ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

Systematic Resulting Approach for Software Requirement Priority Reordering

M.H.Revathi
Guest Lecturer,
Department of Computer Science,
Thiruvalluvar University Constituent College of Arts and Science, Arakkonam

***Abstract*— We have some set of requirements that are considered for strategic process in software development lifecycle process. This resulting procedure used for ranking the results based on priority. In this paper requirements ranking will use Case-Based Ranking, in which combines project stakeholders preferences with requirements reordering accurately by computed through systematic resulting(learning) techniques, bringing best result as advantages. Initially, the human input preferences used to reduce the process, also to get the final ranked results. In other hand, the domain knowledge would be encoded some attributes that is defined in the requirement attributes can be utilized. The CBRank techniques associated prioritization process are handled. A set of data compared with a state-of-the-art prioritization method, providing accurate result, ability to give the support to the management effort and ranking accuracy in the domain knowledge. In proposed CBRank method that are supports and coordination among different stakeholders through negotiation. Further we analyze the "anytime prioritization method" for real time software project with resulting and ranking the requirements. Which is updating requirements ranking when new requirements are added, its update every time ranking and resulting is performed.**

Keywords*— *Case Based Ranking(CB Ranking); Analytical Hierarchy Process (AHP).

I INTRODUCTION

1.1 MACHINE LEARNING

Machine learning, a branch of artificial intelligence, concerns the construction and study of systems that can learn from data. For example, a machine learning system could be trained on email messages to learn to distinguish between spam and non-spam messages. After learning, it can then be used to classify new email messages into spam and non-spam folders.

The core of machine learning deals with representation and generalization. Representation of data instances and functions evaluated on these instances are part of all machine learning systems. Generalization is the property that the system will perform well on unseen data instances; the conditions under which this can be guaranteed are a key object of study in the subfield of computational learning theory.

Machine learning and data mining

These two terms are commonly confused, as they often employ the same methods and overlap significantly. They can be roughly defined as follows:

- Machine learning focuses on prediction, based on known properties learned from the training data.
- Data mining focuses on the discovery of (previously) unknown properties in the data. This is the analysis step of Knowledge Discovery in Databases.

The two areas overlap in many ways: data mining uses many machine learning methods, but often with a slightly different goal in mind. On the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy.

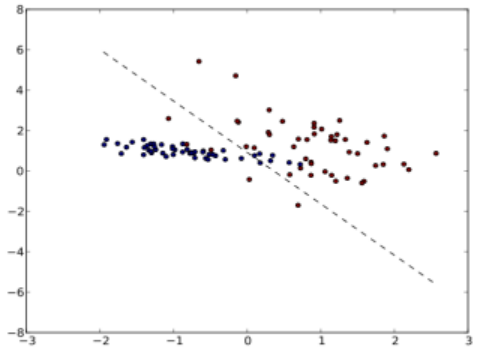


Figure 1 Linear-svm-scatterplot.svg

Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to reproduce known knowledge, while in Knowledge Discovery and Data Mining (KDD) the key task is the discovery of previously unknown knowledge.

Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

supervised methods, while in a typical KDD task; supervised methods cannot be used due to the unavailability of training data.

ARTIFICIAL INTELLIGENCE

Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is also an academic field of study. Major AI researchers and textbooks define the field as "the study and design of intelligent agents", where an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1955, defines it as "the science and engineering of making intelligent machines".

AI research is highly technical and specialized, and is deeply divided into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers. AI research is also divided by several technical issues. Some subfields focus on the solution of specific problems. Others focus on one of several possible approaches or on the use of a particular tool or towards the accomplishment of particular applications.

The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects. General intelligence (or "strong AI") is still among the field's long term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science, psychology, linguistics, philosophy and neuroscience, as well as other specialized fields such as artificial psychology.

REQUIREMENTS PRIORITIZATION

Requirement prioritization is used in Software product management for determining which candidate requirements of a software product should be included in a certain release. Requirements are also prioritized to minimize risk during development so that the most important or high risk requirements are implemented first. Several methods for assessing a prioritization of software requirements exist.

Complex decision-making situations are not unique to software engineering. Other disciplines, such as psychology, and organizational behavior have studied decision- making thoroughly. Classical decision-making models have been mapped to various requirements engineering activities to show the similarities. A comprehensive overview of decision-making and decision support in requirements engineering. Current chapter primarily focuses on requirements prioritization, an integral part of decision-making. The intention is to describe the current body of knowledge in the requirements prioritization area. The quality of a software product is often determined by the ability to satisfy the needs of the customers and users. Hence, eliciting and specifying the correct requirements and planning suitable releases with the right functionality is a major step towards the success of a project or product. If the wrong requirements are implemented and users resist using the product, it does not matter how solid the product is or how thoroughly it has been tested.

Most software projects have more candidate requirements than can be realized within the time and cost constraints. Prioritization helps to identify the most valuable requirements from this set by distinguishing the critical few from the trivial many. The process of prioritizing requirements provides support for the following activities;

- For stakeholders to decide on the core requirements for the system.
- To plan and select an ordered, optimal set of software requirements for implementation in successive releases.
- To trade off desired project scope against sometimes conflicting constraints such as schedule, budget, resources, time to market, and quality.
- To balance the business benefit of each requirement against its cost.
- To balance implications of requirements on the software architecture and future evolution of the product and its associated cost.
- To select only a subset of the requirements and still produce a system that will satisfy the customer(s).
- To estimate expected customer satisfaction.
- To get a technical advantage and optimize market opportunity.
- To minimize rework and schedule slippage (plan stability).
- To handle contradictory requirements, focus the negotiation process, and resolve disagreements between stakeholders.
- To establish relative importance of each requirement to provide the greatest value at the lowest cost.

Requirements prioritization plays a crucial role in software development, and in particular it allows for planning software releases, combining strategies for budget management and scheduling, as well as market strategies.

It is, in fact, considered a complex multi-criteria decision making process.

State-of-the-art approaches tend to share a common model for this process, which consists of the following steps.

1. The definition of a target criterion for ordering.
2. The specification of requirement attributes to encode the chosen criterion.
3. The acquisition of specific values for those attributes, for all requirements under consideration.
4. The composition of rankings induced by requirement attributes associated to the target criterion.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

REQUIREMENTS MANAGEMENT

Requirements management is the process of documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. It is a continuous process throughout a project. A requirement is a capability to which a project outcome (product or service) should conform.

The purpose of requirements management is to ensure that an organization documents, verifies and meets the needs and expectations of its customers and internal or external stakeholders.

1.2 OBJECTIVES

The objective of this Machine Learning is to offer a detailed and comprehensive presentation of the CBRank method, providing:

1. A formal definition of the prioritization problem it solves,
2. An intuitive description of the machine learning technique it is based on and a characterization of the prioritization process supported by CBRank,
3. A comprehensive overview of the empirical measurements which have been performed to assess key properties of the method, and
4. A positioning of CBRank with respect to state-of-the art requirements prioritization methods.

- ❖ It produces of a highly secure, "bug free" system.
- ❖ Reduced no of comparison would be handled.
- ❖ High accuracy in prioritization resulting and ranking process.
- ❖ Less no of comparison takes only less time consumption.
- ❖ Systematically handles the ranking and resulting process.

1.5 PRIORITIZATION TECHNIQUES

The purpose of any prioritization is to assign values to distinct prioritization objects that allow establishment of a relative order between the objects in the set. In our case, the objects are the requirements to prioritize. The prioritization can be done with various measurement scales and types. The least powerful prioritization scale is the ordinal scale, where the requirements are ordered so that it is possible see which requirements are more important than others, but not how much more important. The ratio scale is more powerful since it is possible to quantify how much more important one requirement is than another (the scale often ranges from 0 - 100 percent). An even more powerful scale is the absolute scale, which can be used in situations where an absolute number can be assigned (e.g. number of hours). With higher levels of measurement, more sophisticated evaluations and calculations become possible. Below, a number of different prioritization techniques are presented.

1.5.1 ANALYTICAL HIERARCHY PROCESS (AHP)

The Analytic Hierarchy Process (AHP) is a systematic decision-making method that has been adapted for prioritization of software requirements. It is conducted by comparing all possible pairs of hierarchically classified requirements, in order to determine which has higher priority, and to what extent (usually on a scale from one to nine where one represents equal importance and nine represents absolutely more important). The total number of comparisons to perform with AHP are $n \times (n-1)/2$ (where n is the number of requirements) at each hierarchy level, which results in a dramatic increase in the number of comparisons as the number of requirements increases. Studies have shown that AHP is not suitable for large numbers of requirements. Researchers have tried to find ways to decrease the number of comparisons and variants of the technique have been found to reduce the number of comparisons by as much as 75 percent.The result from a prioritization with AHP is a weighted list on a ratio scale.

1.5.2 IMPORTANCE

When prioritizing importance, the stakeholders should prioritize which requirements are most important for the system. However, importance could be an extremely multifaceted concept since it depends very much on which perspective the stakeholder has. Importance could for example be urgency of implementation, importance of a requirement for the product architecture, strategic importance for the company, etc. Consequently, it is essential to specify which kind of importance the stakeholders should prioritize in each case.

1.5.3 PENALTY

It is possible to evaluate the penalty that is introduced if a requirement is not fulfilled . Penalty is not just the opposite of importance. For example, failing to conform to a standard could incur a high penalty even if it is of low importance for the customer (i.e. the customer does not get excited if the requirement is fulfilled). The same goes for implicit requirements that users take for granted, and whose absence could make the product unsuitable for the market.

1.5.4 COST

The implementation cost is usually estimated by the developing organization. Measures that influence cost include: complexity of the requirement, the ability to reuse existing code, the amount of testing and documentation needed, etc. Cost is often expressed in terms of staff hours (effort) since the main cost in software development is often primarily related to the number of hours spent. Cost by using any of the techniques, but also by simply estimating the actual cost on an absolute or normalized scale.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

1.5.4 TIME

As can be seen in the section above, cost in software development is often related to number of staff hours. However, time (i.e. lead time) is influenced by many other factors such as degree of parallelism in development, training needs, need to develop support infrastructure, complete industry standards, etc.

1.5.5 RISK

Every project carries some amount of risk. In project management, risk management is used to cope with both internal (technical and market risks) and external risks (e.g. regulations, suppliers). Both likelihood and impact must be considered when determining the level of risk of an item or activity. Risk management can also be used when planning requirements into products and releases by identifying risks that are likely to cause difficulties during development .Such risks could for example include performance risks, process risks, schedule risks etc.. Based on the estimated risk likelihood and risk impact for each requirement, it is possible to calculate the risk level of a project.

Volatility

Volatility of requirements is considered a risk factor and is sometimes handled as part of the risk aspect. Others think that volatility should be analyzed separately and that volatility of requirements should be taken into account separately in the prioritization process. The reasons for requirements volatility vary, for example: the market changes, business requirements change, legislative changes occur, users change, or requirements become more clear during the software life cycle Irrespective of the reason, volatile requirements affect the stability and planning of a project, and presumably increase the costs since changes during development increase the cost of a project.

Supporting the Requirements Prioritization Process a Machine learning approach

Requirements prioritization has been pointed out as a relevant research area in requirements engineering, calling for the definition of effective methods and techniques that enable to rank a whole set of requirements, according to relevant criteria, such as business goals (e.g. customer value) or technical features (e.g. development cost) . Several approaches have been recently proposed which adopts a common model for the requirements prioritization process, based on the following three steps:

(i) selection of one or more prioritization criteria (or prioritization features) among business goals and technical features;

(ii) acquisition of a requirements ordering according to a specific criterion from one or more stakeholders (e.g. customers, users, project manager);

(iii) Composition of the acquired orderings into a final one based upon an appropriate composition schema. These approaches tend to focus on how to choose the most relevant criteria and on how to combine them, while giving minor emphasis to the acquisition of the ranks according to a given criterion.

In our approach we exploit machine learning techniques to reduce the elicitation effort by approximating part of the pair wise preferences. The approximation step computes an estimate of unknown preferences looking at the other ranks acquired according to predefined prioritization criteria. Moreover, we adopt a Boolean metrics to lower the human effort associated to the requirements evaluation and we prove that it can be effective as much as multi values metrics, as far as a large set of requirements has to be prioritized.

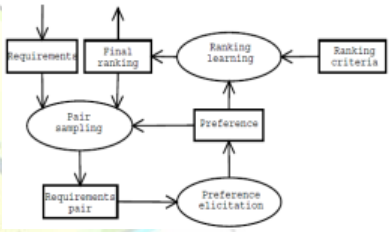


Figure 2: the basic iteration of the requirements prioritization process.

Figure 2, depicts the basic process that the evaluator undertakes. The types of data involved in the process are depicted as rectangles, namely: **Requirements** represent data in input to the process, that is the finite collection of requirements that have to be ranked; **Requirements pair** is a pair of candidate requirements whose relative preference is to be specified; **Preference** is the order relation between two alternative requirements elicited from the stakeholder.

The preference is formulated as a Boolean choice on a pair; **Ranking criteria** are a collection of order relations that represents ordering induced by other criteria (e.g. the cost for the realization of the requirements, the estimated utility) defined on the initial set of requirements; **Final ranking** represents the resulting preference structure on the set of requirements. This final ranking which results from the output of the process represents an approximation of the exact ranking. Notice that this ranking may become the input to a further iteration of the process.

The steps of the basic process iteration are depicted as ovals in Figure 1, they are:

1. Pair sampling

An automated procedure selects from the requirements repository a pair of requirements and submits it to the stakeholder who can judge their relative priority. Notice that in this step, the selection of a pair takes into account information on the current available rankings (this information is stored in the data Preference, see the arrows between Preference and Pair sampling in Figure 1);

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

2. **Preference elicitation**

This represents a mixed initiative step in the process: given a pair of requirements the stakeholder chooses which one is to be preferred with respect to the current criterion;

3. Ranking learning

Given a partial elicitation of the user preferences, a learning algorithm produces an approximation of the unknown preferences and a ranking of the whole set of requirements is derived.

II LITERATURER SURVEY

The objective of this paper is to conduct a controlled experiment with the three requirements prioritization techniques: Numerical Assignment (NA), Analytic Hierarchy Process (AHP) and Extensive Numerical Assignment (ENA), each based on ordinal, ratio and interval scales respectively. NA and AHP are widely used traditional requirements prioritization techniques. ENA is a novel technique introduced by the authors, which acknowledges the uncertain and incomplete nature of human judgment about requirements priorities, which are in turn uncertain guesses about the upcoming product. The aim of the experiment is to examine the three techniques using various objective and subjective measures like number of decisions, time consumption, ease of use, attractiveness, scalability and reprioritizability. The experiment was executed by prioritizing the requirements of a university website system with students as participants in the experiment. The results of the experiment proved that ENA transcends NA and AHP.

A. Supporting the Requirements Prioritization Process. A Machine Learning Approach

Requirements prioritization plays a key role in the requirements engineering process, in particular with respect to critical tasks such as requirements negotiation and software release planning. This paper presents a novel framework which is based on a requirements prioritization process that interleaves human and machine activities, enabling for an accurate prioritization of requirements. Similarly to the Analytic Hierarchy Process (AHP) method, our framework adopts an elicitation process based on the acquisition of pair wise preferences. Differently from AHP, where scalability is a big issue, the framework enables a prioritization process even over a large set of requirements, thanks to the exploitation of machine learning techniques that induce requirements ranking approximations at run time, and to the use of a Boolean metrics. Moreover the new approach allows reducing the bias of a dominance hierarchy, a strategy introduced by AHP to deal with the scalability issue. The paper describes also a methodology for the experimental evaluation of the framework and discusses the results of a first set of experiments designed on a real case study which shows that a high accuracy in the final ranking can be obtained within a limited elicitation effort.

B. Facing Scalability Issues in Requirements Prioritization with Machine Learning Techniques

Case-based driven approaches to requirements prioritization proved to be much more effective than first-principle methods in being tailored to a specific problem, that is they take advantage of the implicit knowledge that is available, given a problem representation. In these approaches, first-principle prioritization criteria are replaced by a pairwise preference elicitation process. Nevertheless case-based approaches, using the analytic hierarchy process (AHP) technique, become impractical when the size of the collection of requirements is greater than about twenty since the elicitation effort grows as the square of the number of requirements. We adopt a case-based framework for requirements prioritization, called case-based ranking, which exploits machine learning techniques to overcome the scalability problem. This method reduces the acquisition effort by combining human preference elicitation and automatic preference approximation. Our goal in this paper is to describe the framework in details and to present empirical evaluations which aim at showing its effectiveness in overcoming the scalability problem. The results prove that on average our approach outperforms AHP with respect to the trade-off between expert elicitation effort and the requirement prioritization accuracy.

C. Case Based Ranking For Decision Support Systems

Very often a planning problem can be formulated as a ranking problem: i.e. to find an order relation over a set of alternatives. The ranking of a finite set of alternatives can be designed as a preference elicitation problem. While the case-based preference elicitation approach is more effective with respect to the first principle methods, still the scaling problem remains an open issue because the elicitation effort has a quadratic relation with the number of alternative cases. In this paper we propose a solution based on the machine learning techniques. We illustrate how a boosting algorithm can effectively estimate pair wise preferences and reduce the effort of the elicitation process. Experimental results, both on artificial data and a real world problem in the domain of civil defense, showed that a good trade-off can be achieved between the accuracy of the estimated preferences, and the elicitation effort of the end user.

III ARCHITECTURE OF MACHINE LEARNING SYSTEM USING THE CASE-BASED RANK METHOD

3.1 INTRODUCTION

The CBRank method rests on a framework, which supports decision-making for ordering a set of items, e.g., product features or software requirements.

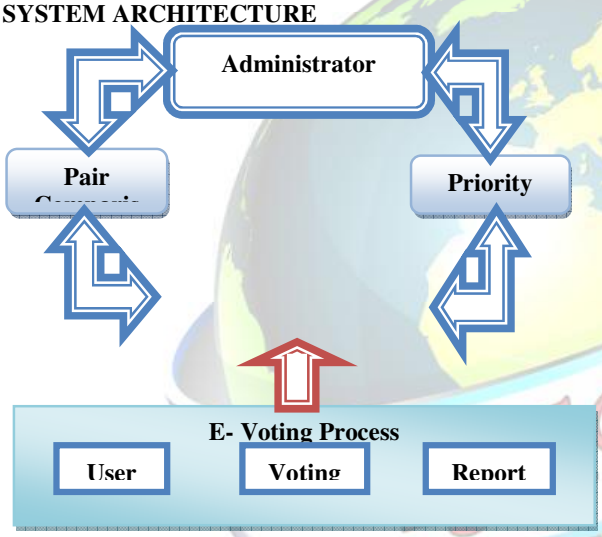
ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

The framework provides an iterative prioritization process that can handle single and multiple human decision makers (stakeholders) and different ordering criteria. A peculiarity of this framework is the use of machine learning to reduce the elicitation effort, that is, the amount of information required from stakeholders, for achieving rankings of a given quality degree. Besides the problem of requirements prioritization, the framework has been applied to the problem of prioritizing test cases in software testing.

In order to illustrate CBRank, we first define a set of basic concepts that help describe the prioritization process, then we introduce the specific machine learning techniques it is based on in terms of an algorithmic procedure that we apply to a toy example to give an intuitive account of how the algorithm works.

3.2 SYSTEM ARCHITECTURE



3.3 EXISTING SYSTEM

The Analytical Hierarchy Process (AHP) can be considered the reference method among those which are based on the case-based paradigm. In this method, the ranking criteria are defined upon an assessment of the relative priority between a couple of requirements, expressed by project stakeholders. This assessment encompasses all possible pairs of requirements. The effort required by the human evaluator when pair preferences are elicited grows rapidly with the number of requirements since the number of pairs grows quadratically. This makes AHP difficult to use with large sets of requirements, a problem that is typically dealt with by defining ad hoc heuristics for deciding when the pair preference elicitation process can be stopped without compromising the accuracy of the resulting ranking. This may be a main reason for ex-post approaches being less commonly used than ex-ante approaches in requirements prioritization practices.

Disadvantages

1. More time consumption of performing prioritization tasks.
2. Less accuracy in resulting and ranking.
3. Large no of comparison would be handled.

3.4 PROPOSED SYSTEM

We propose a method called Case-Based Ranking(CBRank). First, it allows for combining sets of preferences elicited from human decision makers with sets of preferences, which are automatically computed through machine learning techniques. These techniques exploit knowledge about (partial) orders of the requirements that may be encoded in the description of the requirements itself (i.e., in terms of the actual requirement attributes), thus enabling what we call domain adaptively. This accounts for the straightforward applicability of CBRank to different application domains and for the fact that the accuracy of machine-estimated ranking increases with the level of significance of the encoded domain knowledge.

Second, CBRank is organized according to an iterative schema which allows for deciding when to stop the elicitation process on the basis of a measure of the tradeoff between the elicitation effort and the accuracy of the resulting ranking. With a reasonable effort, the method can be applied up to 100 requirements. The objective of this paper is to offer a detailed and comprehensive presentation of the CBRank method, providing:

1. A formal definition of the prioritization problem it solves, an intuitive description of the machine learning technique it is based on and a characterization of the prioritization process supported by CBRank,
2. A comprehensive overview of the empirical measurements which have been performed to assess key properties of the method, and a positioning of CBRank with respect to state-of-theart requirements prioritization methods.

Advantages

1. It produces of a highly secure, “bug free” system.
2. Reduced no of comparison would be handled.
3. High accuracy in prioritization resulting and ranking process.
4. Less no of comparison takes only less time consumption.
5. Systematically handles the ranking and resulting process.

3.5 THE PRIORITIZATION PROCESS

The CBRank requirements prioritization process interleaves human activities with machine computation. The process is sketched in Fig. 1, where three steps are represented as rounded corner rectangles. The basic artifacts in input and

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

output (see dashed arrows) are: the set of Requirements (Req), the decision maker's Priorities (Φγ), the set of Ranking Functions (F), encoding the requirement attributes, and the Approximated Rank (^Hγ) or the Final Approximated Rank (H) when Γ corresponds to the last process iteration. Additional artifacts are produced by internal activities of the process's steps, namely, the set of Sampled Requirements Pairs that is the set of requirements pairs for which the end user preference is unknown and that have been selected for the following priority elicitation.

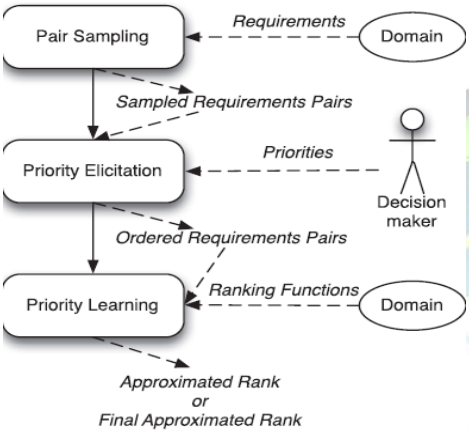


Figure: Basic steps of the requirements prioritization process in CBRank

1. Pair Sampling. An automated procedure selects from the set of Requirements a set of Sampled Requirements Pairs whose relative preference is unknown (i.e., Unordered Requirement Pairs, as defined in (1)), according to a sampling policy. A sampling policy can be a random choice or it may take into account the rankings.
2. Priority Elicitation. This takes the collection of Sampled Requirements Pairs produced by the Pair Sampling step in input and produces as output a set of Ordered Requirements Pairs on the basis of the Priorities expressed by a decision maker.
3. Priority Learning. Given a partial elicitation of the stakeholder priority and eventually a set of Ranking Functions, the learning algorithm produces an approximation of the unknown preferences and then the correspondent Approximated Rank for the requirements.

Algorithm 1. A sketch of the Rank Boost algorithm

IV IMPLEMENTATION OF MACHINE LEARNING APPROACH TO SOFTWARE REQUIREMENTS PRIORITIZATION

- Administrator
- Login Process
- Stakeholders Decision Making
- Pair Sampling
- Priority Elicitation and Learning

Administrator

This module provider by administrator with details of users. The administrator has to maintain the stake holder's information, who have involve in this process. Administrator performs the resulting and ranking method. Evaluate this expensive task by using the machine learning approach. It performs the pair s of requirements comparison of the data sets.

Login Process

The login page has very simple code and through this page we will collect user id and password and then send the data by form post method to another page where we will do the matching with the database. Users of an e-voting may be asked to decide which of the requirements "Graphical layout of the voting form" and "Getting feedback during the voting procedure" is more important.

Stakeholders Decision Making

First, it allows for combining sets of preferences elicited from human decision makers with sets of preferences, which are automatically computed through machine learning techniques. These techniques exploit knowledge the requirements that may be encoded in the description of the requirements itself (i.e., in terms of the actual requirement attributes).

Pair Sampling

This accounts for the straightforward applicability of CBRank to different application domains and for the fact that the accuracy of machine-estimated ranking. It supports decision-making for ordering a set of items, e.g., product features or software requirements. The framework provides an iterative prioritization process that can handle single and multiple human decision makers and different criteria. It take two different set of information and performs the pair of comparison with the current requirements.

Priority Elicitation and Learning

The human effort to input preference information can be reduced, while preserving the accuracy of the final ranking estimates. The machine learning to reduce the elicitation effort, that is, the amount of information required from stakeholders, for achieving rankings of a given quality degree. Besides the problem of requirements prioritization, the framework has been applied to the problem of prioritizing test cases in software testing. Output of the process, represents an approximation of the exact ranking and may become the input for a further iteration of the process. If the result of the learning step is considered accurate result.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

V PERFOMANCE EVALUATION OF REQUIREMENTS PRIORITIZATION

The results of a prioritization exercise must be used judiciously. Dependencies between requirements should be taken into consideration when choosing which requirements to include. Dependencies could be related to cost, value, changes, people, competence, technical precedence, etc.. Such dependencies might force one requirement to be implemented before another, implying that it is not possible to just follow the prioritization list. Another reason for not being able to solely base the selected requirements on the priority list is that when the priority list is presented to the stakeholders, their initial priority might have emerged incorrectly. This means that when the stakeholders are confronted with the priority list, they want to change priorities. This is a larger problem in techniques where the result is not visible throughout the process (e.g. AHP). The product may have some naturally built-in constraints. For example, projects have constraints when it comes to effort, quality, duration, etc.. Such constraints make the selection of which requirements to include in a product more complex than if the choice were solely based on the importance of each requirement.

A common approach to make this selection is to propose a number of alternative solutions from which the stakeholders can choose the one that is most suit able based on all implicit context factors By computerizing the process of selecting nominated solutions, it is possible to focus the stakeholders' attention on a relatively small number of candidate solutions instead of wasting their time by discussing all possible alternatives. In order to automate and to provide a small set of candidate solutions to choose from, it is necessary to put some constraints on the final product. For example, there could be constraints that the product is not allowed to cost more than a specific amount, the time for development is not allowed to exceed a limit, or the risk level is not allowed to be over a specific threshold.

To illustrate the different aspects, prioritization techniques, trade-offs between stakeholders, and combinations of prioritization techniques and aspects, an example of a prioritization situation is given. The method used in this example is influenced by a model proposed by Wiegers but is tailored. The example analyses 15 requirements (R1-R15) in a situation with three known customers. The analysis is rather sophisticated to show different issues in prioritization but still simple with a small amount of requirements. While many more requirements are common in industry, it is easier to illustrate how the techniques work on a smaller example. Each of the 15 requirements is prioritized according to the different aspects. Table 4.1 presents the aspects that are used in the example together with the method that is used to prioritize the aspect and from which perspective it is prioritized.

Table 4. 1. Aspects to Prioritize.

Aspect	Technique	Perspective	Prioritization
Strategic importance	AHP	Product Manager	
Customer importance	100-dollar / Top-ten	Customers	
Penalty	AHP	Product Manager	
Cost	100-dollar	Developers	
Time (7)	Numerical Assignment	Project Manager	
Risk (3)	Numerical Assignment	Requirements Specialist	
Volatility	Ranking	Requirements Specialist	

However, two clarifications are in order. First, numerical assignment for time (7) and risk (3) uses a different number of groups to show varying levels of granularity. The customer importance is prioritized both by the top-ten technique and the 100-dollar technique depending how much time and cost the different customers consider reasonable.

First, requirements R1 and R2 are requirements that are absolutely necessary to get the system to work at all. Hence, they are not prioritized by the customers but they are estimated when it comes to cost, risk, etc. since R1 and R2 influence these variables no matter what. This is a way of using the requirements triage approach. Further, two groups of requirements have been identified as having high dependencies (must be implemented together) and should hence be prioritized together. Requirements R3, R4, and R5 are grouped together as R345, and requirements R6 and R7 are grouped into R67.

The next step is to prioritize the importance of the requirements. In the case at hand, the three known customers and the product manager prioritize the requirements. Furthermore, these four stakeholders are assigned different weights depending on how important they are deemed by the company. This is done by using the 100-dollar test to get the relative weights between the stakeholder's presents the result of the prioritization. In the table, the three customers are denoted C1-C3 and the product manager is denoted PM.

Table 4. 2. Prioritization Results of Strategic and Customer Importance. Priority, P (RX) = RPC1 × WC1 + RPC2 × WC2

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

+ RPC3 × WC3 + RPPM × WPM, where RP is the requirement priority, and W is the weight of the stakeholder.

Requirement	C1 (0.15)	C2 (0.30)	C3 (0.20)	PM (0.35)	Priority:
R8	0.25	0.24	0.16	0.15	0.19
R9		0.07	0.14	0.03	0.06
R10	0.25	0.05	0.13	0.29	0.18
R11		0.05	0.01	0.02	0.02
R12		0.16	0.04	0.01	0.06
R13		0.05	0.16	0.02	0.05
R14	0.25	0.02	0.10	0.10	0.10
R15		0.03	0.04	0.05	0.03
R345			0.04	0.18	0.17	0.11
R67		0.25	0.29	0.04	0.16	0.1
Total:		1	1	1	1	1

As can be seen in this table, the different stakeholders have different priorities, and it is possible to combine their different views to an overall priority. The weights (within parenthesis after each stakeholder) represent the importance of each customer and in this case, the product manager is assigned the highest weight (0.35). This is very project dependent. In this case, the mission of this product release is to invest in long-term requirements and attract new customers at the same time as keeping existing ones. As also can be seen, C1 used the top-ten technique and hence the priorities were evenly divided between the requirements that this customer regarded as most important. The list to the far right presents the final priority of the requirements with the different stakeholders and their weights taken into consideration. This calculation is possible since a ratio scale has been used instead of an ordinal scale. The next step is to prioritize based on the other aspects.

Table 4. 3. Descending Priority List Based on Importance and Penalty (IP). $IP(RX) = RPI \times WI + RPP \times WP$, where RP is the requirement priority, and W is the weight of Importance (I) and Penalty (P).

Requirement	Importance	Penalty	IP	Cost	Time	Risk	Volatility
	(0.7)	(0.3)					
R1	1	1	1	0.11	3	1	2
R2	1	1	1	0.13	4	2	1
R8	0.19	0.2	0.20	0.07	1	3	7
R67	0.19	0.09	0.16	0.10	6	3	5
R10	0.18	0.01	0.13	0.24	2	3	11
R14	0.10	0.16	0.12	0.01	1	3	10
R345	0.11	0.02	0.08	0.03	3	2	8
R9	0.06	0.12	0.08	0.09	3	2	9
R15	0.03	0.17	0.08	0.05	5	1	4
R12	0.06	0.06	0.06	0.11	4	2	6
R11	0.02	0.14	0.06	0.02	3	1	3
R13	0.05	0.03	0.05	0.04	7	1	12
Total / Median:	3	3	3	1	3	2	

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

Table 4.3 shows a prioritized list of the requirements (based on IP). With this information there are two options:

1. pick prioritized items from the top of the list until the cost constraints are reached,
2. Analyze further based on other prioritized aspects, if prioritizations of additional aspects are available.
3. The example has two major constraints:
4. the project is not allowed to cost more than 65% of the total cost of the elicited requirements, and
5. The median risk level of the requirements included is not allowed to be higher than 2.5. Based on this, we first try to include the requirements with the highest IP. The result of this is presented in Table 4.4 where the list was cut when the sum of costs reached 65% of the total cost of elicited requirements.

Table 4. 4. Selected Requirements Based on IP and Cost.

Requirement	IP	Cost	IP/Cost	Time	Risk	Volatility
R1	1	0.11	9.09	3	1	2
R2	1	0.13	7.69	4	2	1
R8	0.20	0.07	2.80	1	3	7
R6 7	0.16	0.1	1.59	6	3	5
R10	0.13	0.24	0.54	2	3	11
Total / Median:	2.48	0.65	21.71	3	3	

Table 4.4 shows that we managed to fit within the cost constraints but could not satisfy the risk constraint. As a result, the project becomes too risky. Instead, another approach is taken to find a suitable collection of requirements. In this approach, we take the IP/Cost ratio into consideration. This shows which requirements provide most IP at the least cost. In this case, we try to set up a limit of only selecting requirements that have an IP/Cost-ratio higher than 1.0. The result is presented in Table 4.7. Table 4. 5. Selected Requirements Based on Cost and IP/Cost Ratio.

Requirement	IP	Cost	IP/Cost	Time	Risk	Volatility
R1	1	0.11	9.09	3	1	2
R2	1	0.13	7.69	4	2	1
R8	0.20	0.07	2.80	1	3	7
R67	0.16	0.1	1.59	6	3	5
R14	0.12	0.01	11.70	1	3	10
R345	0.08	0.03	2.71	3	2	8
R15	0.08	0.05	1.50	5	1	4
R11	0.06	0.02	2.94	2	1	3
R13	0.05	0.04	1.17	7	1	12
Total / Median:	2.73	0.56	41.19	3	2	

CONCLUSION

In this paper, we provided a detailed account of the CBRank method for requirements prioritization.

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 pair wise 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 tradeoff between preference elicitation effort and

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 4, Special Issue 4, March 2017

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.

FUTURE WORK

In Future work, we should address the non-monotonic case and more sophisticated pair sampling policies, possibly contributing to improving the effectiveness of the method in more complex real settings. Further we implement the CBRank method that are supports and coordination among different stakeholders through negotiation. Further we analyze the anytime prioritization method for real time software project with resulting and ranking the requirements. The requirements prioritization problem such as handling requirements dependencies and "anytime" prioritization, which is updating requirements ranking when new requirements are added, its update every time ranking and resulting is performed.

REFERENCES

[1] A. Aamodt and E. Plaza, "Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches," Artificial Intelligence Comm., vol. 7, no. 1, pp. 39-59, 1994.

[2] 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.

[3] Y. Akao, Quality Function Deployment: Integrating Customer Requirements into Product Design. Productivity Press, 1990.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] K. Beck, Extreme Programming Explained. Addison-Wesley, 1999.

[10] P. Berander and A. Andrews, "Requirements Prioritization," Eng. and Managing Software Requirements, A. Aurum and C. Wohlin, eds., Springer, 2005.

[11] 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.

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

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

[14] A. Finkelstein, M. Harman, S.A. Mansouri, J. Ren, and Y. Zhang, "A Search Based Approach to Fairness Analysis in Requirement Assignments to Aid Negotiation, Mediation and Decision Making," Requirements Eng., vol. 14, no. 4, pp. 231-245, 2009.

[15] Y. Freund, R.D. Iyer, R.E. Schapire, and Y. Singer, "An Efficient Boosting Algorithm for Combining Preferences," Proc. 15th Int'l Conf. Machine Learning, pp. 170-178, 1998.

[16] Y. Freund, R.D. Iyer, R.E. Schapire, and Y. Singer, "An Efficient Boosting Algorithm for Combining Preferences," J. Machine Learning Research, vol. 4, pp. 933-969, 2003.

[17] T. Gilb, Competitive Engineering: A Handbook for Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage. Elsevier, 2005.