the analysis of the data in this study because the sample size is small, and because the data is ordinal instead of ratio or interval data which is preferred for parametric tests.

The analysis suggests that the release plans created with the use of the methodology are consistent with the plans created informally. Kendall's Coefficient of Concordance W was used to test the correlation of the release plans of all participants (control and experimental groups). For Kendall's W, 1 is complete agreement and 0 is no agreement. In this study Kendall's W of 0.319 with a probability of occurrence of $p<0.001$ reflects a modest, but highly-significant, amount of agreement between all participants. Therefore, it can be concluded that the agreement among the participants is higher than it would be had their rankings been random. Thus, the null hypothesis is rejected for these cases.

Due to the modest amount of agreement between all participants, additional analysis was performed to see what effect the use of the methodology had on the consistency of release plans within the experimental and control groups. As shown in Table 2, Kendall's W for each group has a probability of occurrence of $p < 0.02$ and $p<0.001$ respectively. This means that there is significant agreement within each group of participants. Since W=.498 for the experimental group is bigger than that of W=.347 for the control group, it would suggest that the experimental group participants are in more agreement than the control group. The analysis suggests that participants exposed to the formal methodology are more consistent as a group. This is not surprising given that generally if a treatment relates to the topic it should create a more consistent response. Further research is warranted in this area, but consistency in using this methodology versus informally creating release plans may be beneficial as it may increase the agreement among participants and make the resulting releases more predictable. The larger consistency among the experimental group participants than the control group may also reflect the fact that use of the

methodology does change the way that release plans are created with a small number of requirements and it raises the question of what would be the affect with large numbers of requirements.

Between Releases	W	α
Control Group	0.347	$0.01<p<0.02$
Experimental Group	0.498	$p<0.001$

Table 2. Correlation among the release plans of each group of participants

Additional Test

Since the release plans created formally and informally are highly correlated, further analysis of the plans was conducted to determine if the participants did, in fact, apply the methodology to rank the requirements and to create their releases, or did they seem to rank the requirements using the methodology, but in the end create releases based on their own assessment? Refer to Table 3 for the average Essence and Risk rating and associated ranking assigned to each requirement by the experimental group using the methodology.

Requirement Number	B1		B2		B3		B4		B5	
	Avg*	Rank	Avg	Rank	Avg	Rank	Avg	Rank	Avg	Rank
1	2	24	5	2	2	13	2	25	7	1
2	4	11	2	27	4	11	4	17	3	20
3	7	1	5	12	7	4	6	8	5	5
4	3	18	4	16	4	14	2	22	2	26
5	4	13	5	13	5	10	6	12	4	9
6	5	5	4	14	5	7	4	10	4	23
7	3	12	3	24	1	26	1	27	3	17
8	2	21	3	18	3	16	4	15	5	11
9	2	25	3	23	2	17	2	18	4	22
10	2	27	3	19	1	27	2	19	5	6
11	3	14	4	5	5	3	5	16	4	3
12	3	19	5	1	2	12	3	23	6	2
13	3	15	5	7	4	9	4	3	6	10
14	4	7	5	4	5	5	4	11	3	16
15	6	2	3	20	5	1	5	1	5	7
16	4	8	3	25	3	20	3	6	3	21
17	2	22	3	21	1	25	4	24	3	25
18	4	9	3	17	3	15	4	14	5	12
19	5	3	3	22	2	19	4	5	4	18
20	5	4	4	11	5	8	5	2	4	15
21	3	17	5	6	3	17	4	20	4	24
22	4	10	5	3	5	2	3	4	4	4
23	3	23	4	8	2	21	3	21	4	19
24	2	20	3	15	1	24	3	9	5	13
25	3	16	3	26	3	18	2	7	4	8
26	4	6	4	9	5	6	5	13	5	14
27	2	26	4	10	2	22	1	26	1	27

** Avg refers to the average of the Essence and Overall Risk ratings assigned to each requirement by the study participants.*

Table 3. Rankings assigned by the experimental group

The Spearman Rank-Order Correlation Coefficient was calculated to test the correlation between the average Essence and Risk rating of each requirement and its ranking, refer to Table 4. The requirements were ranked (from 1 to 27) and then

assigned to releases based on that ranking and any interdependency considerations. It was found that the correlation for each participant was at the 0.01 level, therefore the results from this test indicate that those participants exposed to the RP methodology do in fact follow the methodology to create the release plans. Those requirements that were rated high on average on the Essence and Risk criteria were also ranked high (e.g. requirement 3 for participant B1 received the highest rating of 7 and was ranked number 1), and therefore would be assigned to an earlier release. This means that a requirement regarded as part of the basic functionality of the software and that had higher risks to develop would be recommended to be developed first by the participant. This is important because it suggests that the methodology is trustworthy as the resulting release plans reflect the participants' opinions.

Participant	r_s
B1	-.965
B2	-.944
B3	-.903
B4	-.659
B5	-.824

Table 4. Correlation between criteria and the final priorities

CONCLUSIONS

This new methodology is a first step towards a theoretically based methodology to improve RP and reduce software development risks. As discussed, it is proposed that there are characteristics of requirements such as essence, risk and interdependencies that are considered in parallel, weighted, and reflected upon by software developers both consciously and intuitively in the RP process. No other studies have looked specifically at the criteria considered by developers so this is where this paper makes its greatest contribution. The findings of the experiment comparing the release plans created by developers using an informal versus this new methodology suggest that these criteria may be considered by developers. At this point in the research it is not possible to conclude if the fact that the releases created informally and formally are consistent is due to the fact that the participants did not use the methodology and therefore just ranked the requirements as they would have informally, or if the participants did in fact use the approach and it does reflect the essence, risk and interdependency criteria that are inherently considered (i.e. part of their own schema) when creating releases . However, the additional test did indicate that the developers did in fact use the methodology, and that the resulting release plans do reflect the participants' opinions.

This study found that with small sets of requirements there is no significant difference in releases created with a formal versus an informal methodology; however, where benefits may be felt is as the number of requirements increases. With large numbers of requirements it may be difficult for developers to intrinsically rank the requirements and they may need such a formal methodology with operationalized criteria as the methodology introduced in this paper. This RP methodology for developers could be used in conjunction with other existing RP methodologies to supplement the decision making process of the developer stakeholder group. For example, it could be used with EVOLVE where the developer's plans captured through the methodology introduced in this paper could be incorporated with other stakeholders' plans through EVOLVE's genetic algorithms.

Limitations

The exploratory nature of this study must be reiterated. The scope of this study involved a small number of participants and involved one experimental problem (the spreadsheet program); therefore, the representativeness of the participants and the spreadsheet program used in this study is unknown, thus limiting the ability to generalize the findings. Consequently, the results of this study are discussed as preliminary findings and as areas requiring further research. However, other studies of RP methodologies have faced similar limitations due to small sample sizes (e.g. Du et al., 2006; Karlsson et al., 2007; Lehtola and Kauppinen, 2004) while still contributing to the literature.

When considering the results of the experiment a couple of points should be considered. First, the subjects may be limited in their ability to apply the criteria in this experimental situation. In the "real-world" they could talk to users or other developers to clarify requirements, risks, etc. Second, it is not possible to completely control how the control group did the RP. However, as Ziemer and Calori (2007) noted when they used students as a control group in their RP experiment, the students "will probably have no knowledge about RP methods, and will just try to solve the problem as best they can," (p.110).

Implications for Future Research

The findings of this study are preliminary and offer a framework for future analysis. More definitive research studying the impact of use of the new methodology on developers' creation of releases is needed to determine whether the results of this study are generally applicable. For example, additional studies could examine the impacts of the use of this methodology on multiple projects with different software programs, and with more developers and requirements. Studies could be designed to test the impact of the use of this method in conjunction with other methodologies, and against informal and other formal methodologies. It may prove useful in the future to compare the resulting releases created by developers at varying levels of expertise to determine how the releases differ and to see if specifically identifying and operationalizing criteria upon which to base the releases has an impact. Finally, further testing is necessary before it can be determined whether these criteria are sufficient and if other criteria, such as essence, can be operationalized (i.e. need for compliance, aides usability, functionality, manual workaround possible, etc.). Research in all these areas could have impacts on RP in the future.

APPENDIX

There are 27 software requirements assigned to 12 releases.

		Experimental Group					Control Group				
	Requirements	**A1**	**A2**	**A3**	**A4**	**A5**	**B1**	**B2**	**B3**	**B4**	**B5**
1	Row, column – Read/Write/Select (e.g., group of cells).	4	3	6	2	10	10	1	5	12	4
2	Bottom text floating or fixed. (e.g. Freeze Frame)	11	10	12	9	12	5	2	5	9	9
3	Simple formulas in cell (parse, interpret) e.g.=sum(c1:c3)	10	9	6	5	2	2	2	2	3	7
4	All text cells and labels -font size, colour, style, alignment.	2	8	5	7	4	8	9	7	11	12
5	Multiple matrices per window.	1	1	12	5	3	6	4	4	5	3
6	Insert before selected row or column.	4	4	3	1	6	3	6	3	4	9
7	Rows, columns and cells showing default value on open.	2	11	2	3	4	5	5	11	12	10
8	Variable height rows.	2	2	8	6	8	9	7	8	7	6
9	Title above and bottom of matrix for description info.	8	10	7	4	4	11	8	9	9	11
10	Scroll bars optional when matrix fits in window.	12	1	11	11	12	11	10	11	9	4
11	Input in cell (type in cell).	5	5	1	8	10	7	2	1	8	2
12	Cells – Read/Write/Select (e.g., individual cell).	5	5	5	5	10	8	1	5	11	3
13	Error message area.	1	12	4	3	9	7	3	4	1	5
14	Insert row or column at end.	4	4	3	1	6	3	6	3	4	5
15	One matrix per window.	1	1	1	1	3	1	4	1	1	7
16	Row and column labels auto generated.	3	7	9	10	5	3	5	10	2	8
17	Row and column labels optional (e.g., no row headers).	3	7	10	11	5	10	5	11	12	12
18	Variable width columns.	2	2	8	6	8	4	7	8	7	6
19	Row and column labels top and left.	3	7	9	4	12	1	2	9	2	9
20	Input in input area (type in separate input area).	6	5	1	1	9	1	2	4	1	8
21	Click resize on columns and rows.	7	2	8	12	11	9	12	8	10	11
22	Cell data types – Real, Integer, Boolean, Date, String	2	6	4	3	1	4	2	1	2	1
23	Automatic resize view/ view on window resize.	9	2	11	12	11	11	3	10	11	10
24	Default row, column settings - width, formats, data types.	2	11	2	3	4	10	2	11	3	1
25	Row and column labels unique and having text for names.	3	7	9	4	5	7	5	9	2	2
26	Delete selected row or column.	4	4	3	2	7	3	11	3	6	7
27	Title above and below optional.	8	10	7	11	4	11	8	10	12	12

REFERENCES

1. Aurum, A., Wohlin, C. (2003) The fundamental nature of requirements engineering activities as a decision-making process, *Information and Software Technology,* 45, 945–954.
2. Bagnall, A. J., Rayward-Smith, V. J., and Whittley, I. M. (2001) The Next Release Problem, *Information and Software Technology*, 43, 14, 883-890.
3. Beck, K., and Fowler, M. (2001) Planning Extreme Programming, Addison-Wesley, Toronto.
4. Berander, P. (2004) Using Students as Subjects in Requirements Prioritization, *Proceedings of the 2004 International Symposium on Empirical Software Engineering (ISESE'04),* August 19-20, Redondo Beach, CA, USA, 167-176.
5. Boehm, B., Grunbacher, P., and Briggs, R. (2001) Developing Groupware for Requirements Negotiation: Lessons Learned, *IEEE Software*, May/June, 46-55.
6. Boehm, B.W. (1991) Software Risk Management: Principles and Practices, *IEEE Software*, January, 32-42.
7. Carlshamre, K, Sandahl, M., Lindvall, B., Regnell, Natt och Dag, J. (2001) An industrial survey of requirements interdependencies in software product release planning, *International Symposium on Empirical Software Engineering*, CA, USA, 84–92.
8. Denne, M. and Cleland-Huang, J. (2004) The Incremental Funding Method: Data Driven Software Development, *IEEE Software*, 21, 3, 39–47.
9. Du, G., McElroy, J., and Ruhe, G. (2006) A family of empirical studies to compare informal and optimization-based planning of software releases, *Proceedings of the 2006 ACM/IEEE International symposium on empirical software engineering*, Sept. 21-22, Rio de Janeiro, Brazil, 212–221.
10. Greer, D. and Ruhe, G. (2004) Software release planning: an evolutionary and iterative approach, *Information and Software Technology*, 46, 243–253.
11. Greer, D., and Ruhe, G. (2003) Quantitative studies in software release planning under risk and resource constraints, *Proceedings of the 2003 International Symposium on Empirical Software Engineering (ISESE'03)*, September 30-1 October 2003, 262 – 270.
12. Hofmann, H.F., and Lehner, F. (2001) Requirements Engineering as a Success Factor in Software Projects, *IEEE Software*, July/August, 58-66.
13. Jung, H.-W. (1998) Optimizing Value and Cost in Requirements Analysis, *IEEE Software*, 74-78.
14. Karlsson, J., Ryan, K. (1997) A cost-value approach for prioritizing requirements, *IEEE Software*, 14, 67–74.
15. Karlsson, L., Thelin, T., Regnell, B., Berander, P., and Wohlin, C. (2007) Pair-wise comparisons versus planning game partitioning--experiments on requirements prioritisation techniques, *Empirical Software Engineering.* 12, 1, 3-33.
16. Lehtola L, Kauppinen M (2004) Empirical evaluation of two requirements prioritization methods in product development projects, *Proceedings of the European Software Process Improvement Conference*, November 10-12, Trondheim, Norway, 161–170.
17. Lehtola, L., Kauppinen, M., Kujala, S. (2004) Requirements Prioritization Challenges in Practice, *Proceedings of 5th International Conference on Product Focused Software Process Improvement*, April 5-8, Kansai Science City, Japan, 497-508.
18. Siegel, S. and N.J. Castellan Jr. (1988) Nonparametric Statistics for the behavioural sciences, (2nd Edition), McGraw Hill, Toronto.
19. Standish Group (The) (1995) The Chaos Report.
20. Wiegers, K.E. (1999) First things first: prioritising requirements, *Software Development,* 7, 9, 48–53.
21. Wohlin C, Aurum A. (2005) What is important when deciding to include a software requirement in a project or release? *Proceedings of the International Symposium on Empirical Software Engineering*, Nov. 17-18, Noosa Heads, Australia, 237–246.
22. Ziemer, S. and Calori, I.C. (2007) An Experiment with a Release Planning Method for Web Applciation Development. In: Lecture Notes in Computer Science, Springer, Berlin, Germany, 106-117.