

Situation-Oriented Evaluation and Prioritization of Requirements

Nimanthi L. Atukorala¹(✉), Carl K. Chang¹, and Katsunori Oyama²

¹ Department of Computer Science, Iowa State University, Ames, USA
{nimanthi, chang}@iastate.edu

² Department of Computer Science, Nihon University, Koriyama, Japan
oyama@cs.ce.nihon-u.ac.jp

Abstract. Evaluation and prioritization of requirements is one of the key aspects in requirements engineering. Although the existing studies in this area are greatly focused on addressing business goals such as development budget and deadlines of completion, we believe that human-centered concerns, including end-users' personal desires, goals, beliefs and constrained environment, must also weigh in. In this paper we present a new human-centered requirements evaluation and prioritization approach that effectively considers such concerns. The proposed method is based on the situation–transition structure introduced in our previous study that was used to elicit human-centered requirements. We illustrate the applicability of the proposed methodology through a case study.

Keywords: Evaluation · Prioritization · Requirements · Human-centered · Situation · Situation–transition structure

1 Introduction

High end-user satisfaction is always the ultimate goal of any software development project. Fulfilling the right set of end-users' requirements is the key to achieving this goal. However, finding that right set of end-users' requirements is challenging. Various requirements prioritization techniques have been introduced to support this task mainly based on aspects such as their importance for the overall system functionality, limits on budget, required time, potential risk, etc. [1].

Each individual human being is unique and has his or her own unique interests, desires, personal goals, etc. Moreover, their origin, culture and surrounding environmental factors lead them to have their own unique beliefs. Individuals' requirements are molded by these unique human factors and surrounding environmental factors. When it comes to software development, each individual end-user has a set of prioritized requirements imposed upon the particular software, and the satisfaction depends on whether or not those requirements are fulfilled by the software. It is also important to note that the requirements that have higher priority to one end-user can be entirely unimportant to another. Based on these observations, we believe that the requirements prioritization process can be made more effective by considering the information about the uniqueness of an individual's prioritization of those requirements [2]. In this paper we

propose a new approach to prioritizing requirements on an individual basis by considering user's human nature. The requirements are evaluated based on two main factors:

1. The importance of the requirement to a particular end-user (encoded as the Importance factor)
2. The likelihood that the requirement leads to an undesirable or risky situation (encoded as the Risk factor)

According to the proposed requirements prioritization approach a requirement will be assigned higher priority if its Importance factor is high and the Risk factor is low. The proposed requirements prioritization approach is an extension to our previous study on eliciting new human-centered requirements [3]. In a previous study we introduced a situation-transition structure that represent the human behavioral patterns in a computational format, and used it for requirements elicitation. The proposed requirements prioritization technique uses the information included in that situation-transition structure. In addition, we also extend the computerized system developed in the previous study so that it can be used to identify and prioritize requirements for each individual. It is important to note that the proposed requirements prioritization approach is not aiming to completely eliminate the use of other approaches, but instead provide additional information to the requirements analysts to make better decisions during the requirements prioritizing phase.

The rest of this paper is organized as follows: Sect. 2 reviews previous studies about requirements evaluation and prioritization. Section 3 summarizes the concept of situations, situation-transition structure and reviews the requirements elicitation procedure proposed in our previous study. Section 4 describes the proposed method in detail. In Sect. 5 a summary of a real-life case study is given for illustration. Section 6 includes a discussion and finally, Sect. 7 concludes the paper with some future work suggestions.

2 Related Work

The fast growing software development industry strives to release new products and their enhancements as frequently as possible. In order to reduce the problems starting from resource limitations and time constrains and maximize revenue, many software development companies pay more attention to the evaluation and prioritization of requirements than ever before. The rest of this section will discuss some of the common techniques as well as some studies in requirements evaluation and prioritization.

2.1 Direct Stakeholder Collaboration Based Approaches

Most of the traditional requirements prioritization techniques allow the stakeholders to prioritize the requirements according to their personal preferences and then form an agreement through identifying conflicts.

Numerical assignment is the most common approach where stakeholders are requested to place requirements in priority groups [4]. The number of priority groups

depends on the software development practice, but three groups: critical, standard, and optional are very common [5]. Win-win, also known as Theory-W [6], is a prioritization technique that allows each stakeholder to categorize requirements according to importance and potential risk, whereas the Top-Ten requirements approach [1] allows stakeholders to pick their top-ten requirements from a larger set without assigning an internal order between the requirements. In a 100-Dollar Test [1], each stakeholder is given 100 imaginary units (such as money, hours, etc.) to distribute among requirements and consider the ratio of the assignment as the scale of the priority. Although most of these techniques are simple and easily manipulated by stakeholders, each has its own limitations. One common problem in these approaches is that most of the stakeholders think that everything is rather critical. For example, a study [7] shows that stakeholders most likely consider 85 percent of the requirements as critical.

The proposed approach does not take direct responses from the end-users, but instead uses the observational data to make an unbiased decision on requirements prioritization. We believe that this will be an effective alternative method to discern the actual critical requirements of the end-users. In addition, each requirement will be assigned a unique priority, which will also be helpful for better decision making. Our proposed approach can be considered as a special case of win-win prioritization technique, where the requirements are prioritized based on their relative importance and the potential risks to the stakeholders.

2.2 Search Based Approaches

A well-known search based requirements prioritization approach applies Binary Search Trees (BST) where the prioritization is performed by constructing a binary search tree so that less important requirements are inserted to the left and more important ones to the right. A list of ranked requirements is obtained by using the bubble sort or binary search tree algorithms [1]. This allows stakeholders to compare the relative value of individual requirements and can be used to prioritize relatively large sets of requirements [6]. However, it is believed that original BST ranking is more suitable for a single stakeholder regardless of the sorting algorithm used for ranking since aligning several different stakeholders' views at the same time might be difficult [1].

Bebensee et al. [8] introduced Binary Priority List (BPT), a variation of BST structure, for prioritizing software requirements. The level of a requirement in the BPT represents its priority level. The top-most level has the highest priority and the priority decreases from top to bottom. Beg et al. [9] proposed requirements prioritization technique using B-tree, a self-balancing tree data structure aiming to reduce the number of comparisons between requirements pairs.

In comparison with BST, the situation-transition structure used in the proposed approach is a complex graph with set of nodes, directed edges and weight values on edges. The unique rank for each requirement is obtained using a graph traversal algorithm developed by modifying the depth first search algorithms. In other words, the procedure of the proposed approach is similar to BST ranking; however, the proposed approach can be used in both single and multiple end-user domains. Therefore, it is more powerful.

2.3 Machine Learning Based Approaches

Some researchers are focused on applying data mining and machine learning techniques to the requirements prioritization process in order to improve their effectiveness in large software development projects with multiple stakeholders. In [10], Duan et al. proposed a Pirogov approach that uses clustering techniques to place requirements into multiple independent clusters that capture the diverse and complex roles played by individual requirements. Stakeholders determine the relative value of each cluster and weight the importance of each clustering method. An objective function then generates prioritization decisions at the level of the individual requirement.

Tonella et al. [11] proposed an Interactive Genetic Algorithm (IGA) that includes incremental knowledge acquisition and combines it with the existing constraints, such as dependencies and priorities. This approach aims at minimizing the disagreement between a total order of prioritized requirements and the various constraints that are either encoded with the requirements or that are expressed iteratively by the user during the prioritization process. An interactive genetic algorithm was used to achieve such a minimization, taking advantage of interactive input from the user whenever the fitness function cannot be computed precisely based on the information available. The process terminates when a low disagreement is reached, the time out is reached or the allocated elicitation budget has been consumed.

Perini et al. [12] proposed a requirement prioritization method based on Case-Based Ranking (CBRank). CBRank originated from a framework that supports decision making on ordering a set of items which can handle single and multiple stakeholders and different ordering criteria. The prioritization is performed by considering the stakeholders' preferences and the approximated ordering of requirements predicted by machine learning techniques. Similarly, Babar et al. [13] proposed the PHandler, which is an intelligent requirements prioritizing technique that uses artificial neural networks to predict the priority of the requirements.

The algorithm used to generate the situation-transition structure in our proposed approach is a modified version of Chow-Liu Bayesian structure learning algorithm [14], which is popular in the machine learning area. In other words, the proposed approach also uses machine learning techniques to predict the priority of the requirements based on the end-user's observational data.

2.4 External Factors

Some recent research on stakeholder based requirements prioritizations are focused on enhancing the reliability of the approaches. For example, Ahmad et al. [15] discusses limitations of existing requirements prioritization techniques with respect to geographical distribution of stakeholders and provides a framework to identify important requirements of a product to be useful for distributed development. As this recent study suggested, we also believe that consideration of end-users' human nature and environmental factors such as geographical distribution will improve the quality of the requirement prioritization. Note that the basic computational unit *situation* used in our proposed approach is representing information about both aspects.

3 Situations, Situation-Transition Structure and Requirements Elicitation Using Situation-Transition Structure

3.1 Situations

Situ defined in [16] is a computational framework that describes the process of inferring human desires through relevant human actions and the environmental contexts. The term “*situation*” defined in the Situ framework encapsulates these three factors: human desire, actions and environmental context are a single computational unit. Our proposed methodology uses this definition of situation in order to embed human factors as well as origin, culture and surrounding environmental factors into the situation-transition structure.

Definition: A situation at time t , is a 3-tuple $\{d, A, E\}_t$ in which d is the predicted end-user's desire, A is a set of end-user's actions to achieve a goal which d corresponds to, and E is a set of environmental context values with respect to a subset of the context variables at time t .

According to the above definition of situation, it is possible to pair time stamped records of desire, set of actions and environmental context values of a particular person into situation tuples which leads to a *sequence of situations* over time. Moreover, it is possible that a person may have multiple desires at a given time t and actions to be performed during that time to reflect one or more of those multiple desires. In other words, a sequence of situations of an individual may include time periods where multiple situations occur simultaneously (a set of concurrent situations).

3.2 Situation-Transition Structure

One significant observed property in the sequence of situations of an individual is that some situations are more likely to transition to a specific subset of future situations than others. We call this property *situation transition*. Hence the term *situation-transition structure* in this paper refers to a domain specific directed weighted graph generated using all possible situation transitions of a person in a particular domain during a pre-defined time period. The overall process of deriving situation-transition structure from observational data is given in [3].

Figure 1 shows an example of such situation transition structure. Here, each node represents a possible situation or a set of concurrent situations in the sequence. An edge from node X to Y represents that in the observational data situation, or a set of concurrent situations represented by node X , is more likely to transit to the situation or a set of concurrent situations represented by node Y . Each edge is annotated with a mutual information value. The mutual information of the directed edge from node X to node Y , $I(X,Y)$ is calculated using formula (1). Suppose X,Y represent situations (or a set of concurrent situations) $SituX$, $SituY$ respectively.

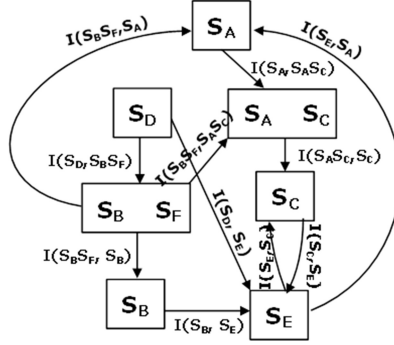


Fig. 1. An example situation-transition structure

$$I(X, Y) = \sum_{x \in \text{values}(X)} \sum_{y \in \text{values}(Y)} P(x, y) \log_2 \left(\frac{P(x, y)}{P(x)P(y)} \right) \quad (1)$$

where,

$\text{values}(X) = \{\text{True}, \text{False}\}$

True: When X is the parent node of the pair. i.e. *SituX* occurred just before *SituY*

False: When X is not the parent node of the pair. i.e. *SituX* does not occur just before *SituY*

$\text{values}(Y) = \{\text{True}, \text{False}\}$

True: When Y is the child node of the pair. i.e. *SituY* occurred just after *SituX*

False: When Y is not the child node of the pair. i.e. *SituY* does not occur just after *SituX*

$P(x, y)$ = Joint probability of $X = x$ and $Y = y$

$P(x)$ = Probability of $X = x$ ($0 < P(x) < 1$)

$P(y)$ = Probability of $Y = y$ ($0 < P(y) < 1$)

Note that the mutual information is directly proportional to the likelihood of this transition. That is, if the mutual information is high, the likelihood of the transition is also high and vice-versa.

3.3 Requirements Elicitation Using Situation-Transition Structure

The procedure of eliciting new requirements from situation-transition structure starts with rearranging the structure into a rooted structure by selecting the appropriate node as the root. Selection of root depends on the domain and set of available situations. We assume that the requirements analyst manually performs this task. Usually, the node representing the initial situation of the sequence of situations or the situation where all the actions and environmental context are in their default values is selected as the root. In cases where all the possible paths in the situation-transition structure cannot be

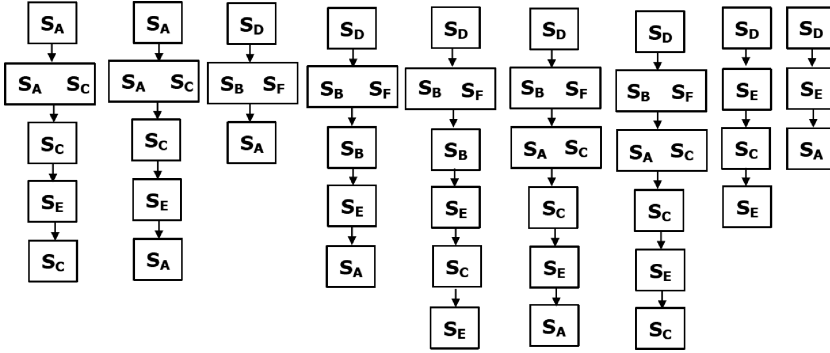


Fig. 2. Set of possible paths in situation-transition structure in Fig. 1.

traversed by starting from a single root node, it is required to select multiple root nodes. Once the root nodes are selected, all the possible paths starting from the root are listed. Figure 2 shows the set of possible paths in the situation-transition structure given in Fig. 1. Here S_A and S_D are selected as the root nodes.

In our earlier study [3] on feature extraction terms and the new requirements elicitation procedure, we proposed five feature extraction terms: direct cause, leads to, terminate, sustain, and prevent in order to analyze the situation-transition structure paths. These five feature extraction terms are then used to define some requirements construction terms so that the elicited new requirements can be presented in semi-formal manner.

4 Requirements Evaluation and Prioritization Using the Situation-Transition Structure

As mentioned earlier, the proposed requirements prioritization approach is based on two main factors: Importance factor and Risk factor. The rest of this section explains each of these two factors.

4.1 Importance Factor of Requirements

According to the definition, the Importance factor in the proposed approach measures the importance of the requirement to the end-user. Here we assume that the importance or relevance of the requirement to the end-user is directly proportional to how often the user requires the feature provided by that requirement. As given in Sect. 3, the elicited requirements are related to one or more situation transitions in the situation-transition structure. Therefore, based on our previous assumption, it is possible to claim that the importance of a particular requirement is directly proportional to the likelihood that the user is engaged in the situation transitions which leads to eliciting that particular requirement. As described in Sect. 3.2, the mutual information can be used as a quantity that represents the likelihood of this transition. Hence, we can use mutual information of the transitions to evaluate the Importance factor of a particular requirement.

1. Group Importance Factor

As defined earlier, each path in the situation-transition structure is a collection of edges where each edge annotated with a mutual information value that represents the likelihood of the transition between the two nodes. We define the path frequency indicator factor as follows:

Definition: The path frequency indicator factor of a path p ($pathFreq(p)$) is the sum of mutual information values of the associated situation transitions.

$$pathFreq(p) = \sum_{\substack{\langle X, Y \rangle \in \text{Set of transitions} \\ \text{associated with path } p}} I(X, Y) \quad (2)$$

where, $I(X, Y)$ = Mutual information value of the transition from X to Y

First, we calculate the path frequency indicator factors of the possible paths identified in the situation-transition structure. Next, we arrange the paths in the situation-transition structure according to the descending order of the path frequency indicator factor. Then, we find the group of requirements for each ordered path such that a requirement belongs to at most one group. For each group, assign an Importance factor within a pre-defined range decided by the requirements analyst such that groups where a path related to higher path frequency indicator factor gets a higher Importance factor and the groups with a lower path frequency indicator factor gets a lower Importance factor. Note that the requirements in a particular group are assigned the same Importance factor.

2. Individual Requirement Importance Factor

The situation-transition structure can also be used to get the Importance factor for each individual requirement as described follows: Consider a requirement R . Let p denote the path in the situation-transition structure that leads to eliciting R and $I(X, Y)$ denote the mutual information value of a transition $\langle X, Y \rangle$ in the path. Then, Importance factor of R , IMP_R can be defined as:

$$IMP_R = \sum_{\substack{\langle X, Y \rangle \in \text{Set of transitions} \\ \text{associated with path } p \text{ that is related to } R}} I(X, Y) \quad (3)$$

4.2 Risk Factor of Requirements

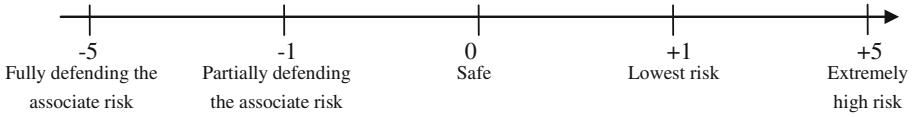
In general the term risk in requirements prioritization refers to the risks involved in business and common issues such as technical or market risks, violation of regulations, unavailability of suppliers, etc. [1]. However, in this paper the term *risk* is used to represent the possible risks related to end-users. We believe that the derived situation-transition structure can be used to gain knowledge about such potential risks.

Note that sometimes end-users' behaviors can result in situations with undesired or harmful outcomes. We call such situations *risk situations* or *hazard situations*. Conversely, some other end-users' behaviors can result in un-harmful situations. We call such situations as *safe situations*. Moreover, some safe situations can also mitigate the possible damage from risk situations. We call those situations *safe defender situations*. Note that set of safe defender situations is contained inside the set of safe situations.

In an ideal case, all the situations in the situation-transition structure must be labeled as either safe or risk situation. However, in practice there can be situations that cannot be clearly labeled as safe or risky. Such situations are called as *un-labeled situations*.

1. Identifying Three Categories of Situations

The first step is to label each node in the situation-transition structure using one of the three categories: risk, safe, safe defender. The label is also supplemented with scalar that represents the relative risk, safe and defender. The scalar is defined within the range +5 to -5 as follows:



We assume that a requirement analyst with some background knowledge can perform this labeling task. If the requirement analyst is unable to label a situation, it will be remain as un-labeled situation.

The next step is to identify the mapping between risk situations and the corresponding safe defender situations. Note that, this is not a “one-to-one” mapping. It is possible that there exists a risk situation that can be mapped to more than one safe defender situation while another risk situation may not be mapped to any such safe defender situation at all. However, we assume that the mapping is “on-to” which implies that each safe defender situation is mapped to at least one risk situation. If a safe defender situation is mapped to more than one risk situation, the requirement analyst can label it with multiple scalar values for each risk. Figure 3 visualizes the three categories and the mapping.

2. Finding Risk Factor

Consider a requirement R . Let S denote the situations related to the transitions that leads to eliciting R . Then, the Risk factor of R , $RISK_R$ can be defined as,

$$RISK_R = \sum_{i \in S} \text{Assigned Scalar of } i \quad (4)$$

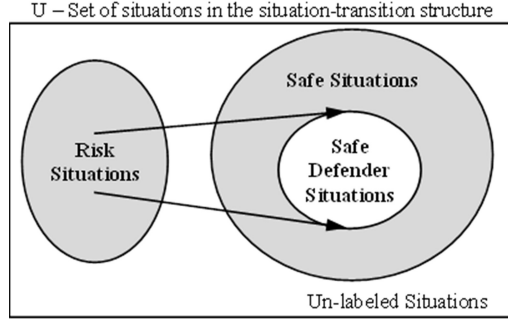


Fig. 3. Three categories of situations: safe, risk and safe defender and the mapping

4.3 Requirements Prioritization

According to the proposed approach the priority of a requirement R is increasing when its Importance factor (IMP_R) is increasing or the Risk factor ($RISK_R$) is decreasing. Hence, the priority of the requirement R ($Priority(R)$) can be estimated as follows:

$$Priority(R) = \begin{cases} k \frac{IMP_R}{RISK_R} & \text{If } RISK_R > 0 \\ kIMP_R & \text{Otherwise} \end{cases} \quad (5)$$

where, k is a positive constant.

A requirements analyst can select positive constant k such that the priorities values lies within a pre-defined range. Note that it is possible to perform both requirements elicitation and prioritization in parallel, so that we can directly elicit a prioritized set of requirements. This will greatly reduce the time and workload of the process.

5 Cooperative Research Environment (CoRE) – A Case Study

In this section we present early stage results of a case study we have conducted to show the applicability of the proposed requirements prioritization approach. CoRE is a website for sharing published and unpublished internal research papers, comments and ideas in a research environment. The existing version of the CoRE website was developed based on a set of requirements elicited by the requirements analysts through traditional brainstorming techniques with no involvement of end-users within the process. Our goal is to find a list of prioritized requirements for the next version of this website through the end-users' observational data.

We considered the current version of the website as the domain and the participants of the website as the end-users. We have collected the records on 65 end-user actions on the current CoRE website interface, such as button and link clicks, menu option selections, current and next webpages (10062 records within 34 days) [17]. In this dataset the actions of each end-user can be uniquely identified through their login

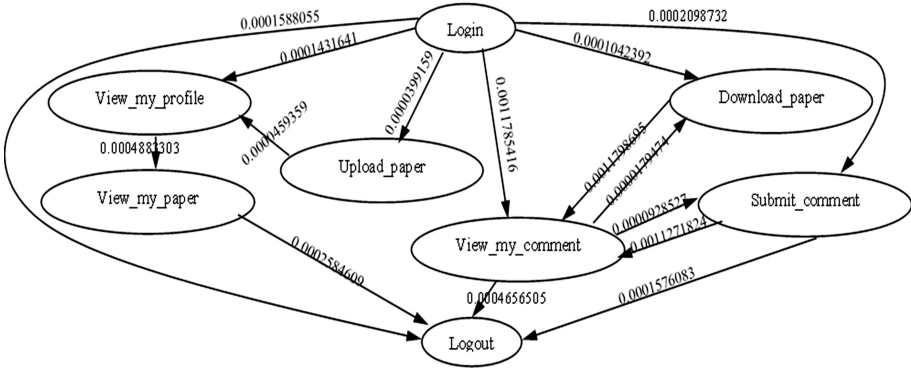


Fig. 4. Part of the situation-transition structure of CoRE dataset

information. Moreover, we have identified 18 end-user desires such as login, view user profile, upload a paper, download a paper, and submit a comment. First, we have derived the situation-transition structure from the observational data of all 65 users and used it to elicit set of new requirements using the procedure given in [3]. Figure 4 shows a part of this situation-transition structure. Each edge is annotated with the mutual information value of the transition. Table 1 listed five sample requirements derived using this part of the structure. These five requirements were unknown when the exiting version of CoRE website was first developed.

Table 1. Sample set of requirements elicited from situation-transition structure generated by considering the behavior of all end-users of CoRE

Requirement	Related situation-transitions
(A) Display the user profile on home page	Login -> View_my_profile
(B) Display the user profile and the list of papers uploaded by the user, once the user uploads a paper	Upload_paper -> View_my_profile -> View_my_paper
(C) Display a link to view submitted comments in download page	Download_paper -> View_my_comment
(D) Display a link to logout, once the user submits a comment	Submit_comment -> Logout
(E) Display the submitted comments once the user submits a comment and then display the link to logout	Submit_comment -> View_my_comment -> Logout

5.1 Evaluating the Individual Requirement Importance Factor

The Eq. (3) is used to evaluate the Importance factor of each individual requirement. Table 2 shows the individual requirement Importance factor of each requirement listed in Table 1.

Table 2. Evaluated individual importance factors of five requirements in Table 1

Requirement	Individual importance factor
(A)	0.0001431641
(B)	0.0005342662
(C)	0.0011798695
(D)	0.0001576083
(E)	0.0015928329

5.2 Evaluating the Risk Factor

In CoRE it is nearly impossible that any end-user behavior could lead to a situation with a physically or mentally harmful or life-threatening outcome. However, end-user behavior in CoRE may cause situations with undesirable outcomes such as uploading or downloading incorrect files, submitting incorrect comments, removing files and comments, etc. Note that the relative risk of each of these outcomes is different. For example, the risk of uploading an incorrect file is higher than submitting incorrect comments in CoRE.

As the first step of evaluating the Risk factors we have labeled each situation in the CoRE situation-transition structure as either risk, safe, safe defender or un-labeled and indicate the relative risk of each situation using a scalar within the range -5 to $+5$. Table 3 shows the assigned labels and the scalars of the situations related to the requirements given in Table 1. Table 4 shows the evaluated the Risk factors of requirement using Eq. (4).

Table 3. Labeled situations with scalar

Situation	Label	Scalar
Login	Safe	0
Logout	Un labeled	-
View_my_profile	Safe	0
Upload_paper	Risk	4
Download_paper	Risk	1
View_my_comment	Safe defend	-2
View_my_paper	Safe defend	-2
Submit_comment	Risk	2

Table 4. Evaluated risk factors of five requirements in Table 1

Requirement	Risk factors
(A)	0
(B)	2
(C)	1
(D)	2
(E)	0

5.3 Prioritizing the Requirements

The priorities of the requirements are evaluated based on their Importance factors and the Risk factors as given in Eq. (5). Table 5 shows the evaluated priorities of the five requirements given in Table 5. Based on the evaluated priorities the five requirements can be ordered as follows: (E) > (C) > (B) > (A) > (D).

Table 5. Priority of five requirements when $k = 10^4$

Requirement R	$\sim$ Priority(R)
(A)	1.43
(B)	2.67
(C)	11.80
(D)	0.79
(E)	15.93

5.4 Prioritizing the Requirements Based on Individual End-User

As mentioned earlier, the preference of requirements is different for each individual end-user. In order to illustrate this difference we have selected three end-users and derived three separate situation-transition structures by only considering observational data related to each of them. Next, the Importance factors and the Risk factors of the five requirements in Table 1 are evaluated based on these three situation-transition structures. Table 6 shows the priorities of the five requirements with respect to the three end-users. The zero priority implies the absence of situation-transitions related to the particular requirement in the user's situation-transition structures.

Table 6. Priority of five requirements with respect to individual end-user when $k = 10^4$

Requirement R	$\sim$ Priority(R)		
	User 1	User 2	User 3
(A)	15.28	0.00	14.78
(B)	0.00	0.00	65.52
(C)	0.00	20.44	0.00
(D)	18.94	0.00	0.00
(E)	76.30	1.30	128.37

Based on evaluated priorities, the five requirements can be ordered with respect to each end-user as follows:

User 1: (E)>(D)>(A)>(B),(C)

User 2: (C)>(E)>(B),(D),(E)

User 3: (E)>(B)>(A)>(C),(D)

6 Discussion

The proposed requirements prioritization approach provides knowledge on what each requirement means to a particular individual end-user or particular end-user group. We believe this will be useful resource to requirements analysts to make better decisions and to gain higher customer satisfaction on the software.

Moreover, the method can also be used to find the alternative requirements in order to select the most important and low risk requirements to perform a particular task. For example, when comparing the two requirements: (D) allowing end-user to directly logout once the comments are submitted and (E) displaying the submitted comments to the end-user before allowing to logout, the latter has higher priority since the risk is low, although the final results of both requirements are same. In addition, the proposed requirements prioritization method can be performed simultaneously with the requirements elicitation process, and the computationally rich nature of the method makes it possible to automate a significant part of the process. This approach will reduce the time and workload of requirements analysts in finding the final prioritized set of requirements.

Despite these significant benefits, the proposed methodology is not without limitations and potential caveats. Although the total number of situations in a particular domain is finite, the manual process of labeling situations in evaluating the Risk factor of requirements is ineffective or sometime impossible in large complex domains. Clustering of situations into risk groups similar to the requirements clustering approach proposed in [10] would be a promising solution for this problem. However, the feature identification for such machine learning based clustering of situations is challenging. We plan to extensively study the applicability of common features such as term similarity [10] in order to perform such automated clustering of situations based on their potential risk.

The proposed method is highly dependent on observational data and it is impossible to guarantee that the data includes all possible situations and situation-transitions that could exist in a particular domain. The unobserved situations and situation-transitions may affect finding the priorities of the derived requirement. Specially, unobserved risk situations can increase the uncertainty of the evaluated priorities. For example, the zero priority values in Table 6 indicate the unobserved situations and situation-transitions that resulted in un-prioritized requirements for each user. One possible solution would be allowing the requirements analyst to identify the unobserved situations or situation-transitions through the background information and modify the existing situation-transition structure by including nodes for new situations and edges for new situation-transitions. The requirements analysts must also estimate the mutual information of these transitions based on their likelihood. However, this manual approach will significantly reduce the effectiveness of the proposed approach. A systematic procedure of handling the unobserved situations and the situation-transitions will be included in future work.

7 Conclusion

In this paper we present a methodology of prioritizing requirements based on end-users' human factors which is novel to the prevalent methods based on business and system perspectives. Our major contribution in this paper is the introduction of a computationally rich procedure for prioritizing requirements by considering end-users' individual interests and behavioral patterns. As mentioned earlier, the main goal of this proposed approach is to support the existing requirements prioritization approaches in order to provide a better understanding of the end-users' personal interest without direct interaction with users. Moreover, this paper establishes our road map towards introducing a new situation-oriented domain-specific risk analyzing procedures in support of requirements prioritizations, which will be an intriguing key for safety critical systems such as healthcare monitoring, mission critical, security, remote sensing systems, etc. Our future work will also investigate an effective and efficient approach to dealing with unobserved situations and situation-transitions.

References

1. Berander, P., Andrews, A.: Requirements prioritization. In: Aurum, A., Wohlin, C. (eds.) *Engineering and Managing Software Requirements, Part 1*, pp. 69–94. Springer, Heidelberg (2005)
2. Chang, C.K.: Situation analytics: a foundation for a new software engineering paradigm. *Computer* **49**(1), 24–33 (2016)
3. Atukorala, N.L., Chang, C.K., Oyama, K.: Situation-oriented requirements elicitation. In: *Proceedings of the 2016 IEEE Computer Society International Conference on Computers, Software & Applications (COMPSAC)*, pp. 233–238 (2016)
4. Brackett, J.W.: *Software Engineering Institute Curriculum Module (SEI-CM) 19–1.2*. Carnegie Mellon University, Pittsburgh (1990)
5. Leffingwell, D., Widrig, D.: *Managing Software Requirements – A Unified Approach*. Addison Wesley, Addison Wesley (1999)
6. Mead, N.: *Requirements prioritization introduction*. Software Engineering Institute Web Publication, Carnegie Mellon University, Pittsburgh (2006)
7. Berander, P.: Using Students as subjects in requirements prioritization. In: *Proceedings of the International Symposium on Empirical Software Engineering (ISESE)*, pp. 167–176 (2004)
8. Bebensee, T., Weerd, I., Brinkkemper, S.: Binary priority list for prioritizing software requirements. In: Wieringa, R., Persson, A. (eds.) *REFSQ 2010. LNCS*, vol. 6182, pp. 67–78. Springer, Heidelberg (2010). doi:[10.1007/978-3-642-14192-8_8](https://doi.org/10.1007/978-3-642-14192-8_8)
9. Beg, R., Abbas, Q., Verma, R.P.: An approach for requirement prioritization using B-Tree. In: *Proceedings of the First International Conference on Emerging Trends in Engineering and Technology*, pp. 1216–1221 (2008)
10. Duan, C., Laurent, P., Cleland-Huang, J., Kwiatkowski, C.: Towards automated requirements prioritization and triage. *Requirements Eng.* **14**(2), 73–89 (2009)
11. Tonella, P., Susi, A., Palma, F.: Interactive requirements prioritization using a genetic algorithm. *Inf. Softw. Technol.* **55**(1), 173–187 (2013)

12. Perini, A., Susi, A., Avesani, P.: A machine learning approach to software requirements prioritization. *IEEE Trans. Softw. Eng.* **39**(4), 445–461 (2013)
13. Babar, M.I., Ghazali, M., Jawawi, D., et al.: PHandler: an expert system for a scalable software requirements prioritization process. *Knowl. Based Syst.* **84**, 179–202 (2015)
14. Glymour, C., Cooper, G.F.: *Computation, Causation, and Discovery*. MIT Press, Cambridge (1999)
15. Ahmad, A., Shahzad, A., Padmanabhuni, V.K., et al.: Requirements prioritization with respect to geographically distributed stakeholders. In: *Proceedings of the IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, pp. 290–294 (2011)
16. Chang, C.K., Hsin-yi, J., Hua, M., Oyama, K.: Situ: a situation-theoretic approach to context-aware service evolution. *Proc. IEEE Trans. Serv. Comput.* **2**, 261–275 (2009)
17. CoRE: Situation Centric Intention Driven Research Experiment for Requirements Engineering and Intrusion Detection. IRB ID 14-347. Iowa State University

Requirements Engineering Toward Sustainable World
Third Asia-Pacific Symposium, APRES 2016, Nagoya,
Japan, November 10-12, 2016, Proceedings
Lee, S.-W.; Nakatani, T. (Eds.)
2016, XXIV, 141 p. 66 illus., Softcover
ISBN: 978-981-10-3255-4