SOFTWARE REQUIREMENT PRIORITIZATION USING AGING PROCESS

Karthik.S
PG Scholar
Department of CSE
SNS College of Technology
karthik90720@gmail.com

Kamalraj.R
Assistant professor
Department of CSE
SNS College of Technology
mailtokamalraj@yahoo.cm

Karthik.S
DEAN & professor of CSE
Department of CSE
SNS College of Technology
profskarthik@gmail.com

***Abstract*: Software engineering is a collection of techniques, methods and tools that contribute to the production of a system of high-quality software with a given budget before a deadline. The requirements are a matter of several activities that must be done in the phase of software development. There are many discussions / constraints that must be taken into account in determining priority needs before making decisions. these are used in the allocation of resources requirement of risk management and quality management. Decide which among a set of requirements that must be considered first and in what order is a strategic process in software development. This task is commonly known requirement of priority. This task is commonly known as priority requirements. This paper describes a requirements prioritization method called case-based Rank (CBRank), which combines stakeholder preferences with project requirements order approximations calculated using techniques of machine learning and aging process bringing promising benefits. Positioning relative to CBRank requirements prioritization methods state-of-the-art is proposed, with a discussion of the advantages and limitations of the method. Other features of the CBRank method worth investigating are its ability to support coordination between different actors through negotiation. In addition, the potential benefits of integrating CBRank with other techniques such as planning game or AHP merit further analysis.**

Keywords: Requirements management, requirements prioritization, machine learning, aging.

1. INTRODUCTION:

The problem of prioritizing software amounted to classify a set of desired features and software features designed along one or several concerns such as commercial aspects (e.g., competition on the market or regulations, customer satisfaction) or technical (e.g., costs and development risks). Priorities requirements plays a crucial role in software development, and in particular allows the planning software releases, combining strategies for the budget management and planning, and market strategies. It is, in fact, considered as a complex multi-criteria decision process. More specifically, the first step is concerned with selection of the most appropriate prioritization criterion based on specific objectives, such as reducing development costs or limit overhead buging. Identifying the needs of the attributes in the second step is carried out so as to define a variable Ranking functions on the requirements. For example, with reference to the objective of reducing development costs and choice of "cost of development " as a classification of target criterion, requirement attributes such as the estimated number of "lines of code" or "components" suitable.

The third step, namely the acquisition of attribute values on all the requirements, generally represents the most consuming task in the process of setting priorities, because it relies on the availability of specialized knowledge or triggering feedback from stakeholders. Since an objective criterion could be encoded by multiple attributes and each attribute induces a ranking of requirement, the fourth step is concerned with the composition of the different attributes rankings based on a global order corresponding to the target criterion. This composition is generally defined terms of a weighted aggregation scheme.

The widespread use of such a process model for requirements priorities is also confirmed by a survey review which examined over 200 articles in the field of requirements priorities for benefits and costs criteria. The assumption underlying the analyzed approaches is that the criteria for classification, requirements attributes, and how to call in case of multi criteria classification may be defined independently of the nature of the current set of requirements being evaluated. In other words, they adopt an ex ante perspective on the prioritization of requirements problem that prevents the use of the available knowledge on the scope of the project. However, ex-post perspective will allow the exploitation of this knowledge a prioritization process based on the actual set conditions assessment and lead to another performing. Namely, the project participants are required for comparison by the current pair.

The analytic hierarchy process (AHP) can be regarded as the reference method among those likely to are based on the paradigm case. In this method, the classification criteria are defined on an assessment of the relative priority between two requirements, the ex-pressed by the project stakeholders. This assessment includes -pass all possible pairs of needs. The effort required by the human evaluator preferences when pairs are caused increases rapidly with the number of requirements for the number of pairs grows quadratic. This makes it difficult to use AHP with large sets of requirements, a problem that is usually treated by define ad hoc heuristics to decide when the pair process revealed preferences may be arrested without compromising the accuracy of the resulting This ranking.1 Perhaps one of the main reasons for the approaches being less ex post commonly used approaches ex ante needs practices prioritization.

Many organizations have software seniors were developed many years ago and have been continuously modified and expanded until today .Although they are essential to the survival of organizations, they cannot be removed. However, to maintain the old software updates , maintenance costs tend to increase from year to year due " Change the software " goes the more difficult to change software. In other words, the software aging becomes more complex, unreadable and unchangeable over time. To successfully maintain a great system for many years, organizations need to keep software modules not to become too complicated. They need to keep improving the maintainability of each module cleaning codes spaghetti, reconfiguring the module structures, recasting the functional management of the logic and data, etc. For example, functional strength can be gained by re-engineering a large module into several smaller modules with more explicit interfaces. Therefore, in case a charge must add functions to a large system, he / she has several options. For example, to make some changes in several modules, throwing several modules and overhaul new modules, then restructuring modules add functions, and so on. Indeed, it is often very difficult to say who will be the best choice. Make some changes in the modules can be cheaper than redesigning modules However, the future maintenance costs can become more expensive in this case. Companies want some criterion or a guide that can help the above decision

In software engineering, software implementation is as Observed Exhibiting a behaviour That Closely Resembles human aging. Like people, software gets too old and similar to human we

Prevent aging, we can order icts Understand causes and take steps to limit icts effects. Two kinds of software aging by David Parnas: 1) Caused by the failure of the product's to adapted with the dynamic of the environment and 2) is the result That of the exchange are made. As a logical product, software is not getting older physically, purpose in Some Circumstances the relevance and importance of the software is getting lesser and lesser to icts environment. Malthus, this is the phenomena of getting older. Unlike human aging, software aging function can be formulated by falling, failure, cost, technology and etc. The way software is built and the natural

That software can be modified and updated, the requirement changes in dynamic environment, give Flexibility and enable it to stay 'young'.

2. RELATED WORKS:

2.1 THE CASE-BASED RANK METHOD:

CBRank The method is based on a framework, first introduced in [7] and [8] , which supports decision-making for the control a set of elements , for example , the characteristics of products or software requirements . The framework provides an iterative ranking process that can handle single and multiple human decision makers (stakeholders) and the different classification criteria. A feature of this framework is the use of machine learning to reduce the tripping force, that is, the amount of the information required by stakeholders to achieve classification of a given level of quality. Besides the problem of requirements priorities the framework been applied to the problem of prioritized test cases software testing. To illustrate CBRank, we first define a set of base concepts that help describe the process of setting priorities, then we introduce the specific machine learning techniques, it is based in terms of an algorithmic procedure that we ask an example toy realize how intuitive the algorithm works.

1. Pair of sampling. An automatic procedure selects from all the needs of a set of sampled Requirements together with the relative preference is unknown (that is, non - ordered pair requirement as defined in (1)), a policy based sampling. A sampling policy may be a random selection or it may take into account rankings calculated in the previous iteration.

2 .Triggering priority. It takes the collection he required sample pairs produced by the pair

Sampling in the step input and produces as output a set ordered pairs required on the basis of the

Priorities expressed by a decision maker.

3 .Learning priority. Given a partial release of the priority stakeholders and possibly a set of ranking Functions, the learning algorithm generates a approximation of the unknown preferences and then the corresponding approximate Rank for requirements.

The approximate rank, the output of the process represents an approximation of the correct classification and may become the input for a new iteration of the process. If the result of the step of learning is considered accurate enough (or time input runs preferences), the iterations stop and the process gives the final approximation rank (Final rank approximate) output. Note that the first and third steps of the method are automatic. Regarding the second step, we assume, for simplicity that the revelation of preferences is monotone (ie, the decision maker does not evaluate the same pair twice). Using the concepts presented in the previous section, the whole process of prioritization can be designed as an approximation problem where, given a finite set of requirements. The objective is twofold minimize the biasing force, while reducing the disagreement

Agreement between the target (K) and the approximate rank (H).

2.2 The Priority Learning Techniques:

Management software products, there are several sub-processes. First, there is the portfolio management strategy where product development is defined on the basis of market information and business partners. In the road map of product (or TRM), themes and key assets portfolio products are identified and the construction of the roadmap are created. Requirements into software requirements candidate management of a product are collected and organized. Finally, the activity of release planning, requirements are prioritized and selected for publication, after which the launch of the software product can be prepared. Thus, a key step in planning is the release requirements priorities. The method is capable of stimulating produce very accurate prediction of the rank by linearly combination of partial orders that can be modelling accurate.

Prioritize methods guide decision makers in their task analysis needs to assign numbers or symbols reflecting their importance. A-prioritizing its session may consist of three consecutive steps:

(1) The preparation stage where a person structures requirements according to the principle of the hierarchy methods to use. A team and a team leader for the session is selected and provided all the necessary information.

(2) The execution phase where decision makers are the prioritization of real needs in using information they received in the previous step. Evaluation criteria must be approved by the team before the step execution is triggered.

(3) The presentation phase where the results of the implementation are presented for the persons concerned. Some prioritization methods involve different types of calculations that must be made before the results can be presented.

2.3. The analytic hierarchy process (AHP)

The analytic hierarchy process (AHP) is a decision method. Using AHP to prioritize software requirements is to compare all the unique pairs of requirements to determine which one is the highest priority, and extent. In a software project comprising n requirements, n · (n 1 1) / 2 Pairwise comparisons are therefore required by a decision maker. Firstly PLA is a demanding method because of the dramatic increase in some number of comparisons required for the pair when the number of requirements develops. On the other hand AHP is very confident worthy since the huge amount of redundancy in the pair wise comparisons makes the process relatively insensitive to misjudgements. Another advantage is that the resulting and the priorities are based on a relative scale, which provides useful needs assessments. Prioritize the needs of software using AHP involves the three stages of a ranking session (for a Comprehensive description of AHP):

(1) Preparation, describe all the unique pairs of requirements.

(2) Implementation, comparing all pairs described Requirements for using the scale.

(3) Presentation, use the "Average standard the column method (based on comparison of pairs) to estimate the relative priority of each requirement. Calculate the ratio of the coherence pairwise comparisons using methods provided by AHP. The consistency ratio is a reliability of the resulting priorities, and so also estimate errors of judgment in comparisons.

2.4. Hierarchy AHP

The most generalized requirements are placed at the top hierarchy and more specific requirements lower levels. Hierarchies are a common structure for all day use of AHP. But to separate the hierarchical requirements the structure of the flat structure requirements described above, we use the name of the PLA hierarchy in this document.

Using the AHP hierarchy also involve three stages of a session priority:
(1) Preparation, describe all the unique pairs of need at the same level in the hierarchy. Note that not all requirements are compared in pairs to each other, but only those who are at the same level.
(2) Implementation, comparing all pairs described need using the scale.
(3) As presentation do the same thing for AHP at each level hierarchy. Priorities are then propagated down hierarchy.

PLA hierarchy has characteristics similar to AHP .Use of a hierarchical structure reduces the required number decisions, but also the amount of redundancy. Thus, it is more sensitive to errors in judgment that AHP.

2.5 Ranking

As digital awarded, the ranking is based on an ordinal scale, but the requirements are classified without links in the row. This means that the largest re - requirement is ranked 1 and the least important is ranked (for n requirements). Each row has a unique requirement (with respect to the digital work), but it is not possible to see the relative difference between items classified (as in
AHP test). The ranked list of requirements can be obtained in a in various ways, such as by using the bubble sort algorithm or binary - search tree algorithms. Regardless of the sorting algorithm, classification seems to be more appropriate for one stakeholder, because it could be difficult to align several different the views of stakeholders. Nevertheless, it is possible to combine the different viewpoints by taking the average priority of each requirement, but this may lead to re - links requirements to avoid this method.

3. Conclusion:

The CBRank method follows the case-based paradigm for problem solving, according to which a solution to a new problem can be derived from (partial) examples of previous solutions to similar problems. In the context of requirements prioritization, these examples are elicited from project stakeholders as pairwise preferences on samples of the set of requirements to be prioritized, and used to compute an approximated ranking for the whole set. The machine learning technique exploited by the method has been presented, both with the help of an intuitive example and by describing the Rank Boost algorithm, which is implemented in the method. The prioritization process based on CBRank has been presented. A discussion of the method performance, which is defined in terms of trade-off between preference elicitation effort and ranking accuracy and of its domain adaptively, has been given, with the support of a set of different experimental measurements and of a case study. The experimental measurements were taken by applying CBRank to different prioritization problems, varying the number of requirements, the number of elicited pairs, and the accuracy of the computed ranking. Indicators for the statistical significance of the measurements have been provided. Finally, the CBRank method has been positioned with respect to state-of-the art approaches, with particular reference to the AHP method, which can also be considered an instance of the case-based problem solving paradigm. Differently from AHP, the CBRank method enables a prioritization process, even over 100 requirements, thanks to the exploitation of machine learning techniques that induce requirements ranking approximations from the acquired data.

4. REFERENCES

[1] V. Ahl, “An Experimental Comparison of Five Prioritization Methods—Investigating Ease of Use, Accuracy and Scalability,” master’s thesis, School of Eng., Blekinge Inst. of Technology, Aug.2005.

[2] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, “Supporting the Requirements Prioritization Process. A Machine Learning Approach,” Proc. 16th Int’l Conf. Software Eng. and Knowledge Eng. pp. 306-311, June 2004.

[3] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, “Exploiting Domain Knowledge in

Requirements Prioritization," Proc. 17th Int'l Conf. Software Eng. and Knowledge Eng., pp. 467-472, July 2005.

[4] P. Avesani, C. Bazzanella, A. Perini, and A. Susi, "Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques," Proc. 13th IEEE Int'l Conf. Requirements Eng., pp. 297-306, Sept. 2005.

[5] P. Avesani, S. Ferrari, and A. Susi, "Case-Based Ranking for Decision Support Systems," Proc. Fifth Int'l Conf. Case-Based Reasoning: Research and Development, pp. 35-49, 2003.

[6] P. Avesani, A. Susi, and D. Zanoni, "Collaborative Case-Based Preference Elicitation," Proc. Int'l Conf. Innovations in Applied Artificial Intelligence, pp. 752-761, 2005.

[7] P. Berander, K.A. Khan, and L. Lehtola, "Towards a Research Framework on Requirements Prioritization," Proc. Sixth Conf.Software Eng. Research and Practice in Sweden, Oct. 2006.

[8] M.Daneva and A. Herrmann, "Requirements Prioritization Based on Benefit and Cost Prediction: A Method Classification Frame-work," Proc. 34th Euromicro Conf. Software Eng. and Advanced Applications, pp. 240-247, 2008.

[9] A. Davis, Just Enough Requirements Management: Where Software Development Meets Marketing. Dorset House, 2005.