CS409 Software Engineering Requirements

Term Paper

Integration of Goal Analysis and HOOMT for Requirements Prioritization

By Raghavendra Kotikalapudi

Table of Contents

Abstract

One of the major activities in requirements engineering is to use requirements prioritization process to help focus on most important requirements. There are many prioritization techniques available, each with their own advantages and disadvantages. Unfortunately, existing prioritization techniques either don't scale well on large projects or fail to provide a formal approach to determine the costs and benefits associated with the requirements. In this paper, we provide a formal method to estimate costs and benefits of requirements by integrating HOOMT with Goal Analysis. We also introduce a novel method called 'verbosity approach' to further elicit requirements, analyze conflicts, verify prioritization computed by cost benefit analysis and analyze multiple stakeholder perspectives.

1. Introduction

In almost every project, budgetary restrictions and time constraints dictate the need for stakeholders to carefully prioritize and select a subset of requirements for development. Requirements prioritization in the industry has been reported to be very informal and dependent upon experience and tacit knowledge (Lehtola, 2004). This is because the methods applicable to large projects are very complicated and is infeasible to incorporate in practice. Importance of prioritization increases with the size of the project. Therefore, it is critical for any prioritization technique to be scalable to large set of requirements and at the same time, simple enough to adopt.

The requirements engineering (RE) community knows multiple proposals for defining what the term ‘importance’ means. Two key factors are benefit and/or cost associated with each individual requirement (Wiegers K. , 1999; J. & Ryan, 1997). Consequently, requirements prioritization can be supported by means of methods allowing the prediction of cost caused and benefit added by single requirement in a project. By addressing the high-priority requirements before considering the low-priority ones, one can significantly reduce both the costs and duration of a project (Hofmann & Lehner, 2001).

1.1 Related Work

There are several prioritization methods proposed to date (Karlsson, Wohlin, & Regnell, 1988), each of which uses one or more different types of analytic or mathematical approaches to assist with requirements prioritization. One such approach is the Analytic Hierarchy Process (AHP) proposed by Saaty, which uses exhaustive pair-wise evaluation by hierarchy level (Saaty, 1994;

1996). This approach has been commented on as being complicated and time-consuming, thus, it is impractical for large projects with many requirements (Finnie, Wittig, & Petkov, 1995). Understanding this disadvantage of AHP, several researchers proposed to reduce the number of comparisons (Carmone, Kara, & Zanakis, 1997; Karlsson J. , 1996; Karlsson & Ryan, 1997; Harker, 1987). However, by reducing the number of comparisons, judgment errors may remain unidentified and the consistency may be decreased (Karlsson, Berander, Regnell, & och Wohlin, 2004). Even worse, the reduced number of comparisons may still be overwhelming in practice (Lehtola & Kauppinen, 2004).

Instead of using AHP, Frank Moisiadis presented a Requirements Prioritization Tool (RPT) (Moisiadis, 2002). RPT prioritizes requirements based on business goals and stakeholder viewpoints. A graphical fuzzy rating scale is used to elicit stakeholders' ratings, and dependencies among requirements are used for requirement prioritization. Although Moisiadis listed the limitations of commonly used requirements prioritization approaches such as Quality Function Deployment (QFD) (Akao, 1990) and AHP, RPT does not overcome these limitations such as the use of subjective ratings and ordinal scales.

Siv Sivzattian and Bashar Nuseibeh proposed a portfolio-based approach to prioritize and select requirements (Sivzattian & Nuseibeh, 2001). This approach selects requirements based on the trade-off between effort and return. However, treating individual requirements as capital assets and applying the ‘‘US capital market risk-free rates’’ and ‘‘average return rate’’ to the prioritization of requirements deserves more explanation and validation. Again, one disadvantage of this approach is that it is very difficult to apply. In addition, quantification of costs and benefits are not considered in these studies.

1.2 Problems

Each of the existing techniques and models on requirements prioritization has its own advantages. However, common problem with most of the methods is that they do not provide a way to quantify benefits and costs associated with requirements. There is no way to verify if the benefit value provided by the stakeholder is true. Also, cost estimation is done intuitively based on the experience of analyst and developer. This arises the need to develop a model to quantify costs and benefits in a systematic manner. As with AHP method, we'll try to incorporate consistency check for the metrics estimated.

1.3 Proposed Approach

The idea is to distribute the prioritization process throughout the lifecycle of requirements analysis instead of localizing it to a specific stage. To apply this particular prioritization technique, we carry out requirements analysis in two stages. In the first stage, we start by identifying the top level goals of the customer and carry out Goal Oriented Analysis (John, Lawrence, & Eric, 1999) to elicit requirements and identify conflicts. In the second stage, requirements are used to identify context objects (Markose, Liu, & McMillin, 2008) which are then modeled as objects using HOOMT (Markose, Liu, & McMillin, 2008) methodology.

In the Goal Analysis phase, the contributions of existing system to the top level goals are identified. When context objects are decomposed into primitive objects, benefits associated with the new system is identified. Cost analysis is then carried out by mapping the costs of the classes to requirements. Once costs and benefits of requirements are determined, knapsack algorithm can be applied to choose requirements such that the total benefit to cost ratio is maximized, given the resource constraints.

We also introduce an optional phase called ‘verbosity approach’ to further elicit requirements, identify missing and ambiguous requirements; analyze priorities based on multiple stakeholder perspectives. This phase can be used to check the consistency of the benefit indices determined by the analysis. The idea is summarized by the figure shown below:

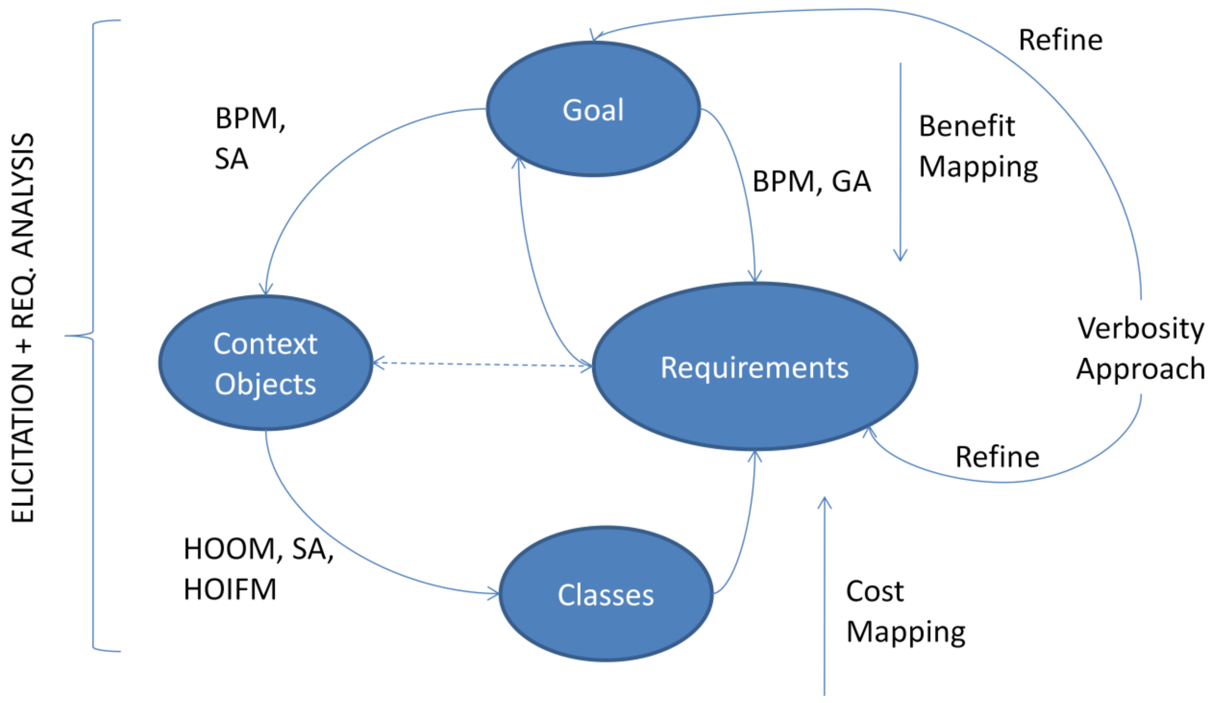

Figure1: Requirements prioritization, various phases

2. Prioritization Model

Instead of asking the stakeholders to assign benefit values, the probable benefit indices of the requirements are determined by finding the contribution of a requirement to the goal. While AGORA (Kaiya, Horai, & Saeki, 2002) method provides a way to consider contribution values, the accuracy of this result is determined by the analysts experience in the related domain. One of the aims of this model is to provide a systematic way to determine benefit indices associated with the requirements based on goal contribution.

During the requirements elicitation phase of a software process improvement project, top level goals are identified by analyzing the purpose for which the system is to be developed. Since each stakeholder may have his/her perception of process improvement, multiple top level goals are constructed.

The goal should be of format: "To <active verb phrase> <target attribute> in/of <target object>", capturing the essential components. For example, in the goal "To reduce paperwork in the organization", the target attribute is 'paperwork', target object is 'organization' and active verb phrase is 'reduce'. This goal format can be mathematically represented as G (F (A) in O), where F (A) represents a function that needs to act upon an attribute belonging to object (O) to achieve the goal (G). It must be noted that 'A' should be a *measurable* or *testable* attribute. If this is not the case, then the new implementation of the system cannot justify customer's goal. For most MIS systems, goal assumes the format G (Min|Max (A) in O) as the goal is usually to minimize or maximize something within an organization. Some of the examples are G (Min (effort) in company), G (Min (operational costs) in organization) etc.

The difference between a goal and requirement is very subtle. Consider the statement “Maintain constant temperature of the system”. This is considered to be a goal if we have a solution space consisting of all possible ways in which this goal can be achieved. It can be viewed as a requirement if there is only one possible way of achieving it. The idea is to say that a Goal is a requirement at higher level of abstraction. A set of requirements within a single solution space contribute to this goal. Moreover, each requirement consists of a set of classes to achieve the functionality or constraint specified by the requirement.

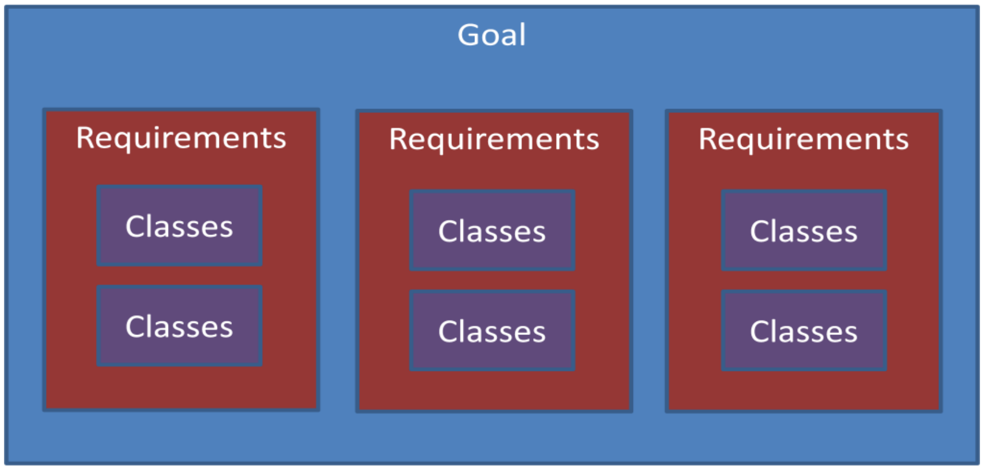

Figure2: Goal, Requirements and classes

2.1 Benefit Assessment

This is carried out by following the five steps as described below. As discussed in the previous section, top level goals are identified from the stakeholders before proceeding to this step.

Step 1: Identify P (G|R) associated with A for each G (F (A) in O)

Once the top level goals are identified in the format G (F (A) in O), these goals are decomposed into requirements by goal oriented analysis. After conflict analysis and negotiations, a set of requirements are selected for implementation. As discussed in previous section, the difference between goal and requirement is subtle. Whenever a goal G is decomposed, we get a sub goal or requirement depending on the context of the system. We call this sub goal as a process P (G|R); where, G|R indicates that this process is either a goal or a requirement.

For each top level goal G (F (A) in O), we identify the processes P (G|R) associated with the attribute A. These processes form the context objects for further elicitation by using HOOMT technique. This is discussed in section 2.2. For now, we will concentrate on benefit analysis.

Step 2: Determine Contrib (P, A) for all P_i's

For all such P's, we determine the contribution of P to attribute A. This is done by analyzing the business processes affecting the attribute A. The contribution of P to A is defined as Contrib (P, A). This should be a quantifiable number as 'A' is a *measurable* or *verifiable* attribute as mentioned earlier. If 'P' affects 'A' differently in different situations, then Contrib (P, A) should be averaged over a time period.

Step 3: Calculate change in contribution ΔC

For every process P_i (G|R) identified in step 1, we model a new process P_{new} to be implemented as per the requirements.

We then compute the change in contribution to attribute A by implementing P_{new} over P_{old} .

$$\Delta C(P_{new}, P_{old}, A) = Contrib(P_{new}, A) - Contrib(P_{old}, A)$$

Step 4: Compute benefit of implementing new process over an old process

For each P_i, compute benefit given by

$$B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, F(A_j)) = \frac{\Delta C(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_j)}{\sum_{for\ all\ such\ Aj} \Delta C(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_j)}$$

As shown by the equation, benefit gives the percentage change in contribution of a process to a goal G (F (A) in O) among all the processes contributing to it.

This equation is only valid if F (A_j) assumes the form of Min|Max (A_j). For other types of F (A), we need to define benefit in terms of ΔA. For example, consider F (A) = Control (Temperature), then we need to define what it means to control in terms of change in temperature. i.e., if ΔT is 0, then the amount of control is 100 units (maximum). In general, we need to plot a graph between F (A) and ΔA to determine the benefit.

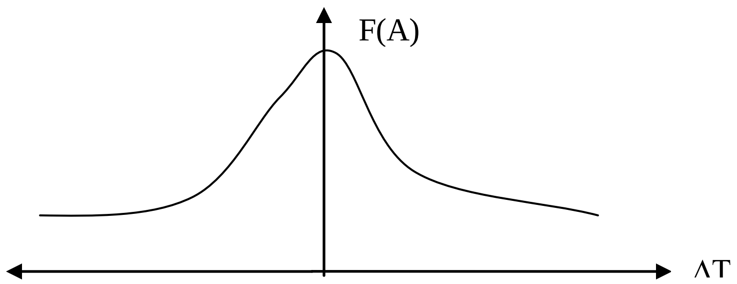

Figure 3: Satisfaction curve for control (temperature)

Definition 1: Weight of the attribute W_A

We define weight of the attribute A as the conversion factor to determine *benefit units.* This is done so that the benefits corresponding to different attributes can be added by taking stakeholders preferences into account.

For example, consider two different attributes cost and effort. Let, $B\left(\mathrm{P_{new}}, \mathrm{P_{old}}, Min(cost)\right) = 30$ and $B\left(\mathrm{P_{new}}, \mathrm{P_{old}}, Min(effort)\right) = 50$. When we try to compute the total benefit associated when P_{new} is implemented over P_{old}, we add both benefits to get 80. But the stakeholder may have different preferences for different attributes. Say, saving cost is twice as important as saving effort, therefore Weight (cost) = 2 * Weight (effort).

In general, W_A is directly proportional to the priority of the top level goal G (F (A) in O). Since, top level goals are limited, their priorities are easy to determine. One can adopt popular methods such as AHP to determine top level priorities.

$$\mathrm{W_A}\ \alpha\ \mathrm{Priority}\{\mathrm{G\ (F\ (A)in\ O)}\}$$

Therefore benefit is defined as:

$$B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, F(A_j)) = \frac{\Delta C(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_j)}{\sum_{for\ all\ such\ Aj} \Delta C(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_j)} \times W_A\ (Benefit\ Units)$$

Step 5: Compute the benefit index, BI (P_{new}, P_{old})

Benefit index defines the total number of benefit units associated by implementing P_{new} over P_{old}. It basically aggregates the benefits associated by implementing P_{new} over P_{old} for all the goal contributions. Different values of A_j correspond to different goals 'P' contributes to.

$$\mathrm{BI}(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}) = \sum_{\mathrm{For\ all\ such\ Aj}} \mathrm{B}(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, \mathrm{A_j})$$

To illustrate the principle, following diagram is shown below:

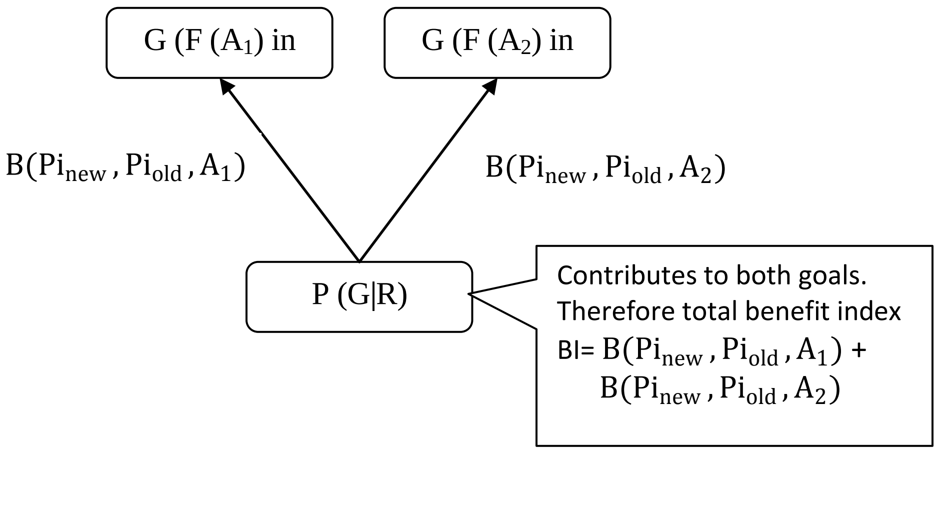
G (F (A_1) in
G (F (A_2) in
$B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_1)$
$B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_2)$
P (G|R)
Contributes to both goals. Therefore total benefit index BI= $B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_1) + B(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, A_2)$

Figure 4: BI computation when P contributes to two goals

2.1.1 Application Example

In this section, we will consider a sample application to illustrate the computations involved in calculating benefit indices. Consider the following scenario. You are asked to develop a school management system. After the initial goal analysis, you identify two major goals

a) To <reduce> <paperwork> in <school>
b) To <reduce> <operational cost> in <finance department>

Comparing with standard goal format as described section 2, we get two goals G (Min (pw) in school) and G (Min (op_cost) in finance dept). Now we will apply steps 1 to 5.

Step 1: Identify P (G|R) associated with A for each G (F (A) in O)

By observing the business process model of the school, we deduce that the attribute 'paperwork' is associated with processes occurring in finance and administration department. Therefore, P = finance and admin, dept are identified to be associated with G (Min (pw) in school)

Step 2: Determine Contrib (P, A) for all P_i's

By identifying the key processes affecting paperwork, let us assume that Contrib (finance, pw) = 300/day (on average) and Contrib (admin, pw) = 100/day.

Step 3: Calculate change in contribution ΔC

Let us assume that the new process (implementation concept) reduces the amount of paperwork in finance and admin department by 100 and 50 respectively.

Therefore,

$(P_{new}\,, P_{old}\,,\ pw) = -100\ for\ p = finance$ & $\Delta C(P_{new}\,, P_{old}\,,\ pw) = -50\ for\ p = admin.$

Step 4: Compute benefit of implementing new process over an old process

Assuming $\mathrm{W}_{\mathrm{op\_cost}} = 2 \times \mathrm{W}_{\mathrm{paperwork}}$, as determined by analyzing top level preferences from stakeholders; we have,

$$B(\mathrm{Pi}_{\mathrm{new}}\,, \mathrm{Pi}_{\mathrm{old}}\,, Min(paperwork) = \frac{-100}{(-100-50)} \times 1 = 0.67;\ for\ p = finance$$

$$B(\mathrm{Pi}_{\mathrm{new}}\,, \mathrm{Pi}_{\mathrm{old}}\,, Min(paperwork) = \frac{-50}{(-100-50)} \times 1 = 0.33; for\ p = admin$$

This means that, for the goal G (Min (paperwork) in school), finance dept contributes to 67% and admin dept contributes to 33%. For p = finance department, let us assume $B(\mathrm{Pi}_{\mathrm{new}}\,, \mathrm{Pi}_{\mathrm{old}}\,, Min(operational\ cost) = 0.47$

Step 5: Compute the benefit index, BI (P_{new}, P_{old})

P = finance dept, contributes to goals G (Min (pw) in school) and G (Min (operational cost) in finance dept). Hence, it is associated with two attributes ‘operational cost’ and ‘paperwork’. The situation is illustrated below:

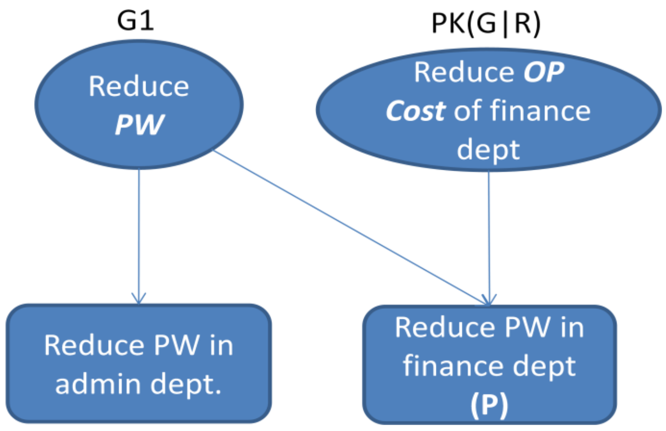
G1
PK(G|R)
Reduce *PW*
Reduce ***OP Cost*** of finance dept
Reduce PW in admin dept.
Reduce PW in finance dept **(P)**

Figure 5: Illustration of P’s contribution to both the goals

Therefore, $\mathrm{BI}(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}) = \sum_{\text{For all such Aj}} \mathrm{B}(\mathrm{Pi}_{\mathrm{new}}, \mathrm{Pi}_{\mathrm{old}}, \mathrm{A}_j) = (0.67 + 0.47) = 1.14$

2.2 Cost Estimation

As discussed in section 2, a goal is decomposed into P (G|R). These P_i's form the context objects, the starting points for HOOMT analysis. For example, in the school management application example discussed in section 2.1.1, G (Min (paperwork) in school) decomposed itself into P (Min (paperwork) in finance dept) and P (Min (paperwork) in admin dept). These form the context objects for HOOM analysis.

For each such P, we now begin with the activity diagram of all the processes affecting attribute 'A'. As high order objects are analyzed structurally by *zooming in* or de-encapsulating the *object structure*, the activity diagram is transformed into swim lane diagrams and the objects within the HO objects are assigned appropriate operations based on the analysis of business process model.

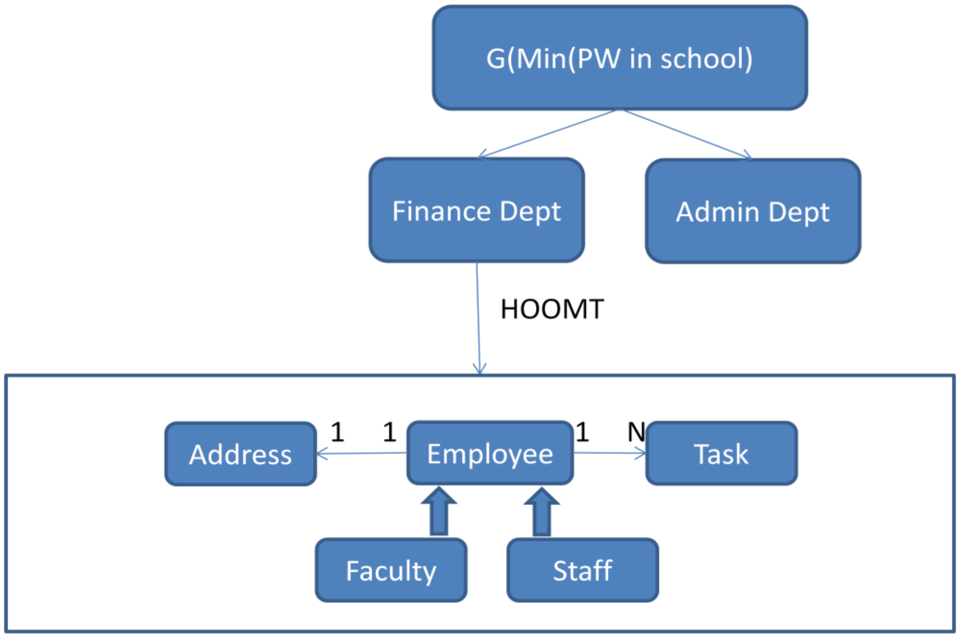

Figure 6: Transformation from Goal to classes via context objects

Once the classes are determined, a cluster of classes are traced back to a requirements generated in phase 1. This is easily done as the context objects are explored on the basis of activity diagrams which are chosen as per the requirements.

There are three types of relationships in UML. They are aggregation, association and generalization. For aggregation and association relationships, the cost of classes participating in the composition is first determined. For inheritance relationships, the cost of the base class is determined first, followed by the cost of subclasses. For example, in Figure 6, the cost of Address and Task is determined, followed Employee, Faculty and Staff classes. In this manner, the cost of an entire cluster of classes is determined. They're then mapped to a requirement whose cost is to be estimated.

Code reusability is one of the major paradigms in OOP. Whenever code from previous projects is reused, the cost estimate of the class is adjusted accordingly. This yields a better estimate as the developers precisely know the functionality and the dependencies involved amongst various objects. It can be shown that this approach provides better accuracy in comparison to the cost estimation based on intuition and experience.

Cost can be estimated in terms of monetary value, time etc.

2.3 Prioritization

Every requirement has a cost and benefit associated with it. The methods described in sections 2.1 and 2.2 consider dependencies, code reusability from previous projects. Given the resource constraints (time, cost etc.), a set of requirements should be selected in such a way that the total benefit index value is maximized. This can be formulated as an operations research problem.

Let, BI_r, C_r be the benefit index and the cost associated with the requirement 'r'. Let B be the resource constraint specified. We define a variable $X_r = \{0, 1\}$ to indicate if requirement 'r' is chosen or not. The objective is to $Maximize\ \sum BI_r \times X_r$, subject to $\sum C_r \times X_r \leq B$.

For each release set, a set of requirements are selected so as to maximize the benefit within the given budget constraints. This is basically a 0-1 knapsack problem, which is known to be N-P complete. Greedy algorithm may or may not yield optimal solution, but is at least close to optimal, which is generally acceptable.

3. Verbosity Approach

Fulfilling your customer's interests determines your market success, but how do you find these requirements effectively and efficiently? As simple as this question sounds, answering it in daily practice is difficult. Often, stakeholders are interviewed about their requirements or asked to write them down, but this approach rarely uncovers the real requirements that reflect a customer's true interests or needs. We need a way of getting information about the customer's core desires-conscious, unconscious, even subconscious. The hottest sellers are products that fulfill these desires.

This section describes a novel multiplayer game based approach to elicit requirements, identify ambiguous and missing requirements, and analyze conflicts from multiple stakeholder perspectives based on the data collected during the game play. This approach is inspired by 'Verbosity' (Luis, Mihir, & Manuel, 2006), a game used to collect common sense facts, hence the name. The idea of this game is to collect facts about requirements and goals, while keeping the game play as enjoyable as possible.

3.1 Game play

Stakeholders are clustered into related groups such as stakeholders in finance department, administration etc. Two stakeholder sets A and B are formed with half of them from each cluster. Goals and requirements are input to the game along with their traceability. All the stakeholders are asked to participate in the game. One player from set A is randomly paired against a player from set B.

The interface screens for player A and player B are as shown in figure 7. Player A's screen consists of a combo box containing all the final selected goals. Whenever a goal is selected, mid screen shows a list of requirements associated with that goal. To the bottom of the screen is the hint template. The goal of player A is to make B guess his/her selection. A can either select a goal or a requirement.

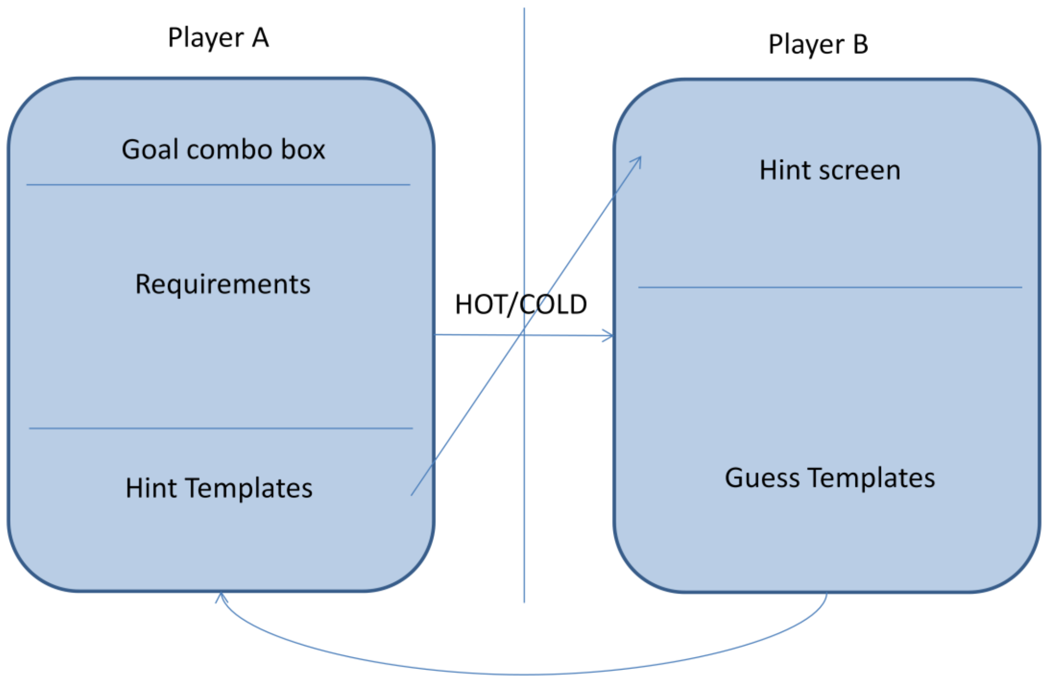

Figure 7: UI for players A and B

‘A’ provides a hint for the selection my using sentence templates. These hints are displayed to player B. Player B tries to guess A’s selection (requirement or goal) using the guess templates on his/her screen. Every guess from B is rated by a fuzzy hot or cold indicator to indicate the closeness to the selection. HOT indicates that the guess is very close. The game play proceeds in this fashion until B skips the round or makes a right guess. The game is played in real time distributed environment and all the events are logged, including the time taken by B to formulate the guess. The idea is to determine the stakeholders sub conscious preferences towards requirements and goals by inferring from the game play.

A sample screen of the game play is shown in figure 8 to illustrate the principles.

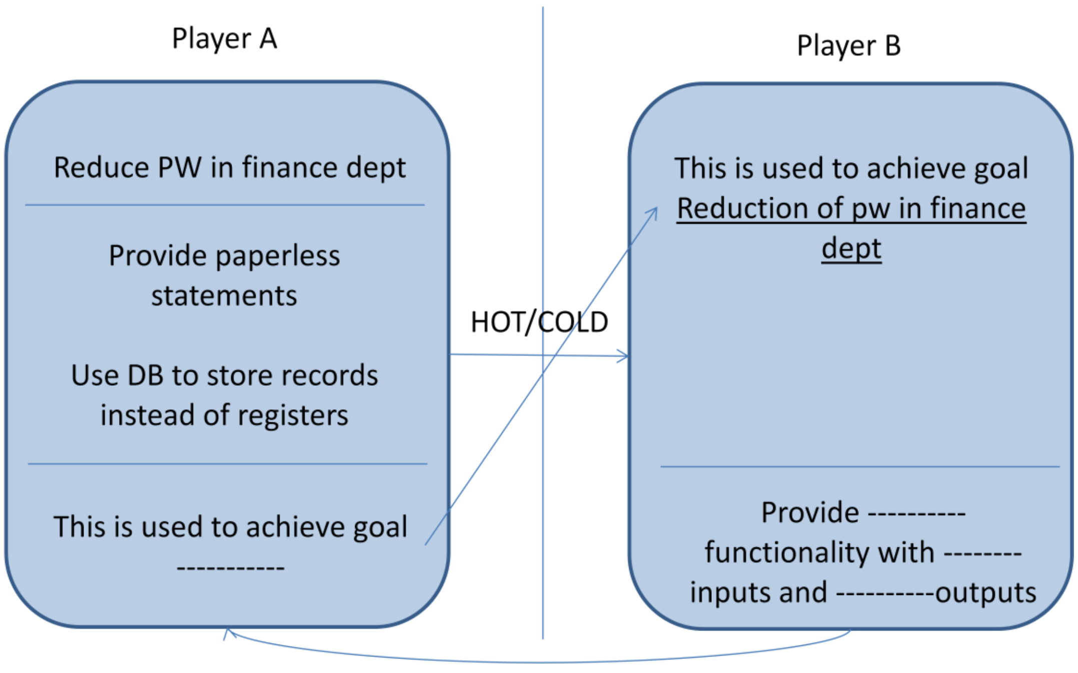
Player A
Reduce PW in finance dept
Provide paperless statements
Use DB to store records instead of registers
This is used to achieve goal -----------
HOT/COLD
Player B
This is used to achieve goal Reduction of pw in finance dept
Provide ---------- functionality with -------- inputs and ----------outputs

Figure 8: Sample game play screen of A and B

In figure 8, the goal ‘Reduce paperwork in finance dept” is selected. The requirement “Provide paperless statements” is chosen from the list of requirements associated with the selected goal. The sentence template for providing the hint is “This goal is used to achieve -----------“ and the template for submitting a guess is “Provide ----------- functionality with ------------ inputs and ----------- outputs”. The reason for using sentence templates, their purpose is described in section 3.2.

3.2 Sentence Templates

The purpose of sentence templates is to facilitate useful inferences by observing the game play. As the guesses provided by A and B are analyzed, sentence templates should be designed in such a way that it allows for a useful deduction. For example, consider the screen of game play in figure 8, by using the sentence template "This requirement satisfies the goal --------------", depending on what A fills in the blank, we can get more information about the goal as perceived by A. It also makes the hint less obvious, i.e., it is not that easy for B to guess it. This provokes some kind of thinking on B's side to guess the requirement. As the requirement guess is not obvious, game play will continue for a while, enabling us to collect useful information for analysis.

In general, sentence templates should be restrictive so that the information collected from the game play is useful and in unambiguous format. Unambiguous format is important, as the inference engine should not misunderstand the hint provided by A. It allows us to create an inference engine easily as the grammar for sentence templates are known. Sentence templates should not be too restrictive; otherwise the player will simply skip the round and no useful data will be collected.

Sentence templates will vary for the selection of goal and requirement. Moreover, they are different for various types of projects. For example, in real time projects, sentence templates should be oriented to collect useful information about NFR preferences of a customer. The choice of sentence templates and their format is something that has to be determined by future research. For now, we only introduce the concept of sentence templates.

Lastly, sentence templates prevent collaboration or cheating within the game. i.e., they prevent A from giving a very obvious hint to B. If this is the case, no useful inference will be made.

Moreover, inclusion of templates makes the game play a challenging experience. It encourages extended hours of game play for collecting useful information. We are not sure about this, but it certainly worked for ‘Verbosity’ (Luis, Mihir, & Manuel, 2006).

3.3 Inferences

The following inferences can be made from the game play between player A and B.

1) By using player A’s choice of selection of goals and requirements, priorities from his/her perspective can be determined. The more he/she selects a type of goal or requirement, more is his sub conscious priority towards that selection.
2) When player B makes a guess, player A rates it as HOT or COLD depending to indicate the closeness of the guess provided by B. HOT, COLD ratings provided by A tells about his/her perspective to the similarity of goals or requirements guessed by B.
3) If B guesses something not available in the goal or requirements list of A, and supposedly, ‘A’ rates this guess as HOT, then this guess is probably a missing requirement or a goal. Once such missing elements are identified, they are included in the game play as a bonus tiles to encourage others to clarify this further.
4) Time taken by B to guess the right requirement tells about the ambiguity of the goal or requirement selected by A.
5) A’s choice of selection NFRs or FRs tells about his/her sub conscious preferences towards functionality or quality. The type of NFRs chosen tells about A’s preference towards a class of NFRs (reliability, response time etc.)

6) The benefits determined by analyzing the game play can be compared to the benefits determined using steps described in section 2.1. If they vary significantly, it either means that true customer goal is not identified or stakeholder's requirement is misunderstood.

Therefore, by using the inferences described above, we get requirements elicitation, priorities with respect to various stakeholders, conflict analysis, missing and ambiguous requirements. The usefulness of the game will mostly depend on the type of sentence templates used. Most of the inferences stated above are common sense facts. They however need to be tested for their correctness by surveying them in various projects. We only introduce the concept for now; surveying the effectiveness of this approach will be our future work.

4 Conclusions

In this paper, we provided a way to estimate benefits and costs in a systematic manner. As our model distributes this process amongst other phases of requirements analysis, we can conclude that the perceived scalability of this approach is better. Costs and benefits estimated by this approach consider the dependencies amongst various requirements, and the code reusability from other projects. Thus, this approach is very practical; moreover it *integrates naturally* with various phases of requirements analysis and hence is easy to adopt. Currently, benefits can only be assessed for goals of format G (Min|Max(A) in O), a more systematic way for estimating benefits for all kinds of F(A) needs to be researched in the future.

Towards the end of this paper, we introduced verbosity approach. This method has not been surveyed but it has the potential to uncover subconscious priorities and requirements from the stakeholders. Lot of work needs to be done in researching useful sentence templates for inference. When used together with the metric estimation model proposed, verbosity approach complements the former by providing a consistency check for benefit indices. This approach is especially useful in situations where the stakeholders are distributed in various parts of the world.

5 References

1. Akao, Y. (. (1990). *Quality Function Deployment: Integrating Customer Requirements into Product Design.* Cambridge, MA: Productivity Press.

2. Carmone, F., Kara, A., & Zanakis, S. (1997). A Monte Carlo investigation of incomplete pairwise comparison matrices in AHP. In *Eur. J. Oper. Res.* (pp. 538–553).

3. Finnie, G., Wittig, G., & Petkov, D. (1995). Prioritizing software development productivity factors using the Analytic Hierarchy Process. J. Syst. Softw.

4. Harker, P. (1987). Incomplete pairwise comparisons in the analytical hierarchy process. In *Math. Modell* (pp. 837–848).

5. Hofmann, H., & Lehner, F. (2001). Requirements engineering as a success factor in software projects. *IEEE Softw. (July/August)* , 58–66.

6. J., K., & Ryan, K. (1997). A Cost-Value Approach for Prioritizing Requirements. *IEEE Software* , 67-74.

7. John, M., Lawrence, C., & Eric, Y. (1999). From Object Oriented to Goal Oriented Requirements Analysis. *Communications of the ACM January Vol. 42, No. 1* , 31-37.

8. Kaiya, H., Horai, H., & Saeki, M. (2002). AGORA: Attributed Goal-Oriented Requirements Analysis Method. *Requirements Engineering, IEEE International Conference* (p. 13). Los Alamitos, CA: IEEE Computer Society.

9. Karlsson, J. (1996). Software requirements prioritizing. *Second International Conference on Requirements Engineering* (pp. 110–116). Los Alamitos, California: IEEE Computer Society Press.

10. Karlsson, J., & Ryan, K. (1997). A cost-value approach for prioritizing requirements. *IEEE Softw. (September)* , 67–74.

11. Karlsson, J., Wohlin, C., & Regnell, B. (1988). *An evaluation of methods for prioritizing software requirements.* J. Inf. Softw. Technol.

12. Karlsson, L., Berander, P., Regnell, B., & och Wohlin, C. (2004). Requirements prioritisation: an experiment on exhaustive pair-wise comparisons versus planning game partitioning. In *Empirical Assessment in Software Engineering (EASE 2004)* (pp. 145–154). Edinburgh, Scotland.

13. Lehtola, L. (2004). *Requirement prioritization challenges in practiceLecture Notes in Computer Science.* Berlin: Springer.

14. Lehtola, L., & Kauppinen, M. (2004). Empirical evaluation of two requirement prioritization methods. In *product development projectsLecture Notes in Computer Science* (pp. 161–170). Berlin: Springer.

15. Luis, v. a., Mihir, K., & Manuel, B. (2006). Verbosity: a game for collecting common-sense facts. *Human Factors in Computing Systems* (pp. 75 - 78). Montréal, Québec, Canada : ACM.

16. Markose, S., Liu, X. F., & McMillin, B. (2008). A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems. *Embedded and Ubiquitous Computing* , 75-81.

17. Moisiadis, F. (2002). The fundamentals of prioritizing requirements. *Systems Engineering, Test and Evaluation Conference.* Sydney, Australia.

18. Saaty, T. (1994). *Fundamentals of Decision Making and Priority Theory with the Analytical Hierarchy Process.* Pittsburgh, PA: RWS Publications.

19. Saaty, T. (1996). *Multicriteria Decision Making: The Analytic Hierarchy Process.* Pittsburgh, PA: RWS Publications.

20. Sivzattian, S., & Nuseibeh, B. (2001). Linking the selection of requirements to market value: a portfolio-based approach. *Proceedings of Seventh International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ 2001).*

21. Wiegers, K. (1999). First things first: prioritizing requirements,. *Software Development* .

22. Wiegers, K. (1999). *Software Requirements.* Redmond, Washington: Microsoft Press.