Abstract—There are usually more requirements than feasible in a given schedule. Thus, it’s imperative to be able to choose the most valuable ones for implementation to ensure the delivery of high value software system. There are myriad requirements prioritization frameworks and selecting the most appropriate one is a decision problem in its own right. In this paper we present our approach in selecting the most appropriate value based requirements prioritization framework as per the requirements of our stakeholders. Based on our analysis a single framework was selected, validated by requirements engineers and project managers and deployed for company-wide use by a major IT player in India.

Keywords—requirements prioritization; decision making; framework;

I. INTRODUCTION

The study started by finding out the current status of value based software engineering (VBSE) [1] practices at a major IT company in India, particularly with a focus on value based requirements prioritization (VBRP). About 50 stakeholder interviews were conducted – ranging from business analysts (BAs) to project, program and portfolio managers (PMs). The interviews focused on seeking answers to the following questions:

- Why and if VBSE/VBRP is indeed required?
- The current techniques that are being practiced to factor the notion of value into the various phases of the software development lifecycle (SDLC)
- Eliciting the requirements of an ideal VBRP framework if one is to exist
- Identifying the stakeholders and practitioners for championing and piloting a would-be VBRP tool/method

Based on the responses to the interviews it was concluded that a VBRP framework was necessary and would be very valuable if one were to be designed and implemented, for use by the stakeholders.

Various ad hoc techniques were being practiced and there seemed to be a universal need for more rigor. Another constraint was that of an easy-to-use VBRP tool/framework – especially with established credibility.

Owing to a large number of requirements (about 300-500 on average) simple prioritization techniques like 1-10 ranking or MoSCoW (must, should, could and won’t haves) [10] weren’t able to capture the true value of the requirements. There were too many ties and the process was repeated to re-rank the tied requirements (e.g. must must-haves in case of MoSCoW). Although prioritization was achieved and helped with the overall planning, it was difficult to validate that the prioritization was indeed value-centric. It was assumed that the stakeholders were capable and correct at ranking each requirement as per its intrinsic value.

The simple techniques did not lend themselves well to incorporating change requests (CR) i.e. when a new CR was received, it was difficult to ascertain its value and the need for incorporating it within the current release. Requirements engineers (REs) would engage in extended negotiations with the client to understand the true priority of the requirement. There was no model to insert a requirement and see how it compares to the existing ones, to better understand its true priority and channelize the negotiation effort.

It was decided to survey the literature and compare existing requirements prioritization frameworks that would help alleviate the aforementioned issues and make the prioritization process more akin to decision making [33]. The House of Quality (HoQ) framework [11] was selected for doing the comparative analysis 1. Seventeen different

1 We tailored the Quality Function Deployment (QFD) [12] framework for comparative analysis since it seemed to be a good fit and had the necessary rigor and credibility to help support our findings.

Nupul Kukreja  
Center for Systems and Software Engineering (CSSE)  
University of Southern California  
Los Angeles, USA  
nkukreja@usc.edu

Sheetal Swaroop Payyavula  
Infosys Labs  
Infosys Technologies  
Bangalore, India  
sheetal_payyavula@infosys.com

Barry Boehm  
Center for Systems and Software Engineering (CSSE)  
University of Southern California  
Los Angeles, USA  
boehm@usc.edu

Srinivas Padmanabhuni  
Infosys Labs  
Infosys Technologies  
Bangalore, India  
srinivas_p@infosys.com

Keywords
- requirements prioritization
- decision making
- framework

I. INTRODUCTION

The study started by finding out the current status of value based software engineering (VBSE) [1] practices at a major IT company in India, particularly with a focus on value based requirements prioritization (VBRP). About 50 stakeholder interviews were conducted – ranging from business analysts (BAs) to project, program and portfolio managers (PMs). The interviews focused on seeking answers to the following questions:

- Why and if VBSE/VBRP is indeed required?
- The current techniques that are being practiced to factor the notion of value into the various phases of the software development lifecycle (SDLC)
- Eliciting the requirements of an ideal VBRP framework if one is to exist
- Identifying the stakeholders and practitioners for championing and piloting a would-be VBRP tool/method

Based on the responses to the interviews it was concluded that a VBRP framework was necessary and would be very valuable if one were to be designed and implemented, for use by the stakeholders.

Various ad hoc techniques were being practiced and there seemed to be a universal need for more rigor. Another constraint was that of an easy-to-use VBRP tool/framework – especially with established credibility.

Owing to a large number of requirements (about 300-500 on average) simple prioritization techniques like 1-10 ranking or MoSCoW (must, should, could and won’t haves) [10] weren’t able to capture the true value of the requirements. There were too many ties and the process was repeated to re-rank the tied requirements (e.g. must must-haves in case of MoSCoW). Although prioritization was achieved and helped with the overall planning, it was difficult to validate that the prioritization was indeed value-centric. It was assumed that the stakeholders were capable and correct at ranking each requirement as per its intrinsic value.

The simple techniques did not lend themselves well to incorporating change requests (CR) i.e. when a new CR was received, it was difficult to ascertain its value and the need for incorporating it within the current release. Requirements engineers (REs) would engage in extended negotiations with the client to understand the true priority of the requirement. There was no model to insert a requirement and see how it compares to the existing ones, to better understand its true priority and channelize the negotiation effort.

It was decided to survey the literature and compare existing requirements prioritization frameworks that would help alleviate the aforementioned issues and make the prioritization process more akin to decision making [33]. The House of Quality (HoQ) framework [11] was selected for doing the comparative analysis 1. Seventeen different

1 We tailored the Quality Function Deployment (QFD) [12] framework for comparative analysis since it seemed to be a good fit and had the necessary rigor and credibility to help support our findings.
frameworks were selected for comparison across 17 distinct criteria. Each VBRP framework analyzed, was evaluated against each of the 17 criteria. As mentioned, these criteria were obtained by interviewing the stakeholders to understand their needs for an ideal VBRP framework. The needs were consolidated into 17 distinct groups and their priorities ascertained using a Kano Survey [13]. The corresponding weights (priorities) were then added to the HoQ matrix to see which framework seemed to be the most akin for use. As per the above model and the elicited needs it seemed that TOPSIS (Technique for Ordered Preference by Similarity to Ideal Solution) [4]) would be a good fit.

The comparative analysis was repeated using TOPSIS and the weights from the Kano Survey, in order to evaluate itself (and provide a comparative analysis with the HoQ model). A tool based on TOPSIS was built to run a test pilot. The framework was modified for handling prerequisites and hierarchical prioritization, based on the feedback from our pilot (elaborated later). The framework has now been developed into a web-based system for company wide use.

The forthcoming sections elaborate the analysis approach of selecting a particular framework along with the evaluation of the results.

II. FRAMEWORKS FOR COMPARISON

After a literature survey the most commonly used (or cited) frameworks for requirements prioritization [14, 27, 38] were selected for comparative analysis – they are listed below. It was also decided to evaluate MoSCoW and the simple 1-10 ranking techniques that were currently in use, to better understand how they compared with the others. The frameworks evaluated for this study are:

1. Kano Analysis [31]
2. 1-10 Ranking of each requirement
3. Binary search tree based prioritization [27]
4. 100-point assignment (a.k.a. $100 test) [14]
5. Simple Additive Weighting (SAW) [23]
6. Quality Function Deployment (QFD) [12]
7. Analytical Hierarchy Process (AHP) [8]
8. Wieger’s prioritization [15]
9. Incremental Funded Methodology (IFM) [20]
10. MoSCoW [10]
11. Cost of Delay [21]
12. Purpose Alignment Model [22]
13. Theory-W based prioritization [17]
14. Planning Game (i.e. planning poker) [19]
16. TOPSIS [5]
17. Cost-Value approach [9]

Due to space constraints the detailed working of the frameworks won’t be elaborated. The interested reader is directed to the references. However, the criteria for comparison are elaborated in the next section.

III. REQUIREMENTS FOR THE VBRP FRAMEWORK

As mentioned earlier, these requirements were gained after various interviews with many different stakeholders. Thirty-one requirements were obtained which were condensed into 17 distinct groups to eliminate overlaps. The 17 requirement groups (i.e. criteria) are elaborated below.

A. Scalability

This requirement was almost always the first one to be mentioned by the stakeholder(s). As mentioned above, most REs deal with about 300-500 requirements on average and it’s important to be able to scale up to at least that many requirements (i.e. the framework should be able to handle a large number of requirements with sufficient ease without too much effort and time overhead).

B. Sensitivity Analysis

Almost all REs wanted the ability to perform sensitivity analysis on the overall prioritization. They wanted to explore various what-if scenarios to help take better decisions – both during requirements elicitation and negotiation workshops.

C. Scientific Credibility

There seemed to be a need for using a framework that had worked in practice and is well referenced in literature. The main premise being that it would be easier to get a buy-in from clients if something that was documented and proved to work was being employed, rather than a proprietary solution. The underlying math in effect had to be scientifically verifiable if an RE wanted to reverse engineer the prioritization results or understand the algorithm better.

D. Low effort and time overhead

Time being of constraint, the REs were unwilling to spend more than 10-15% of their requirements elicitation effort on prioritizing. That is about one day of effort for a 2 week requirements engineering workshop (10 working days).

E. Default high priority for security and regulatory requirements

This highlighted the need for having the framework ‘auto prioritize’ these requirements as a must-have and show up at the ‘top of the list’. A better solution was to just not run these requirements through the prioritization. It was still decided to include it as a criterion, to verify the same.

F. Linkability to End Benefits/Goals

REs wanted the ability to link requirements to end benefits/goals of the project so as to have better visibility and control over the prioritization and for better expectations management of the clients².

\[2\] Due to multiple roles in the communication path from clients to REs end-benefits were easy to lose sight of
G. Attaching weights to different criteria

The framework should allow for attaching weights to the different criteria against which the requirements were prioritized. (Closely related to sensitivity analysis but still considered as a separate criterion.)

H. Tool-izable

The framework should be a ready to use tool rather than a set of principles/best practices to be followed, for widespread adoption. This criterion was concerned with the development effort involved in converting the framework to a ready-to-use tool in practice.

I. Ease of use for distributed teams

The company follows a global delivery model for software development and hence the framework must conveniently lend itself for distributed usage, if possible. (It implied that it shouldn’t be mandatory for everyone to be co-located. It should be easy to perform prioritization in a distributed environment.)

J. Tailorability

The framework should be tailorable and not mandate the form/criteria used for prioritization. This primarily meant that it should be easy to add/remove existing dimensions/criteria for prioritization and not be mandated (fixed) by the framework.

K. Downstream traceability

The prioritization output should be such so as to cater for downstream traceability i.e. it should lend itself for re-use and back-reference through all the phases of the project post requirements elicitation (e.g. prioritizing testing and/or maintenance activities based on the requirements prioritization etc.)

L. Low dependence on domain knowledge

It should be relatively easy for an RE with less experience than domain experts to be able to use the framework and produce sufficiently reliable results (i.e. prioritization represents the clients’ view with some accuracy).

M. Bucketed vs. Ordinal ranking

The preference of the output – either bucketed (into large buckets like must-haves, could-haves etc.) or an ordinal scale where each requirement got a ‘numeric score’ stating its importance and rank ordered accordingly. (Owing to confusion in understanding it was decided to split this criterion into two separate criteria – bucketed and ordinal.)

N. Consideration of ‘Ease of Realization’

It was important that the prioritization framework be easily augmented by dimensions particular to the development vendor (e.g. technical feasibility, cost, schedule, effort etc.) Being able to prioritize on business value alone wasn’t valuable for a development organization.

O. Fault tolerance

The ability of having either an inbuilt mechanism or being able to tailor the framework to find inconsistencies and/or judgment errors in prioritization.

P. Ease of Use

There should be a low learning curve for using the framework to facilitate widespread adoptability.

IV. KANO ANALYSIS AND CRITERIA WEIGHTS

In order to elicit the relative importance of each of the above criteria we went back to the stakeholders with a Kano Questionnaire [31] (schedule constraints allowed for surveying only 15 stakeholders). Kano Analysis is used for prioritizing the criteria since prioritizing on desirability seemed to be the most appropriate in this scenario – the more desirable a criterion is the more valuable it is. It was also necessary to test drive the scalability of using a Kano Questionnaire in order to ascertain its ease of use and creation complexity, since it was one of the frameworks for comparison. The questionnaire consisted of two questions for each criterion (Ease of Use was merged with Low effort/time overhead since they were strongly correlated):

- Functional form: How would you feel if <criterion> was present?
- Dysfunctional form: How would you feel if <criterion> was absent?

It was decided to augment the Kano Survey with a question asking for the relative importance/value of each criterion on a scale of 1-10 (1 = least important, 10 = the most important). The overall scores were added up (and normalized) instead of taking averages since the authors felt it would hide the variance in the results. These scores were then used as weights for each criterion in the HoQ matrix.

Fig. 1 shows the data gained from conducting the Kano survey – the ranking of the criteria (as per relative importance) and the Kano category it belongs to.

![Figure 1 Criteria weights and their Kano categorization](image-url)
The Kano Analysis revealed interesting results regarding the various criteria. Although scalability was an important requirement the analysis revealed that the stakeholders were indifferent towards it. When questioned, it was stated that they would be able to distribute the work amongst their colleagues to counter it. They were also indifferent to the output format (bucketed or ordinal) as long as the prioritization was valuable and helped them take better decisions. The same indifference was observed with being able to give security and regulatory compliance requirements a default priority of a must-have – the stakeholders concurred to leave them out of the prioritization if the need be.

The facility of attaching weights to the criteria, toolizability and amenability for use with distributed teams stood out as constraints that the framework must satisfy (i.e. must-haves).

The dark bars (delighters) represented those criteria which the stakeholders didn’t probably have in their current technique(s) of prioritization – satisfying these would be tremendously value adding.

Linkability to benefits (a.k.a upstream traceability) and being lightweight for use fell in the linear category i.e. the more lightweight the framework could be the higher the satisfaction and the more upstream linkability it provided the better the traceability to end-benefits.

Kano analysis hinted early on at the type of framework(s) that would be the most suitable for the stakeholders. In order to have a better understanding to the extent of dissatisfaction (or satisfaction) of the various criteria we analyzed the results using the Customer Satisfaction Index (CSI) as suggested in [16] (Fig. 2).

V. COMPARATIVE ANALYSIS USING HOUSE OF QUALITY

Each framework is evaluated against each criteria using a 1-3-9 scale (1 = low satisfaction, 3 = moderate and 9 = very high satisfaction. Zeroes/blanks imply that the criterion is not applicable or absent for that framework). The House of Quality (HoQ) is used for comparative analysis since it gives the added benefit of stating which frameworks can be combined with each other, along with the overall scores and a graphical comparison with an ideal benchmark. This model also proved tremendously useful for communicating with our project sponsors – the HoQ provided an excellent summary view of the analysis. (The authors were effectively test-driving the Quality Function Deployment (QFD) approach so as to more accurately compare it against the various criteria since it was one of the 17 frameworks considered.)

For each framework its prioritization approach was analyzed and based on judgment given a score of 1, 3 or 9 (or zero) indicating its relative satisfaction against each criterion. This would be different for a different set of evaluators, but the differences were reconciled by having more than one evaluator of each of the techniques. (Three of the authors provided their views on the relative scores before agreeing to a final value.)

VI. EVALUATION OF RESULTS

The weights of the criteria along with the relative contribution of each framework against them are obtained, as explained above. The final score is computed using simple additive weighting of the contribution with the criteria weights. The results are shown in Fig. 3.

The top 3 frameworks as per the evaluation are (in order): TOPSIS [3], Simple Additive Weighting [23] and Wieger’s Prioritization [15]. TOPSIS came out as the most preferred alternative. It was decided to redo the analysis with TOPSIS itself to compare its output with the results in Fig. 3. The same weights and relative contribution scores are used, only the prioritization algorithm is changed. (The analysis was redone with a 1-9 contribution scale instead of the 1-3-9 scale since the QFD template being used fixed the three values to be entered and couldn’t be changed. The results were similar and we don’t report them here.)

TOPSIS is used to evaluate itself along with other frameworks and the output is shown in Fig. 4. TOPSIS results in the similar top 3 frameworks – TOPSIS itself, Simple Additive Weighting and Wieger’s Prioritization. Based on the analysis of the prioritization approaches of the various frameworks, the authors feel that TOPSIS provided a more accurate ranking than the HoQ model.

In order to evaluate our findings the framework was shown to our stakeholders – it was developed in MS Excel® since the current practice at the organization was to gather and manage requirements using Excel worksheets. The framework was well received. However, two critical requirements were uncovered after the initial demos – factoring in prerequisites and the ability to perform hierarchical prioritization. We elaborate on them below.
A. Prerequisite Handling

After a hands-on demo almost all REs mentioned the need for indicating a requirement as a prerequisite of another. The absence of such a feature would be a deal-killer and the framework deemed inappropriate. This was unexpected since it was never brought up during the initial interview sessions with the stakeholders. Having prerequisites complicates the prioritization process and is usually a human intensive task to factor in during planning. When further probed, the REs said that the rank of a requirement should be less than its prerequisites and that’s what they would want to see in the final prioritization output. In their opinion this would represent a true priority order. A post processing macro is applied to the output of TOPSIS – the rank of a particular requirement is made less than each of its prerequisite(s), if any. The following rule is applied for post processing:

Let:

A → B denote that A is a prerequisite of B
If (A → B)
Then Score (B) < Score (A)
Thus, if X → V → W → Z and M → N → Z
Then Score (Z) < Min (Score (X), Score (M)).

The prerequisites of a requirement are entered as a comma separated list of requirement IDs and the scores are recalibrated after post processing. (Cycles are not handled and the REs were made aware of the same.) Although they were aware of post-processing leading to inaccuracies in the results, they were more comfortable with the prioritization output since it echoed a delivery sequence based prioritization. (This feature was added as an option to be enabled if the RE felt the need to do so and could be disabled if accuracy of the results was more important.)

B. Hierarchical Prioritization

Most REs were content with a flat model for prioritization i.e. all requirements were at the same level of granularity. A product management group in the company however, seemed to value hierarchical analysis – prioritizing high level modules with respect to strategic goals and then have the priorities of the low level modules (or requirements) be influenced by the output of their parents. Out of the 17 frameworks evaluated only AHP and the Cost-Value approach (based on AHP) had hierarchical prioritization capabilities built-in. However, these frameworks were readily dismissed due limited scalability (requiring $O(n^2)$ comparisons). The ease of use and scalability of TOPSIS was appreciated but there was a need for hierarchical prioritization. TOPSIS by itself is a flat decision analysis framework, unlike AHP. The capability of

4 Each requirement in the template is given a unique auto-generated identifier
hierarchical analysis was added to the framework using mathematical normalization and is explained below.

Let the parent requirements be at the top-most level of the hierarchy. The final scores of the parents as output by TOPSIS are normalized on a scale of 0 to 1. The next child level of requirements are also normalized from 0 to 1 and then scaled as per their parent i.e. the sum of the scores of the children is equal to that of their parent. (The process is repeated in case of multiple levels.) This can be visualized by the tree structure as shown in Fig. 5 – the sum of scores at any level add up to one and the sum of the children equal that of their parent.

Let:

A ↓ B denote that A is child of B and
C ↔ D denote that C and D are siblings.
If ((X ↓ P) → (Y ↓ Q) and P ↔ Q)
Then Score (Y) < Score (X) may not be true

This implies that if a child in one sub-tree is a prerequisite of a child in another sub-tree, the priority order (after prerequisite post-processing) may not be preserved. Since the priority of a child is scaled (multiplied) by the priority of its parent, the value may change. However, the order is preserved if the prerequisites are among the children of the same parent i.e.:

If ((X ↓ P) → (Y ↓ P))
Then Score (Y) < Score (X) is always true

Along with facilitating hierarchical prioritization normalizing the scores offers the benefit of stating the overall contribution of a particular requirement to the whole i.e. if a requirement receives a score of 0.15 it implies a 15% contribution to the overall set of requirements – a convenient proxy for value (elaborated later).

VII. TOPSIS PRIMER

For the detailed algorithm of TOPSIS the interested reader is directed to the references [3, 4, 5]. A quick overview of the mathematical concept underlying the framework is provided in this section.

TOPSIS is a decision analysis framework where alternatives are scored against a set of criteria. The scoring may be against an absolute scale (e.g. dollars, effort-hours etc.) or a relative scale (e.g. 1-9 Likert scale or 1-3-9 etc.) Each criterion has a particular direction of preference (i.e. the more or less of that criterion is preferred for the prioritization. For example, effort may be less effort the better – depicted with ‘−’ symbol; or the other way round if higher effort implies higher priority, depicted with a ‘+’ symbol).

The prioritization algorithm is based on the Vector Space Model of computation. Each alternative is assumed to be a multi-dimensional vector in vector space. The ideal alternative S* is the one which has the best value for each of the criteria. Consequently, S* is the non-ideal alternative – the one with the worst values for each of the criteria. (Depending on the direction of preference of the criteria (+/-) the best/worst is the maximum or minimum score in the set of alternatives for that criterion.)

The algorithm rank orders the alternatives so that the distance from the ideal solution (S’) is minimized and that from the non-ideal solution (S*) is maximized – hence the name, Technique for Ordered Preference by Similarity to Ideal Solution (TOPSIS).

For selecting the criteria weights it was decided to use Project Success Sliders [24] as a means of eliciting the weights. It’s a simple set of visual sliders which the stakeholders can slide and state their relative importance of a particular criterion. The sliders are ratio-comparable i.e. a particular criterion may be twice as important as the other – the slider would then be twice as long as the other, visually. (Numerically speaking it’s similar to giving the criterion a weight of 2 and the other 4 (or 1 and 2 respectively. Any weight(s) as long as the 1:2 ratio is maintained.) These proved to be very lightweight in practice showing visual tradeoff preferences among the set of criteria. It also proved to be very convenient for performing sensitivity analysis with respect to criteria weights.

VIII. EVALUATION – DEPLOYMENT IN PRACTICE

The TOPSIS template is developed and deployed for practical use both at the organization and for the two-semester software engineering project course, CS577 at University of Southern California. (The student teams develop projects for real clients and prioritization is an intricate activity for project planning, scoping, prototyping and risk analysis.) The evaluations are presented from both perspectives – industrial and academic.

---

5 For each of the 17 criteria the direction of preference was kept as (+) since they were positively framed e.g. ‘low effort’ or ‘ease of use’ implying higher values are better.

6 An interactive implementation can be seen at http://www.mountaingoutsoftware.com/tools/project-success
A. Industrial Evaluation

After the addition of handling prerequisites and the capability of performing hierarchical analysis, the TOPSIS framework was well received by the stakeholders. For evaluating its use and applicability, it was deployed for use by an independent testing team (for value based test case prioritization [25]), business analysts of various projects and an independent product development team in the company.

The testing team used the hierarchical prioritization capability as outlined above. The original requirements were prioritized by the business analysts one level up (i.e. most analysts used the flat model of prioritization since the requirements were captured at the same level of granularity). The testing team created a child level and prioritized the test-cases with respect to the corresponding requirements. Goals/criteria specific to the testing team (e.g. effort-hours, risk of failure etc.) were added to their ‘child level.’ The priority of the each test case was thus influenced by that of the corresponding requirement. This helped them channelize their testing efforts better by focusing on the most valuable test cases first, decreasing their rework effort substantially. The tool thus facilitated downstream traceability and was well received by the testing team. It has since then been piloted with more testing teams and their feedback is being collected to enhance the tool further.

The hierarchical analysis capability was a feature strongly requested by the independent product development team – since they would plan the product roadmap and releases using the same mental model. It was decided to test drive the applicability of TOPSIS based on their current roadmap and release plan as it would provide a good basis for comparison.

A first level prioritization of modules to strategic goals was performed. A child level of low level modules was created along with another level of requirements. The scoring at each level against the relevant criteria was performed and the final prioritization output was compared to the requirements selected for the current release. The team was very satisfied as the output closely reflected their choice of selection in a relatively short span of time. (Typical release planning meetings took days to scope down with extended negotiations). The ability of performing sensitivity analysis helped them explore several what-if scenarios with relative ease. If something did seem amiss, the weights were reconsidered and the model recalibrated. However, there were some requirements (about 10-20%) that seemed out of order. This was expected since the framework provides a mathematical prioritization output and performs prerequisite post-processing (and orders could be reversed, as mentioned above). This however, wasn’t seen as a deterrent and the framework was adopted to provide decision support in the release planning meetings.

As of this writing the framework has been developed into a web based tool for company wide deployment with the hierarchical capability currently being worked on.

B. Academic Evaluation

In the CS577 project course at USC students prioritize each requirement as per the Theory-W prioritization model [18]. The requirements are rated on a scale of 1 – 10 on Business Value (1 = lowest, 10 = highest) and Ease of Realization (1 = extremely easy, 10 = extremely difficult). The former is done by clients and other project stakeholders and the latter usually done by the student developers. The results are then plotted on 2D graph for understanding the relative importance of each requirement and given a MoSCoW categorization.

It was decided to keep the same model but to use TOPSIS instead, for prioritizing. Another dimension of relative penalty was added to the model (i.e. the relative penalty if the particular requirement is not implemented). These dimensions are given the following direction of preference:

- Business Value (+) : The higher the better.
- Ease of Realization (−) : The lower the better.
- Relative Penalty (+) : The higher the more important that particular requirement is.

The weights were selected as per the following rule (using success sliders) – Business Value and Relative Penalty are equally important and Ease of realization is half as important (may be numerically translated as 2, 2 and 1 or 4, 4 and 2 respectively). This was done to emphasize that business value and relative penalty are more important than the technological/implementation aspects of a project and should take precedence during prioritization. Students were encouraged to perform sensitivity analysis with the clients and add other dimensions as they deemed fit (e.g. risk, effort etc.)

The project teams (students and clients) and the CS577 staff gained a very quick understanding of the relative priorities and facilitated on-the-go release planning when a new feature was identified. Students were able to better negotiate with clients if a particular requirement did not seem to be value-adding in comparison to the already existing ones. (The framework could also be conveniently adopted for COTS selection7, but was not evaluated for this scenario.)

The students continued to practice Theory-W prioritization but the underlying prioritization algorithm was changed and multiple dimensions were added for consideration – making the prioritization more fine grained and more accurate. The decision support provided by TOPSIS empowered the teams to be able to better decide scope and priorities for their project planning activities – helping them channelize their effort in prototyping and developing the most valuable and risky items first.

---

7 For COTS intensive projects, it’s mandatory for the student teams to perform tradeoff analysis of various COTS components to justify their selection.
IX. Evaluation of TOPSIS in Practice

Based on our continual use of TOPSIS certain observations have been made for its use in practice, with respect to effort/time overhead and combining it with planning poker for better reliability of results. The framework has proven to be quite extensible along with providing a convenient numeric proxy for value for each individual requirement. We highlight our observations below.

A. Running Time

Unlike pair-wise comparison frameworks the running time of TOPSIS is $O(mn)$ – ‘$m$‘ being the number of criteria and ‘$n$‘ being the number of requirements. However, in most cases, $m << n$. Thus, TOPSIS may be considered to be $O(n)$ with respect to execution time, in practice.8

For hierarchical analysis it can get cumbersome if one has a large number of child levels. Up to two child levels (excluding parent/root level) seems to work well in practice. After the 3rd level of children the numbers (i.e. prioritization scores) become quite small due to normalization and scaling, leading to precision errors (and ties) – unless the floating point range is increased.

It’s proven to be quite manageable for a large number of requirements – either at the same level of granularity in case of a flat model or for a relatively shallow hierarchy (up to 3 levels).

B. Planning Game and TOPSIS

TOPSIS takes a single numeric score of a requirement against a particular criterion. For the project course at USC we combined playing planning poker [19] for estimating the value to use for the ‘Ease of Realization‘ dimension, on a Fibonacci scale. We were effectively able to combine the two techniques rather well and the final value entered was agreeable to the group as a whole. We feel it’s necessary to have a group consensus when entering a value for a requirement against a particular criterion. The planning game can make the discussion more focused and bring the disagreements and issues to light early on, when deciding on a ‘number to enter’ for a requirement against a particular criterion.

C. TOPSIS vs. Simple Additive Weighting

Simple Additive Weighting (SAW) [23] and TOPSIS have the same ‘look and feel’ (i.e. criteria and their weights, direction of preference, scoring scale etc.) Only the underlying algorithm differs. It’s rather easy to have both algorithms in a single template and have it provide the results for both prioritizations. TOPSIS was selected only since the authors felt the underlying math was more rigorous by comparison. In principle, SAW can be used as a drop-in replacement for TOPSIS, with the same benefits as outlined above. It is also possible to have the outputs of both techniques and perform a rank correlation to see the difference between the two prioritizations. However, this was not done for the study.

SAW has the drawback that a linear combination of weights could dominate an optimal alternative [2]. We assumed that TOPSIS wouldn’t face this problem owing to vector space algebra, but this wasn’t formally verified.

D. Extensibility

Tailorability was one of the criteria for selecting an ideal VBRP framework. The advantage of choosing a decision analysis framework is the ability to add/remove dimensions for prioritization. As mentioned above Theory-W based prioritization was successfully combined with TOPSIS – i.e. we used the dimensions of prioritization as suggested by Theory-W but the underlying math of TOPSIS was used to compute a final priority score. Dimensions such as Relative Penalty (Wiegner’s Prioritization [15]) or Cost of Delay [21] etc. advocated by other approaches can be easily incorporated into the framework. The final score however, would be calculated as per the TOPSIS algorithm. Note, we do not claim perfect substitutability – only that valuable dimensions from other algorithms could conveniently be used with TOPSIS (or SAW). One only needs to be aware of the underlying mathematical model being different.

E. The value proxy

It’s usually difficult, if not impossible to quantify ‘value’ – since value lies in the eyes of the beholder. However, with a decision support framework for prioritizing requirements against a set of benefits/goals/criteria, the final scores (normalized) can be used as a convenient proxy for value. From our deployment in practice that seemed to be worthwhile for our stakeholders. They could now answer questions such as ‘which set of requirements generate X% of the overall value’ or given schedule constraints ‘what are the most valuable items to focus on’ and have more value-centric negotiations.

X. Conclusion

In conclusion we would like to point out that TOPSIS is not an ideal framework and may not be suitable in a given context. It was selected based on the criteria specific to our stakeholders. The paper presents our approach to selecting a particular framework. We effectively used a recursive technique that could evaluate itself – VBRP was used (Kano analysis with QFD) to select the most appropriate VBRP framework (TOPSIS), in the given context.

Given the requirements of an ideal framework to use for VBRP, we selected TOPSIS and presented our evaluation on the same. The only criterion that TOPSIS didn’t seem to satisfy well was having built-in support for fault tolerance and consistency indication like AHP [8]. We recommend using extra dimensions like relative penalty etc. so that a RE can at least inspect for inconsistencies with the scores and take decisions accordingly. However, TOPSIS did satisfy

8 Our Excel template takes at most 3000 requirements ($n$) and up to 50 criteria ($m$). $m$ ranged from 4-6 in the test pilots.
the remaining criteria to the fullest — providing a good balance of the various Kano categories outlined earlier. TOPSIS, SAW and Wierger’s were closely ranked implying that either of these approaches could be used in practice with similar benefits. The hierarchical model is based on the concept of mathematical normalization and in principle could be applied to any prioritization approach.

We do not claim that TOPSIS or SAW are the best approaches for prioritization but were the ones that stood out as per our evaluation, with respect to the 17 criteria and their relative weights. We have presented a recursive evaluation approach using decision support tools that can be used to evaluate the various VBRP frameworks (and itself).

Having a decision support framework for use in practice did indeed help REs align requirements with business goals and benefits better (as advocated in [33]). The testing team(s) too was better able to align their test case prioritization effort with that of the requirements. It also provided a proxy for value to help foster value centric discussions and negotiations with the organization’s clients and within the organization itself. It also proved worthwhile in an academic setting where students used it for planning their activities focusing on the most valuable items first. The complete HoQ matrix used for the comparative analysis is shown in Fig. 6 for reference.

REFERENCES


Figure 6 HoQ matrix used for comparative analysis. Relative scores are entered as per the judgmental evaluation of the authors (1 = low, 3 = moderate, 9 = high satisfaction of criteria. Zeroes/blanks imply non-applicability or non-discernability).