Binary Priority List for Prioritizing Software Requirements

Thomas Bebensee, Inge van de Weerd, and Sjaak Brinkkemper

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands
{tbebense,i.vandeweerd,s.brinkkemper}@cs.uu.nl

Abstract. [Context and motivation] Product managers in software companies are confronted with a continuous stream of incoming requirements. Due to limited resources they have to make a selection of those that can be implemented. However, few prioritization techniques are suitable for prioritizing larger numbers of requirements. [Question/problem] Binary Priority List (BPL) is a binary search based technique for prioritizing requirements. Academics and practitioners have referred to it in previous works. However, it has not been described and researched in detail. [Principal ideas/results] This paper introduces BPL, examines how it can be used for prioritizing requirements and assesses its prioritization process quality by comparing it to another prioritization technique. A facilitating tool was developed and applied in two small Dutch product software companies. [Contribution] The paper demonstrates that the technique can be successfully used to prioritize requirements and is especially suitable for a medium amount of low-level requirements.

Keywords: Binary Priority List, Binary Search Algorithm, Requirements Prioritization, Software Product Management, Agile Project Management.

1 Introduction

Product software companies produce packaged software products aimed at a specific market [1]. In product software companies, the number of requirements from the market typically exceeds the number of features that can be implemented in one release due to limited resources. Requirements prioritization is aimed at responding to this challenge. It is defined as "an activity during which the most important requirements for the system (or release) should be identified" [2]. According to the reference framework for software product management, in which key processes, actors, and relations between them are modeled, it is the first step in the release planning process, "the process through which software is made available to, and obtained by, its users" [3]. The main actor in prioritization is the product manager, but other stakeholders (development, sales & marketing, customers, partners etc.) may influence it as well [3].

R. Wieringa and A. Persson (Eds.): REFSQ 2010, LNCS 6182, pp. 67–78, 2010.

One particular important stakeholder in software product management is the customer, or rather the representative of a large number of customers (market). Agile Project Management (APM) takes this into account by "energizing, empowering and enabling project teams to rapidly and reliably deliver customer value by engaging customer and continuously learning and adapting to their changing needs and environments" [4].

In the context of APM, product managers are responsible for requirements management, in contrast to the "traditional understanding" where it was the developers' responsibility [5]. The product manager is considered the customers' voice and in his role as the interface between the market and the development team. As such, he is also responsible for the prioritization of requirements [5]. In order to correspond to the prerequisite of an iterative (re)prioritization, agile prioritization techniques must reflect this dynamic nature. This enables delivering a maximized business value to the market throughout the project [5].

1.1 Requirements Prioritization Techniques

Prioritization of requirements is usually done during a prioritization session. Three stages in a prioritization session are distinguished [6]: (1) the preparation stage to structure the requirements and prepare the execution of the prioritization; (2) the execution stage where the actual prioritization is performed; and finally (3) the presentation stage where the prioritization results are presented.

Racheva *et al.* [5] review a number of agile requirements prioritization techniques. Based on the descriptions provided by them, these techniques can be classified into two main categories: techniques used to prioritize small amounts of requirements (small-scale) and techniques that scale up very well (medium-scale or large-scale), thus can be used for the prioritization of larger amounts of requirements (Racheva *et al.* talk about several dozen requirements).

Small-scale techniques can usually be used without the aid of a software tool and are often relatively simply structured. The techniques mentioned by Racheva *et al.* [5] are the round-the-group prioritization, the $100 allocation technique, the multi-voting system, the pair-wise analysis, the weighted criteria analysis, the dot voting technique, and the Quality Functional Deployment approach.

Medium-scale or large-scale techniques might be based on relatively complex algorithms or at least due to the large amount of requirements need tool support. In this category, Racheva *et al.* [5] refer to the MoSCoW technique, the Binary Priority List (they refer to it as "Binary Search Tree technique"), the Planning Game, and Wiegers's matrix approach (cf. [7]).

In addition, there are some techniques that do not fit very well into this scheme. One is ranking based on product definition, which could be used as a complement to other techniques to first derive a de-personalized measure of priority. Another technique is the application of mathematical programming techniques to interlink the whole release planning process. And yet another one is the Analytic Hierarchy Process, which is rather complex but limited to a small number of requirements due to the high number of comparisons necessary (cf. [8]). This technique has received considerable regard in literature (cf. [8], [6], [9]).

1.2 Research Question

Since product managers in product software companies usually have to prioritize larger amounts of requirements, medium or large-scale techniques are of special interest to them. In this paper we will therefore discuss how one particular of these techniques, which we call Binary Priority List (BPL), can be applied by product managers of such companies. Since it is a relatively simple technique, we expect it to be especially interesting for smaller product software companies that want to formalize their requirements prioritization process. The research question we want to answer is:

How can BPL be applied as a requirement prioritization technique in small product software companies and how reliable are its results?

In order to answer the research question, the remainder of the paper is structured as follows. Section 2 gives a rationale for the research done and explains the research approach used. In section 3, a detailed description of the BPL and the supporting tool is provided. Section 4 presents the case study approach and a description of the research sites and the results of the case studies are presented. Finally, in section 5, conclusions are drawn and areas of further research are indicated.

2 Rationale and Research Approach

Binary search is a popular algorithm to sort and search information [10]. A quick search on the Internet reveals a sizeable amount of publications on it (cf. [11], [12], [13]). However, this algorithm can also be used to prioritize software requirements. In this context we refer to it as Binary Priority List (BPL). There is only little notion of BPL in literature. Only two papers deal with it in more detail.

Karlsson *et al.* [6] compared BPL to five other prioritization techniques. In their research, BPL scored relatively weak in terms of time consumption, ease of use, reliability and fault tolerance. On the other hand, Ahl [8] conducted an experiment in which he compared BPL with four other prioritization techniques in terms of reliability of results, ease of use, time consumption and scalability. In his experiment BPL was considered the best out of the five techniques. Ahl comes to the conclusion that BPL scales up very well and is therefore especially interesting for prioritizing larger amounts of requirements.

A literature research did not reveal any evidence that BPL has been described in detail. Apparently, it has not been examined in the specific context of product software companies yet. We think that this makes it an interesting research object and we therefore describe how it can be applied in that context.

Our further research approach was the following:

1. *Description and tool support:* We created a detailed description on how to apply BPL for prioritizing software requirements. In addition, we developed a tool for product managers to use BPL as a prioritization technique for their software requirements.
2. *Case studies:* In order to validate the technique's application, we conducted case studies at two product software companies. The goal of these case studies was to test whether BPL can be applied as a requirements prioritization technique and

how it is perceived by experience software product managers. In addition, the technique's prioritization process quality, expressed by the factors reliability, time consumption and ease of use, was evaluated by comparing it to Wieger's prioritization approach.

This research approach has been designed according to the guidelines proposed by Hevner, March, Park, and Ram [14]. Case studies (cf. [15]) are considered very suitable for industrial evaluations of techniques [16].

3 Binary Priority List

Binary Search Tree (BST) is a widely used algorithm to sort and search information but it can also be used to prioritize requirements [6]. Applied to the context of requirements prioritization we refer to it as Binary Priority List (BPL). A systematical structure of the technique is shown in Figure 1.

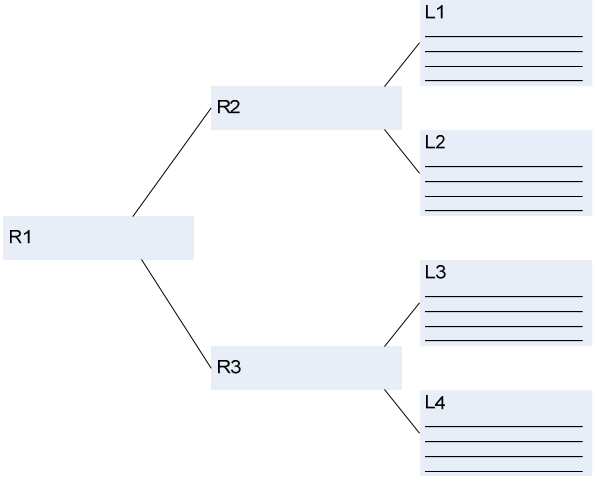

Fig. 1. Systematic structure of Binary Priority List

The figure shows a list of requirements containing three example requirements (R1, R2 and R3) and a number of sub-lists (L1, L2, L3 and L4) containing more requirements. In accordance with a binary tree structure (cf. [17] and [6]) requirements that are further up are more important than requirements further down. Therefore, R1's priority is lower than that of R2 but higher than that of R3. Subsequently, all requirements listed in the sub-lists L1 and L3 have higher and all requirements in L2 and L4 have a lower priority than their root requirement, R2 and R3 respectively.

The steps of applying the technique are (cf. [5], [6] and [8]):

1. Pile all requirements that have been collected from various sources.
2. Take one element from the pile, and use it as the root requirement.

3. Take another requirement and compare it in terms of priority to the root requirement.
4. If the requirement has a lower priority than the root requirements, compare it to the requirement below the root and so forth. If it has a higher priority than the root, compare it to the requirement above the root and so forth. This is done until the requirement can finally be placed as sub-requirement of a requirement without an appropriate sub-requirement.
5. Steps 2 to 4 are repeated for all requirements.
6. Finally, traverse the list from top to down to get the prioritized order of the requirements.

BPL can be applied by placing a number of cards, which represent the requirements, on a blackboard. However, when large numbers of requirements have to be processed, a software tool becomes necessary. Such a tool is also useful when the structure of the list is supposed to be altered in the future and should be stored electronically.

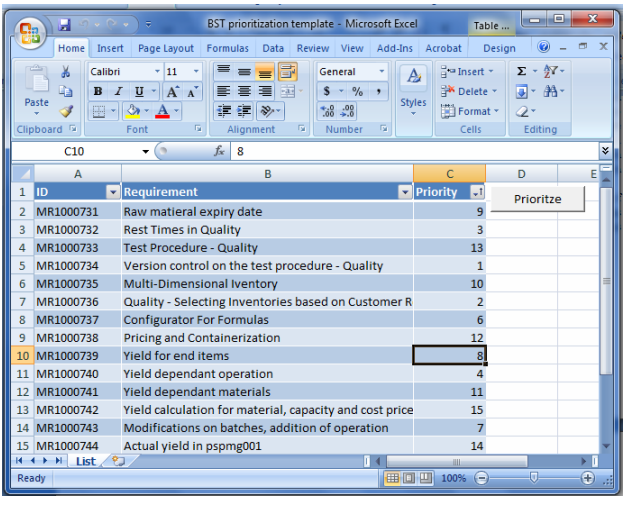

Fig. 2. Screenshot of the BPL tool

We developed a simple spreadsheet tool based on Microsoft Excel (see Figure 2). A macro guides the user through the prioritization process by asking him to compare different requirements and deciding which one is respectively more important than the other. The list structure is saved in an implicit form in a hidden spreadsheet, which allows saving it and changing it at a later time without having to run the whole prioritization again.

Due to its simple nature we expect BPL to be especially useful in environments where no formal prioritization techniques have yet been used to assist the product manager in prioritizing his criteria. This type of environment is mostly likely to be found in small product software companies that have recently grown in terms of requirements to be processed.

4 Case Studies

4.1 Approach

The case studies were divided into three phases: (1) a prioritization with BPL, (2) a prioritization with Wiegers's technique, and (3) an evaluation of the two techniques. To reduce the number of confounding factors as proposed by Wohlin and Wesslen [16], the same way of input, namely Excel spreadsheets, were used.

We conducted case studies in two small product software companies in which we compared BPL with Wiegers's technique in terms of the following three factors:

1. *Time consumption:* indicates the time necessary to prioritize a certain number of requirements.
2. *Ease of use:* describes how easy it is to use the examined prioritization technique assessed by the respective product manager.
3. *Subjective reliability of results:* indicates how reliable the result of the prioritization technique is in the opinion of an experienced product manager and thus how applicable the technique is to the respective company.

The ultimate goal was to show that BPL can be applied by product managers in small product software companies to systematize their requirement prioritization practices. In order give an additional indication of the technique's relative prioritization process quality, we compared it with Wieger's approach, a commonly used prioritization technique that is applied to similar numbers of requirements as BPL (cf. [7], [18] and [19]).

4.2 Research Sites

The first case study took place at Edmond Document Solutions (below referred to as Edmond), a small Dutch product software company. Edmond is a specialist in providing document processing solutions to document intensive organizations. The company employs 15 people whereof six are involved in software development. The software development process is based on Scrum (cf. [20]), an agile software development method.

The current release planning process takes place in two stages. High-level requirements are discussed once a year among the product manager, the operations manager and the sales director. The selection and order of requirements is defined in an informal discussion between the three. To gain a good understanding of the market, they visit exhibitions, read journals and communicate with major customers. The product manager estimates the required resources and makes sure the needed resources do not exceed the resources available.

In the second stage of the release planning process, the product manager derives low-level requirements from the high-level requirements defined in the first stage. To manage requirements together with tasks, bugs, improvements and impediments he uses JIRA (cf. [21]), a bug and issue tracking software. Prioritization of low level requirements takes place by comparing them in pairs with each other. In this process no formal technique is used. Subsequently, he assigns the requirements to Scrum sprints, periods of four weeks where developers work on a certain number of requirements.

The second case study took place at Credit Tools (below referred to as CT), a small Dutch product software company. CT produces encashment management software.

Five out of the company's 25 employees are software developers. In addition, there are two outsourced software developers. The company's development method is Rapid Application Development (cf. [22]).

Requirements are generated from customer requests, from information acquired through consultants and from ideas generated by the company's owners. JIRA is used to collect requirements and to communicate with the outsourcing developers. There is no formal process of requirements prioritization and not all requirements are systematically noted down.

4.3 Results

At Edmond, the product manager had prepared a list of 68 low-level requirements. This number of requirements was not chosen intentionally but corresponded to the number of requirements to be prioritized at that moment. To perform a prioritization based on the Wiegers's approach, he had also estimated for each requirement its relative benefit, its relative penalty, its relative costs, and its relative risk. At CT, 46 low-level requirements were prioritized instead. Again, this corresponded to the product manager's current requirements list.

In the first phase of each case study, the list of requirements was copied into the BPL spreadsheet tool. The product manager then went through the prioritization process, in which he was asked to perform a pair-wise comparison of the requirements' importance following the BST algorithm.

In the second phase, the requirements and the corresponding values estimated before were copied into a spreadsheet prepared for Wiegers prioritization. The spreadsheet automatically calculated relative priorities based on the estimates provided. Subsequently, the requirements were sorted by increasing priority.

In the evaluation phase, the product managers were asked for their opinion on the two compared prioritization techniques, especially with regard to ease of use and reliability. In addition, the time needed to perform a prioritization based on the two techniques was compared. The results are shown in Table 1.

Table 1. Comparison of the two techniques

	BPL		**Wiegers**	
	Edmond	*CT*	*Edmond*	*CT*
Ease of use	8/10	8/10	7/10	4/10
Subjective reliability	7/10	7/10	4/10	5/10
Time consumption	30 min	20 min	120 min	50-60 min

The product managers of both companies had a quite positive impression of BPL, which is reflected by their rating of the technique in terms of ease of use and reliability (see Table 1). One indicated that the ten most highly prioritized requirements corresponded exactly to his own manual prioritization. Lower priority requirements, however, partly differed from it. He supposed that this could be caused by accidentally giving wrong answers while going through the prioritization process and proposed to improve the user interface by including a possibility to correct a wrong choice. The second product manager indicated that the informal approach to prioritizing

requirements that they had used so far has many similarities with BPL. Therefore, he considered it as quite suitable for his company.

To compare the results of two techniques, Table 2 shows the ten requirements with the highest priority according to both techniques.

Table 2. The ten most highly prioritized requirements according to both techniques

	Edmond		**CT**	
Priority	*BPL*	*Wiegers*	*BPL*	*Wiegers*
1	5	54	18	19
2	2	23	35	43
3	28	21	25	11
4	27	61	14	18
5	3	63	24	29
6	6	12	16	42
7	23	28	11	3
8	12	45	42	22
9	16	50	33	24
10	7	53	2	13
Avg. Diff.	**14.75**		**8.54**	

Figure 2 (Edmond case) and Figure 3 (CT case) show the priorities of all requirements assigned by Wiegers's techniques plotted over the priorities assigned by BPL, which is also represented by the straight line. The stronger the two graphs in each figure differ, the bigger the difference between the priorities assigned by the two techniques.

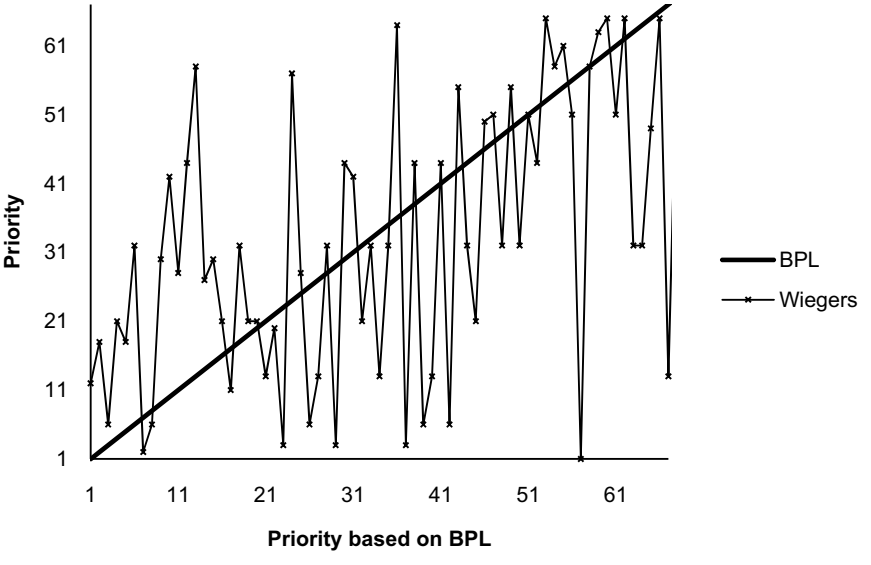

Fig. 3. Edmond Case: comparison of the two prioritization techniques

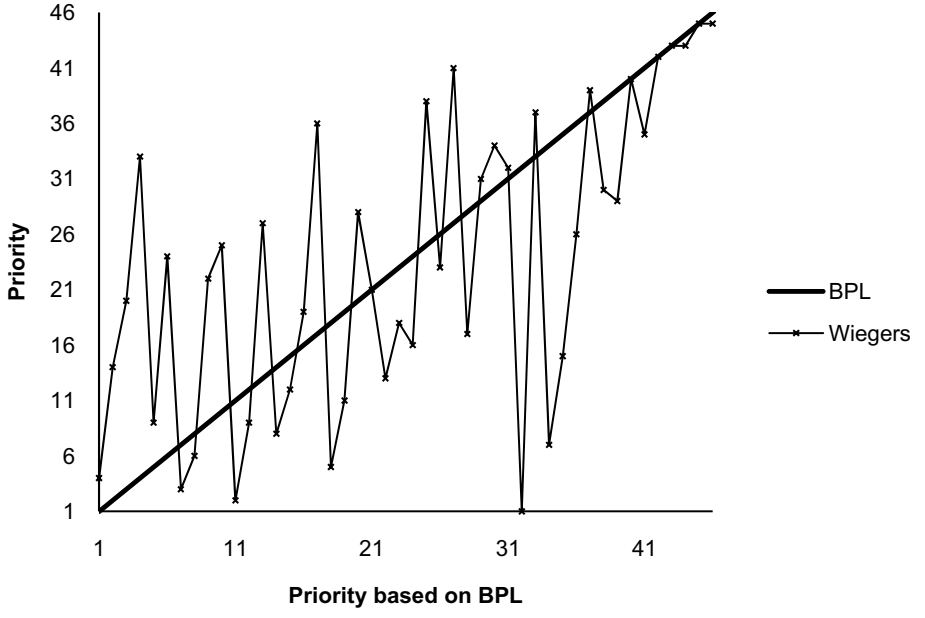

Fig. 4. CT Case: comparison of the two prioritization techniques

In general, in both case studies, the results from both techniques differed strongly from each other. Interestingly, however, in the second case study, the results of the two techniques are close to each other for the requirements with the lowest priority. The average difference of the priorities based on the two techniques was 14.75 in the first case study and 8.54 in the second case study. The maximal difference between BPL and Wiegers's technique was 56 in the first case and respectively 31 in the second. Both product managers rate Wiegers's technique rather low in terms of reliability (see Table 1). However, the product manager of Edmond noted that a better calibration might have resulted in an improvement. In their rating of ease of use of Wiegers's technique, the two product managers differ considerably. The product manager of CT mentioned that he found it difficult to estimate values for relative risk and penalty.

4.4 Discussion

The two techniques compared in these case studies differ in the way the prioritization criterion is articulated. In contrast to Wiegers's technique, BPL does not make the underlying inputs for the prioritization explicit. The results are rather based on the spontaneous intuition of the person applying it. However, ideally the product manager bases his considerations during the BPL prioritization on the same inputs, namely benefit, penalty, costs and risk as in Wiegers's technique or even considers other important factors, as for instance attractiveness for development.

In order to make sure that this happens, the question asked in the prioritization dialogue of the BPL tool should be formulated accordingly. During the case study the

question was “Is req. X more important than req. Y?” Now, we would suggest formulating it as “Do you want to implement req. X before req. Y?” instead. This formulation more explicitly suggests considering other factors than just importance. However, we still recommend avoiding prioritization of requirements that differ considerably in terms of costs to be implemented.

The difference in how explicitly the two techniques require the product manager to express the factors that determine his considerations also explains the different time consumption of the two techniques. The prioritization with BPL consequently only takes one quarter (Edmond case) to one third (CT case) of the time of Wiegers’s technique.

BPL was perceived the easier of the two techniques. This can also be related to its simple structure. The user basically only has to compare two requirements at the time and can apply an own set of criteria.

The strong difference between the two techniques’ prioritization suggests that the result of a prioritization session depends heavily on the respective technique used. The accordance of the two techniques’ results for requirements with low priority is actually the only point where both techniques correspond considerably with each other. It might be explained by the fact that the product manager considered the last four requirements as so unimportant that he assigned very low scores to three of the determining factors of Wiegers’s technique to them.

Both product managers considered the result of BPL considerably more reliable than that of Wiegers’s technique. This may be attributed to the fact that they could directly apply their own comparison criteria. As a consequence, the result always stays relatively close to his intuition. However, Wiegers’s technique might become more reliable when it is fine-tuned to the circumstances of the environment it is applied in by changing the weights of the four input factors. To test this, we would suggest repeating the prioritization with a small amount of requirements and subsequently adjust the weights in such a way that the prioritization result corresponds to the manual one.

Altogether, the results from the two case studies suggest that BPL is an appropriate technique to prioritize a few dozen requirements as they typically occur in small product software companies although we cannot generalize from the case studies due to the limited number of replication. In such an environment, the technique’s overall prioritization process quality, considering the quality of the process itself and the quality of its results, seems to be higher than that of Wieger’s technique. BPL could help small software product companies without a formal prioritization process to systemize it. The best results are expected when requirements are compared that are similar in terms of development costs. The technique was used by one single person. It remains open if it is also suitable with a group of people performing a prioritization as e.g. in the situation mentioned in the Edmond case study where three people are in charge of prioritizing the high-level requirements.

However, the case studies also revealed some limitations of the technique. First of all, BPL does not consider dependencies between requirements. Instead, the user has to keep them in mind while prioritizing or refining the prioritization list afterwards as also suggested in the requirements selection phase of the release planning as also indicated in the reference framework for software product management [3]. In addition, due to the simple structure there might be a tendency to base the prioritization

just on one single criterion, such as importance rather than consider other factors such as costs, penalty and risk.

In terms of scalability, the first case study revealed that a pair-wise comparison of 68 requirements can already be quite tiring and lead to mistakes in terms of the comparison. Balancing the binary tree and incorporating other BST optimization techniques [11] could reduce the number of comparisons necessary. However, we expect BPL not to be practicable for numbers much more than 100 requirements.

5 Conclusion and Further Research

The research question, as proposed in section 1, states:

How can BPL be applied as a requirement prioritization technique in small product software companies and how reliable are its results?

We conducted case studies in two small Dutch product software companies to validate the applicability of the technique. The product managers of these companies used the technique to prioritize a few dozen low-level requirements. To assess the reliability of the results they did the same with another well-known prioritization technique, the Wiegers matrix. Subsequently, the results of the two techniques were compared in terms of time consumption, ease of use and subjective reliability of the results. In both cases, BPL scored higher than Wiegers's technique in all aspects of the comparison.

We can conclude that BPL is a suitable technique for prioritizing medium amounts of requirements and could especially help smaller software product companies to formalize their requirements prioritization process. This finding complements research performed by Ahl [8] who compared BPL with other techniques in an experiment with a relatively small amount of requirements. BPL seems especially applicable when applied for prioritizing low-level requirements of similar size.

Some limitations of the technique that became apparent are (1) the missing consideration of dependencies between requirements and (2) the fact that the BPL prioritization might have the tendency to only be based on one criterion, as for instance benefit, rather than considering other factors (e.g. costs, penalty, and risk) as well. This effect might, however, be mitigated by improving the tool user-interface and making the user aware that he should base his comparison of two requirements on a number of criteria.

Besides the limitation that two case studies are not a large base of generalizability, they only investigated the applicability of the technique in prioritization performed by one person. Further research should examine if BPL can be used in a group prioritization, e.g. in situations that require a discussion between different stakeholders. It is likely that techniques making the prioritization inputs more explicit, such as the Wiegers matrix, are more appropriate for this kind of prioritization since they provide a ground for discussion.

We have already referred to the limitation concerning the comparability of requirements with considerably different costs. Further research could investigate what possibilities exist to further mitigate this limitation. One possibility would be to use BPL to explicitly prioritize requirements in terms of a number of factors, such as benefit, penalty, costs and risk, and weight these sub-results in order to compute an overall prioritization.

References

1. Xu, L., Brinkkemper, S.: Concepts of product software. European Journal of Information Systems 16(5), 531–541 (2007)
2. Berander, P., Khan, K.A., Lehtola, L.: Towards a Research Framework on Requirements Prioritization. In: Proceedings of the Sixth Conference on Software Engineering Research and Practise in Sweden, pp. 39–48 (2006)
3. van de Weerd, I., Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., Bijlsma, L.: Towards a reference framework for software product management. In: Proceedings of the 14th International Requirements Engineering Conference, pp. 312–315 (2006)
4. Augustine, S.: Managing Agile Projects. Prentice Hall, New Jersey (2005)
5. Racheva, Z., Daneva, M., Buglione, L.: Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products. In: Proceedings of the Second International Workshop on Software Product Management 2008, Barcelona, pp. 49–58 (2008)
6. Karlsson, J., Wohlin, C., Regnell, B.: An evaluation of methods for prioritizing software requirements. Information and Software Technology 39(14-15), 939–947 (1997)
7. Wiegers, K.: First things first: prioritizing requirements. Software Developmen 7(9), 48–53 (1999)
8. Ahl, V.: An Experimental Comparison of Five Prioritization Methods. Master's Thesis. Department of Systems and Software Engineering, Blekinge Institute of Technology, Ronneby (2005)
9. Karlsson, J., Ryan, K.: A cost-value approach for prioritizing requirements. IEEE Software 14(5), 67–74 (1997)
10. Knuth, D.: The Art of Computer Programming, vol. 3. Addison-Wesley, Reading (1997)
11. Knuth, D.: Optimum binary search trees. Acta Informatica 1(1), 14–25 (1971)
12. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Communications of the ACM 18(9), 509–517 (1975)
13. Bell, J., Gupta, G.: An evaluation of self-adjusting binary search tree techniques. Software: Practice and Experience 23(4), 369–382 (1993)
14. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. Management Information Systems Quarterly 28(1), 75–106 (2004)
15. Yin, R.K.: Case study research: Design and methods. Sage, Thousand Oaks (2009)
16. Wohlin, C., Wesslen, A.: Experimentation in software engineering: an introduction. Kluwer, Norwell (2000)
17. Smith, J.D.: Design and Analysis of Algorithms. PWS-KENT, Boston (1989)
18. Young, R.R.: Recommended requirements gathering practices, CrossTalk, pp. 9–12 (April 2002)
19. Herrmann, A., Daneva, M.: Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research. In: Proceedings of the 16th IEEE International Requirements Engineering Conference, pp. 125–134 (2008)
20. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice Hall, Upper Saddle River (2001)
21. JIRA Bug tracking, issue tracking and project management software, http://www.atlassian.com/software/jira/
22. McConnell, S.: Rapid Development: Taming Wild Software Schedules, 1st edn. Microsoft Press, Redmond (1996)