Research Journal of Applied Sciences 10 (10): 574-586, 2015
ISSN: 1815-932X

StakeRatreet: A New Approach to Requirement Elicitation Based on Stakeholder Recommendation and Collaborative Filtering

[1]C. Senthil Murugan and [2]S. Prakasam
[1]Department of CSA, SCSVMV University, Kanchipuram, India
[2]Department of MCA, Thiruvalluvar College of Engineer and Tech, Vandavasi, India

Abstract: The customers' needs in a software project are identified in the process of software requirements elicitation. For developing a software system, this process is considered as one of the most important parts. In this part, it is decided precisely what will and when will be built. A close interaction between developers and end-users of the system is needed by requirements' gathering. Meetings can be costly, inconvenient and infrequent if developers and end-users are in different organizations or different locations. The quality of the elicited requirements can greatly be impacted if there is a problem of communication. The miscommunication is leads the project failure. Requirement elicitation is a process difficult to scale to large software projects with many stakeholders which involves identifying and prioritizing requirements. A stakeholder is an individual or a group who can influence or be influenced by the success or failure of a project. In the existing methods to identify and prioritize requirements do not scale well to big projects. The large scale projects tend to be set by three problems: information overload, inadequate stakeholder input and biased prioritization of requirements. Existing methods to identify and prioritize requirements do not scale well to large projects. Existing requirements prioritization methods require substantial efforts from the requirements engineers when there are many requirements. To address the problems stakeholder recommender model will contain the following steps: identify the large project, analysis of requirements, identify and prioritize stakeholders, predict requirements, prioritize requirements. For making predictions, our approach will use one of the most well known algorithms that is k-Nearest Neighbor (kNN) algorithm. KNN is used to identify like-minded users with similar rating histories in order to predict ratings for unobserved users-item pairs, i.e., preprocessing. A unique subset of the community for each user is found out by KNN by identifying those with similar interests. To do so, every pair of user profile is compared to measure the degree of similarity. A neighbourhood is created for each user by selecting the k most similar users. The similarity between each pair of user profiles for users in the neighbourhood is used to compute predicted ratings from any range. Finally, the predicted ratings for the items are sorted according to the predicted value and the top-N items are proposed to the user as recommendations where N is the number of items recommended to the user.

Key words: Requirements specifications, elicitation methods, requirements prioritization, recommender systems, social network analysis, stakeholder analysis

INTRODUCTION

Stakeholder identification plays a key role in the stakeholder management analysis. The stakeholder identification process is the very first steps taken in initiating a new project. It is a precursor to stakeholder analysis report. Only if stakeholder identification is done initially can the requirements of the project can be elicited. It is the stakeholder that can give access to system manuals, users, maintenance logs and the myriad of other assets that can be leveraged during the requirement elicitation phase (Alexander and Robertson, 2004). The first step concerned is with the phase "Who are the stakeholders"-stakeholders are all those who need to be considered in achieving project goals and whose participation and support are crucial to its success. Stakeholders can be individuals within the project individuals and groups within the organization and individuals or groups outside the organization. Thus, there are many software stakeholders to be identified for a project. The following are the different ways to identify stakeholders who play a role in the software project. In

Corresponding Author: C. Senthil Murugan, Department of CSA, SCSVMV University, Kanchipuram, India

doing this, the main objective is to try to break the large group of stakeholders into smaller groups since large groupings can impact the value of information gained from the process (Sharp *et al*., 1999).

Category approach: It is the most commonly used method where categories of stakeholders are created by the project team based on past experience and theses are then used to identify stakeholders. The risk of this approach is that it may be too broad resulting in overlooking of some stakeholders.

Role approach: The project team works from a generic list of stakeholder roles. Because the roles are very generic, this approach makes it easy to overlook stakeholders who don't have a direct interactive role in the system or project.

Interview approach: The interview method is most useful and more convenient for identifying new stakeholder roles and new individuals to potentially fill those roles. Unfortunately, it is time consuming and is found to be unfeasible in majority of software projects.

Search approach: Here stakeholder specific roles are identified that are appropriate for the project. While this approach will usually result in a more complete stakeholder list a key issue is knowing when to stop.

Following approach: The approach follows some specific artifact through the software project management life cycle to identify the stakeholders who provide or use that artifact or who have responsibility for it in some way. This approach helps in identifying new stakeholders to the project as well as missing many stakeholders that are not part of the specific artifact that is being followed.

Goals approach: It is another way to identify stakeholders which is used extensively in software industry. The results of the stakeholder identification effort should include a whole list of stakeholder roles as possible given the facts at that time, the above information is classified and put in the stakeholder registry.

Though initially most of the software companies did not believe it necessary to have a registry (Alexander, 2007), changing times have prompted most of them to give due importance's for stakeholder identification and placing up a registry in the initial phases of the project itself.

The first step in the RE process is the elicitation of requirements. The important goals of requirements elicitation is to find out what problems need to be solved. It is defined as "the process of identifying needs and bridging the disparities among the involved communities for the purpose of defining and distilling requirements to meet the constraints of these communities". It is served as a front end to systems development. Requirements elicitation involves social, communicative issues as well as technical issues (Herlocker *et al*., 2004; Nuseibeh and Easterbrook, 2000).

With requirements elicitation, requirements analysts, developers, sponsors, funders and end users are involved. The elicitation process is further decomposed as follows (Albrecht and Gaffney, 1983): identify the sources of requirements. Sources may be an end user, an interfacing system or environmental factors.

Gather the wish list for each relevant party. Originally wish list contains ambiguities, inconsistencies, infeasible requirements and untestable requirements. Also, it is incomplete.

The wish list for each relevant party is documented and refined. All important activities and data are mentioned in wish list. The data is repeatedly analysed until it is reliable. Data in list is at high level. It is stated in user-specific terms.

The wish lists are integrated across various relevant parties. The conflicts between the viewpoints are resolved. One more important part of this process is consistency checking. Feasibility for wish lists or goals is checked.

The non-functional requirements like performance and reliability are determined. And these are stated in requirements document. These activities are common to most of the process definitions for requirements elicitation found in the literature. The resulting product from the elicitation phase is a subset of the goals from the various parties which describe a number of possible solutions (Alexander and Robertson, 2004). Existing requirements elicitation approaches have proven insufficient to record complete, consistent and correct requirements. Studies conducted have shown that 40% of defects in software projects are due to incorrect recorded requirements. Provoking clear, complete and correct requirements is still a challenge and a difficult undertaking in requirements engineering. Vital information related to the requirements is often ignored and partially or not recorded at all during requirements elicitation (Alexander, 2007; Lehtola *et al*., 2004; Wasserman and Faust, 1994). Engineers documenting the requirements may misapprehend, partially document or omit important statements. Most of

the existing requirements elicitation approaches are clearly lacking capabilities to support gathering complete and detailed requirements in a natural flow.

Our software project proposes an open and inclusive method for requirements elicitation using social networks and collaborative filtering (Wasserman and Faust, 1994). An inherent feature in existing requirements elicitation methods is that they depend on a small number of experts such as the requirements engineers or the project team. These experts become a bottleneck in large-scale software projects where they have to process many requirements from many stakeholders. To remove the bottleneck, this work will shift the emphasis from requirements elicitation involving only the experts to a collaborative approach in which all stakeholders have a say.

MOTIVATION

Software systems are growing daily. The increase in size extends beyond mere lines of code or number of modules in the software systems. A software system can now affect million and millions of people. In an ideal world, large software systems would always function as intended-users' needs would be meet and customers would get value for their money. Projects to deliver the software systems would always be on time and under budget, regardless of the size of the software system. Current software development is far from ideal. Large projects are often late and over budget. They sometimes never deliver (Sarwar *et al*., 2001; Sharp *et al*., 1999; Zave, 1997; Charette, 2005; Alexander and Robertson, 2004).

Today, software projects to build large software systems involve vast numbers of stakeholders the individuals or groups that can influence or be influenced by the success or failure of a software project (Nuseibeh and Easterbrook, 2000). These stakeholders include customers who pay for the system, users who interact with the system to get their work done, developers who design, build and maintain the system and legislators who impose rules on the development and operation of the system (Sharp *et al*., 1999; Nuseibeh and Easterbrook, 2000). In big projects, stakeholders can cut across divisions and organisations. They have diverse needs which may conflict.

PROBLEM DEFINITION

During literatures review, some problems were encountered to us when identify and prioritize the stakeholders regarding software project. Literatures do not help in identifying right stakeholders for a specific system and do not take any step about the solution how to prioritize the selected stakeholder. Here we attempt to find out ten major problems those were encountered during the papers reviewing:

- All relevant stakeholders are consulted with less attention
- Software engineering community does not address how stakeholders are identified in remote regions despite acknowledging that locality and culture typically have an impact on a software product
- Information overload is inevitable in big projects
- Inadequate stakeholder input is a natural outcome of current practices
- Biased prioritization of requirements occurs because current prioritization practices depend on individuals, who may not have a global perspective in large projects
- There is a need of systematic approach that could help in identifying and choosing the appropriate stakeholders
- As stakeholder participation can be enormous approach, it should also embrace prioritization
- Existing methods to identify and prioritize stakeholders do not scale well to large project
- Existing stakeholder identification methods require substantial efforts from the requirement engineering when there are many stakeholders
- The existing prioritization methods fail to appropriately structure the data for stakeholder value. This problem is often compounded by a failure to handle multiple stakeholder view points
- One of the main problems in requirement engineering is lack of necessary skills of stakeholders to elicit the requirements
- Stakeholders normally have different concerns, objectives and responsibilities. When multiple stakeholders participate in a discussion, requirements often conflict
- Maximum time, software engineer groups were utterly confused what are the required elements that need to be run after identifying the stakeholders. Although, the stakeholders play a major rule in a software product, the software engineering community does not find out how to identify and set priority of stakeholders, despite acknowledging that stakeholder is an important part of a successful software project, so less attention to stakeholders may cause for failing down a system

CONTRIBUTIONS

By achieving these objectives, the research makes the following contributions:

- A novel method to classify requirements into layers that change at different rates and its application to analyse the requirements of the software project selected to evaluate the work
- StakeNet, a novel method that uses social networks to identify and prioritise stakeholders
- StakeRare, a novel method that uses social networks and collaborative filtering to identify and prioritise requirements
- StakeSource, a novel web-based tool that automates the StakeNet Method
- The empirical evaluation of StakeNet and StakeRare using a real large-scale software project
- The empirical studies are the first of their kind in requirements elicitation
- The empirical studies are substantial, using post project knowledge to establish the ground truth of stakeholders and their requirements
- The evaluation provides clear evidence that the methods can identify a highly complete set of stakeholders and requirements and prioritise them accurately
- In addition, the methods are straight forward to use and require less time from the requirements engineers and stakeholders compared to the existing method used in the project
- Tool evaluation by practitioners is rare in requirements engineering research but essential to provide confidence that the tool works in practice
- StakeSource is used by practitioners in a large-scale software project and a university-wide research project and evaluated based on the feedback from the practitioners

EXISTING SYSTEM

In large scale software projects there will be a lot of clients who we can't able to meet often because they may be present in somewhere around in the country. Most of the elicitation methods need the customer interaction with the developer but this process needs more time and money. The existing method is lagging in elicitation of requirements from the stakeholders. Because of large number of customer many stakeholder requirements may be over looked. That means it is not taken into account. Because of requirement overlooked many important requirements are omitted in the sea of information.

Demerits of existing system

Information overload: Information overload is a huge problem in big software projects. These projects will always have many stakeholders and a lot of requirements. Users become unfulfilled when the software not meets their requirements. Customers who pay for the system may need to pay for the mistakes.

Biased prioritization of requirements: Occurs because current prioritization techniques depend on individual stakeholder who may not have a major part in large projects. As a result, important requirements known to only to some of the stakeholders can be lost in huge information.

Inadequate stakeholder input: Inadequate stakeholder input caused by selection of insufficient stakeholder neglecting stakeholders is one of the common mistakes in requirement elicitation.

LITERATURE REVIEW

During the software development life cycle, the actual requirement is collected from the potential stakeholders for a specific system. There are so many proposals on discovering all stakeholders of a specific system which can be domain-independent, effective and pragmatic. All of the references emphasis the importance of identifying stakeholders and although, they provide examples or broad guidance for identifying them, none describes a model or a concrete approach for identifying stakeholders for a specific system. Several approaches have done to satisfy these issues. A stakeholder in an organization is (by definition) any group or individual who can affect or is affected by the achievement of the organization's objectives. According to the freeman, it has been also said that "A stakeholder is anything influencing or influenced by the firm". But it indicates a wide set of identification of stakeholder. Because, there are so many people involved in an organization may be treat as a stakeholder but infrequent of them would have the aptitude to influence or being influenced by the organization itself (Wasserman and Faust, 1994).

On the other hand, the term stakeholder also means "all those who have a stake in the change being considered those who stand to gain from it and those who stand to lose".

Stakeholders are all those claimants inside and outside the firm who has a vested interest in the problem and its solution and are the concrete entities that affect and in turn are affected by a policy. They suggested ways of identifying stakeholders including: considering standard demographic groups (age, sex, etc.) for relevance; asking people who they consider to be the key stakeholders and studying accounts of ethnographic fieldwork to discover who seems to have a valid interest.

That's why when a project beginning, it is important to figure out the real stakeholder and make sure about their involvement until the project concluding. A set of stakeholder which can be refer as "baseline" stakeholders who can recognize "supplier" stakeholders and "client" stakeholders from this "baseline" stakeholder: the former provides information or supporting tasks to the baseline and the latter processes or inspects the products of the baseline. Other stakeholders that call interact of satellite with the baseline in a variety of ways. Interaction may include communicating, reading a set of rules or guidelines, searching for information and so on. Focus on interactions between stakeholders rather than relationships flanked by the system and the stakeholder because they are laid-back to follow. Baseline stakeholder is divided into four groups: users, developers, legislators and decision-makers (Herlocker *et al*., 2004).

Stakeholders in general can be classified into four types: primary, secondary, external and extended stakeholders. Primary stakeholders are vital since the outcomes of the project affect them directly and their interests in the proposed system are high. Missing any primary stakeholders can affect the project development and influence the achievement of the project goals. Primary stakeholders normally include individuals who have the power, authority and responsibility over the resources such as financial. Secondary stakeholders hold those who are affected by the project outcomes indirectly. They may be the consumers of a product or service.

Even though, they do not participate in software project development matters they monitor the fulfillment of their interests. External stakeholders are not directly a part of the software project team but they add values to the project from outside. Extended stakeholders could be anyone who is often helpful in assisting above-mentioned stakeholders to reach their visions.

Robertson states that the term stakeholder encompasses sponsors (needed for organizational commitment), consultants (knowledgeable either in technical aspects or the problem domain) and influencers (culture, law, inspectors, competition, ..., etc.). Finally, she extends the term to include users who she defines as one of many consumers. She adopts the term consumer to refer to the many different roles that a person may take and the impact they have on the success of the product, namely that of the buyer and users. People who play the role of buyers are those who decide whether or not to buy the product. Whereas, she identifies users as people who come into direct contact with the product (Sarwar *et al*., 2001).

The center of the theory is success-critical stakeholders" win-win theory W which address what values are significant and how success is guaranteed for a given software engineering organization. The four supporting theories that it draws upon are dependency theory (identifying all of the success-critical stakeholders), utility theory (Understanding how the success-critical stakeholders want to win), decision theory (Having the success-critical stakeholders negotiate win-win product and process plans) and control theory (Adaptively controlling progress toward a success-critical stakeholders win-win outcome) (Sharp *et al*., 1999; Zave, 1997).

Stakeholder mapping is an important step to understand who your key stakeholders are where they come from and what they are looking for in relationship to your business which can draw from multiple viewpoints to determine a key list of stakeholders from corner to corner the whole stakeholder spectrum. Gather a cross functional group of internal members and identify sources external that may have important knowledge about or perspective on the issues and reach out to these sources for input as a list of stakeholder. This list will alteration as the environment around you evolves and as stakeholders they make decisions or change their opinions. It is true that there is no magic list of stakeholder which will not change for any kind of software project. The final list will depend on your business, its impacts and your current engagement objectives-as a result it should not remain static (Charette, 2005).

At the beginning of any software project, the main task is to set clear project definitions that include project goals and system descriptions (type and domain). The software project goals specify what the business wants to achieve through the software project while the system descriptions define the characteristics of the system to be built. The definitions lead to the recognition of which types of stakeholders (primary, secondary, external and extended) are required. Stakeholders have specific roles. They can therefore, be categorized based on the roles that they are playing. Stakeholders may also be assigned to

more than one role. Moreover, roles have degrees of importance. Some stakeholders roles are more influential and significant than the others. These roles have higher chances of being considered in the next stage.

In order to determine which groups of stakeholders should be considered they can be classified into the following classes:

Mandatory (M): Stakeholders that must be included or else the success of the system is threatened.

Optional (O): Stakeholders that are not essentially selected. By neglecting their needs does not threaten the success of the system.

Nice-to-have (N): Stakeholders that do not influence the system's success if they are not selected. Due to its importance, primary stakeholders are considered as M regardless the roles that they are playing. Secondary stakeholders may fall under M or O, depending on their role's degrees of importance. External stakeholders are mainly O but they may become N if their involvement or roles on the way to the project are insignificant. Lastly, extended stakeholders fall under N as their contributions are quite minimal (Alexander and Robertson, 2004). Software projects have constraints which hinder project managers to include all possible stakeholders into a project. A kind of sorting process has to be established where certain aspects of the stakeholders enable them to be on top of the list. The sorting is called prioritization. Interpersonal skills are important to ensure an effective RE process.

The skills therefore, can be used as the final measures to succeed the selected stakeholders as the best possible participants. The skills include negotiation, collaboration and communication (written and oral). These three skills have to be considered holistically which can be measured through predetermined tests. Some possible prioritization techniques that can be adopted include the ones that are normally employed in prioritizing requirements such as Analytical Hierarchy Process (AHP) (Alexander and Robertson, 2004), Case-Based Ranking (CBR) (Alexander, 2007) and Hierarchical Cumulative Voting (HCV) (Lehtola *et al*., 2004). Reviewing the literatures which are based on identifying the stakeholders in requirement engineering, we understood that the sources of favorable requirements depend on wide range of stakeholders. In spite of several ideas to identify the right stakeholders, these techniques are kept down and isolated. It is not immaculate that how elements of project are interrelated between sources of requirements and stakeholders. We are supposed to discover a new model integrating these elements so that their effects on the matter can be seen expressly. On the other hand, there is no appropriate method available to prioritize identified stakeholder based on any value metric. Despite, stakeholder prioritization is another ticklish issue for a successful project because to elicit the collected requirements of a specific system, it can be conflicted for various approaches encountered from wide range of stakeholders. To get over this type of problems, we have to create a new effective model so that based on that model; a software team can easily prioritize the identified stakeholders regarding a specific system.

Requirements elicitation

Elicitation techniques: In requirements elicitation, traditional techniques such as interviews and focus groups, form the basis of existing practice (Sharp *et al*., 1999). In interviews, the requirements engineers approach stakeholders with questions to gain information about their needs. Focus groups brings takeholders together in a discussion group setting where they are free to interact with one another. These techniques are effective but require direct interaction between the requirements engineers and stakeholders. As such they are difficult to scale to a large number of stakeholders.

More advanced elicitation techniques improve the completeness and variety of the identified requirements by catalysing discussions and exploring the stakeholders' needs. These techniques include prototyping, metaphors (Saaty, 1980), storyboards (Berander and Jonsson, 2006) and model-driven techniquessuch as use cases, scenarios and goal models (Charette, 2005). Nevertheless, similar to traditional techniques they require face-to-face meetings, hence do notscale well to large projects.

Prioritisation techniques: Software projects often have more requirements than time, resource and budget allow for. As such requirements should be prioritised and managed so that those that are critical and most likely to achieve customer satisfaction can be selected for implementation (Zave, 1997; Charette, 2005). A prioritisation technique commonly used in practice is the numeral assignment technique. In this technique, each requirement is assigned a value representing its perceived importance. For example, requirements can be classified as mandatory, desirable or inessential (Sarwar *et al*., 2001). Numeral assignment is straightforward but a study by Karlsson found that the participants' sentiments about the numbers in the numeral assignment method differ

and the scoring system is often unpredictable as different people make use of different personal scales. Nevertheless, this method is widely used due to its simplicity.

Another popular method is the pairwise comparison approach (Lehtola *et al*., 2004). In this method, requirements engineers compare two necessities to determine the more important one which is then entered in the corresponding cell in the matrix (Lehtola *et al*., 2004). The comparison is repeated for all requirements pairs such that the top half of the matrix is filled. If both requirements are equally significant then they together appear in the cell. Then, each requirement is ranked by the number of cells in the matrix that comprise the requirement. The pairwise comparison is simple. However, since all unique pairs of necessities need to becom pared, the effort is substantial when there are many requirements (Zave, 1997). The prioritising n requirements needs n! (n-1)/2 comparisons. Hence, a project with 100 requirements would require 4,950 comparisons. Many existing approaches including those previously mentioned, prioritise requirements from an individual's perspective. Other similar approaches include the cost value approach which prioritises requirements based on their relative value and implementation cost and the Value-Oriented Prioritisation Method which prioritises requirements based on their contribution to the core business values and their perceived risks (Wasserman and Faust, 1994). As prioritisations involve a small subset of stakeholders, the results are biased towards the perspective of those involved in the process. More sophisticated methods combine prioritisations from multiple stakeholders. In the 100-point test, each stakeholder is given 100 points that they can distribute as they desire among the requirements. Requirements that are more important to a stakeholder are given more points. Requirements are then prioritised based on the total points allocated to them. The 100-point test incorporates the concept of constraint in the stakeholder's prioritization by giving each of them a limited number of points. One criticism of this approach is that it can be easily manipulated by stakeholders seeking to accomplish their own objectives. For example, stakeholders may distribute their points based on how they think others will do it (Price and Cybulski, 2005). In addition, it is difficult for stakeholders to keep an overview of a large number of requirements (Anwar *et al*., 2011). In the requirements triage method, Davis proposed that stakeholders should be gathered in one location and group voting mechanisms used to prioritise requirements. One method to collect group vote is to use the show of fingers to indicate the stakeholders' enthusiasm for a requirement. A disadvantage is the relative priorities of requirements depend on the stakeholders who attended the prioritisation meeting and dominant participants may influence the prioritisation. In the win-win approach proposed by Boehm, stakeholders negotiate to resolve disagreements about candidate requirements (Alexander and Robertson, 2004; Sharp *et al*., 1999). Using this approach, each stakeholder ranks the requirements privately before negotiations start. They also consider the requirements they are willing to give up on. Stakeholders then work collaboratively to forge an agreement through identifying conflicts and negotiating a solution. Win-win negotiations encourage stakeholders to focus on their interest rather than positions, negotiate towards achieving mutual gain and use objective criteria to prioritise requirements.

Nevertheless, the approach is labour intensive, particularly in large projects. Another method that involves multiple stakeholders isthe value, cost and risk method proposed by Wiegers. In Wiegers' method, the customer representatives estimate the value of each requirement which is the relative benefit each requirement provides to them and the relative penalty they suffer if the requirement is not included. The project team estimates the relative cost of implementing each requirement and the relative degree of risk associated with each requirement. The priority of each requirement is calculated from its value, cost and risk such that requirements at the top of the list have the most favourable balance of the three elements. This technique is limited by the individual's ability to determine the value, cost and risk for each requirement. Many existing prioritisation methods consider requirements to have a flat structure and be independent of one another (Roy, 1996). However, requirements are often defined at different levels of abstraction. For example, a high-level requirement can be refined into several specific requirements (Wood and Silver, 1995; Woolridge *et al*., 2007; Sommerville and Sawyer, 1997; Stark *et al*., 1999). Hierarchical Cumulative Voting (HCV) proposed by Berander and Jonsson enables prioritisations to be performed at different levels of a hierarchy. Stakeholders perform prioritisation using 100-point test within each prioritisation block. The intermediate priorities for the requirements are calculated based on the characteristics of the requirements hierarchy. Final priorities are calculated for all requirements at the level of interest through normalisation. If several stakeholders have prioritised the requirements their individual results are then weighted

and combined. When doing so, different stakeholders may have different weights. Although, the hierarchical prioritisation in HCV makes it easier for the stakeholders to keep an overview of all the requirements, the prioritisations need to be interpreted in a rational way as stakeholders can easily play around with the numbers.

There is a plethora of methods to prioritise requirements such as multi-attribute utility theory, top 10 requirements, outranking (Sarma *et al*., 2009), minimal spanning tree, cost benefit analysis (Sarwar *et al*., 2001) and quality function deployment (Nurmuliani *et al*., 2004). Many of these methods have similar short comings: significant effort is required when there are many requirements and the requirements' priorities are easy to manipulate. For example, the quality function deployment suggests the limit of 30 requirements (Nuseibeh and Easterbrook, 2000). Cost benefit analysis relies on the type of costs included in the analysis by the decision-makers which maybe biased due to their vested interest (Sarwar *et al*., 2001). One of the few methods that can scale to a large number of requirements is the Binary Search Tree (BST) (O'Neil *et al*., 1986). In BST, a requirement from the set of requirements is selected as the root node. Then, a binary tree is constructed by inserting less important requirements to the left and more important ones to the right of the tree. A prioritised list of requirements is generated by traversing the BST in order. The output is a prioritised list of requirements with the most important requirements at the start of the list and the least important ones at the end. This method is simple to implement but provides only a simple ranking of requirements as no priority values are assigned to the requirements.

For projects with many requirements, recent work by Macaulay (1993) propose Pirogov which uses data mining and machine learning techniques to support requirements prioritisation. Pirogov uses various clustering techniques to organise requirements into different categories. The requirements engineers then prioritise the clusters and determine the importance of each clustering technique. Using the information, Pirogov generates a list of prioritised requirements. By automatically clustering the requirements into different categories, Pirogov reduces the number of manual prioritisations. Itis a significant step towards large-scale requirements elicitation. But at the moment, the results of prioritisation depend on the requirements engineers' subjective prioritization of the clusters and clustering techniques.

Social network analysis: Social network analysis is the application of methods to understand the relationships among actors and on the patterns and implications of the relationships (Ohira *et al*., 2005). In social network analysis, actors are discrete individuals, corporate or collective social units such as employees within a department, departments within a corporation and private companies in a city (Ohira *et al*., 2005). These actors are linked to one another by relational or social ties such as evaluation of one person by another (e.g., friendship orrespect), transfers of material resources (e.g., business transaction) and formal relations (e.g., authority) (Ohira *et al*., 2005). In social network analysis, the snowballing method proposed by Goodman is used to sample social network data for large networks where the boundary is unknown (Ohira *et al*., 2005). It is also used to track down "special" or "hidden" populations such as business contact networks, community elites and deviant sub-cultures. Snowball sampling begins with a set of actors (Ohira *et al*., 2005). Each of these actors is asked to nominate other actors. Then, new actors who are not part of the original list are similarly asked to nominate other actors. As the process continues, the group of actors builds up like a snowball rolled downa hill. The process continues until no new actors are identified, time or resources have run out or when the new actors being named are very marginal to the actor setunder study.

A social network is a structure that consists of actors and the relation(s) defined on them (Ohira *et al*., 2005). It is often depicted as a graph in which the actors are represented as nodes and the relationships among the pairs of actors are represented by lines linking the corresponding nodes (Ohira *et al*., 2005). The graph can be binary or valued, directed or undirected, depending on the relations between the actors. If the relations are directed then the links have direction and if the relations are valued, the links have weights attached to them. Using graph structures to represent social networks enables large sets of social network data to be visualised.

The centrality of actors in their social networks is of great interest to social network analysts. Actors that are more central have a more favourable position in the network. For example, in a friendship network, an actor who is connected to many actors in the network is popular. In a business contact network an actor that sits in between clusters of networks has high influence on the information that passes between the clusters. A number of different social network measures have been developed to measure the centrality of social network actors such as betweenness centrality, load centrality, degree centrality, in-degree centrality and out-degree centrality (Ohira *et al*., 2005). In requirements engineering, Damian used social network analysis to study collaboration, communication and awareness among project team members. The nodes

were members of the development team who are working on assigned to or communicating about the requirements in the project. Social network measures such as degree centrality and betweenness centrality were used to analyse the collaboration behaviour. Forexample, degree centrality indicated active members and betweenness centrality indicated members who control interactions between other members.

Collaborative filtering: Collaborative filtering is a technique to filter large sets of data for information and patterns (Muller *et al*., 2001). This technique is used in recommender systems to forecast a user's preference on an item by collecting preference information from many users (Well and Myers, 2003). For example, Amazon 1 uses collaborative filtering to recommend books to their customers and MovieLens2 uses it to recommend movies (Well and Myers, 2003; Nas, 1997). The underlying assumption is that users who have had similar taste in the past will share similar taste in the future.

In collaborative filtering, users are the individuals who provide ratings to a system and receive recommendations from the system. Items can consist of anything for which ratings can be provided such as art, books, songs, movies, vacation destinations and jokes (Newman, 2005). A rating is an umerical representation of a user's preference for an item. A profile is the set of ratings that a particular user has provided to the system. Collaborative filtering systems take a set of ratings from the user community as input, use this set of ratings to predict missing ratings and use the predictions to create a list of items that is personalized for each user. This list of items are then presented to the user as recommendations.

To produce predictions, collaborative filtering systems use a variety of algorithms. One of the most well-known algorithms is the k-Nearest Neighbour (kNN) algorithm (Well and Myers, 2003; Liu and Yu, 2004). kNN is used to identify like-minded users with similar rating histories in order to predict ratings for unobserved users-item pairs (Liu and Yu, 2004). kNN first finds a unique subset of the community for each user by identifying those with similar interests. To do so, every pair of user profile is compared to measure the degree of similarity. A popular method is Pearson's correlation coefficient which measures the degree of linearity between the intersection of the pair of users' profiles (Well and Myers, 2003). Then, a neighbourhood is created for each user by selecting the k most similar users. The similarity between each pair of user profiles for users in the neighbourhood is used to compute predicted ratings. Finally, the predicted ratings for the items are sorted according to the predicted value and the top-N items are proposed to the user as recommendations where N is the number of items recommended to the user (Well and Myers, 2003).

In requirements engineering, Castro-Herrera uses collaborative filtering to facilitate online discussions for requirements identification (Lopez-Fernandez *et al*., 2004; Low and Jeffery, 1990). Their method, named Organiser and Promoter of Collaborative Ideas (OPCI), uses clustering to group the stakeholder's ideas into an initial set of discussion forums and construct a stakeholder profile for each stakeholder. These profiles are used by the kNN algorithm to identify stakeholders with similar interests and suggest additional forums that might be of interest to the stakeholders. By recommending suitable forums to stakeholders, OPCI aims to encourages takeholders to contribute to relevant forums and increase the quality of the elicited requirements. OPCI uses collaborative filtering to recommend forums of interest to stakeholders. It has inspired the work described in this study to use collaborative filtering to recommend requirements of interest to stakeholders in order to support large-scale requirements elicitation. Recommending relevant requirements to stakeholders can reduce the number of requirements each stakeholder has to identify and prioritised while still ensuring they are aware of the requirements they may be interested in.

STAKERATREET: OUR PROPOSED APPROACH

Large projects tend to be beset by three problems: information overload, inadequate stakeholder input and biased prioritisation of requirements. StakeRatreetis a method that uses social networks and collaborative filtering to elicited requirements in large software projects. To address the problem of in adequate stakeholder input, StakeRatreet aims to be open and inclusive, so that are presentative sample of stakeholders participates in the requirements elicitation process. As stakeholders are socially related to one another they can be identified and prioritised using their relations. StakeRatreet exploits previous work to do this, i.e., StakeRare, StakeNet and StakeSource. The previous work asks stakeholders to recommend other stakeholders, builds a social network with stakeholders as nodes and their recommendations as links and prioritises stakeholders from aglobal perspective using social network measures.

Stakeratreet concepts: StakeRatreet stands for stakeholder and recommender-aid technique for requirements engineering elicitation technique. To avoid overloading stakeholders with data, StakeRatreet uses

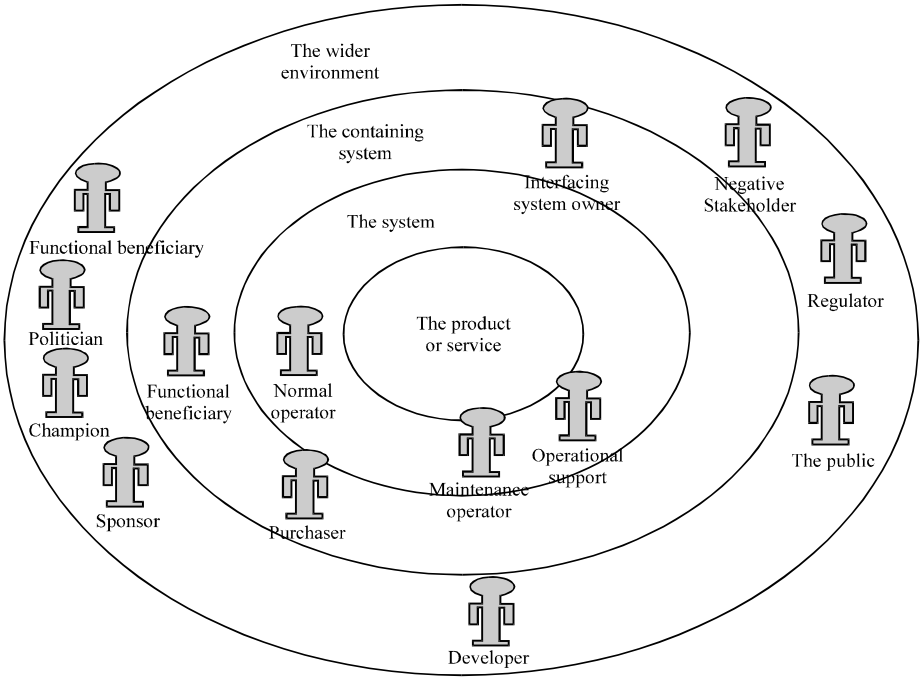

Fig. 1: Onion diagram representing stakeholder roles

collaborative filtering to present only the requirements that are relevant to them. StakeRatreet asks each stakeholder to rate an initial list of requirements and based on the list, identifies a neighbourhood of similars take holders for each stakeholder. Then, it forecasts other relevant requirements for the stakeholder based on the requirements providing by similar stakeholders. These predictions are presented to the stakeholder to be approved and added into their set of ratings. To avoid overloading the requirements engineers with information, StakeRatreet prioritises stakeholders and their requirements. Finally, to avoid biased prioritisation of requirements, StakeRatreet produces a prioritised list of requirements based on each stakeholder's ratings and their influence in the project. The stakeholders' influence in the project is produced from a global perspective by running the social network measures on the stakeholder network. StakeRatreet has four steps (Fig. 1).

Step 1 (identify and prioritise stakeholders): Step 1 identifies and prioritises the stakeholders based on their inspiration in the project (Fig. 1). Stakeholders have to be identified as they are the source of requirements. They have to be prioritised as their level of influence in the project affects the priority of their requirements. The output is a prioritised list of stakeholder roles and for each role, aprioritised list of stakeholders. StakeRatreet uses StakeRare for step 1. StakeRare is a previously published stakeholder analysis method that produces such an

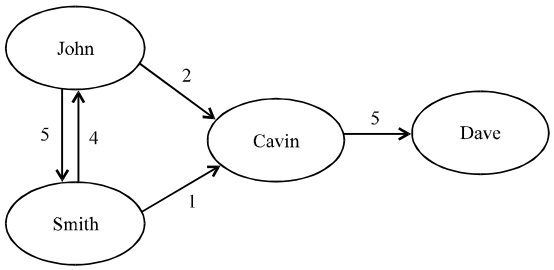

Fig. 2: Example stakeholder network

output. StakeRare identifies an initial set of stakeholders and asks them to recommend other stakeholders and stakeholder roles. A good word is a triple <stakeholder, stakeholder role, salience> wheres alience is a number on an ordinal scale (e.g., 1-5). Forexample, in a software project to implement a university access control system, John, a stakeholder representing the role of Estates that manages the university's physical estate can make a recommendation <Smith, Library, 4>. StakeRare then asks smith to recommend other stakeholders. Based on the stakeholders' recommendations, StakeRare builds a social network with stakeholders asnodes and their recommendations as links. An example stakeholder network is illustrated in Fig. 2. Finally, StakeRare applies various social network measures such as betweenness centrality, degree centrality and closeness centrality to prioritise the stakeholders in the network. The social

network measures produce ascore for each stakeholder. The stakeholder roles are prioritised by the highest score of their stakeholders. An example output is illustrated. Fractional rankingor "1 2.5 2.5 4" ranking (Maguire and Bevan, 2002) is used such that if a tie in ranks occurs, the mean of the ranks involved is assigned to each of the tied items. For example, if Estates and students have the same level of influence then the ranks become Estates: Rank 1.5, students: Rank 1.5, library: Rank 3.

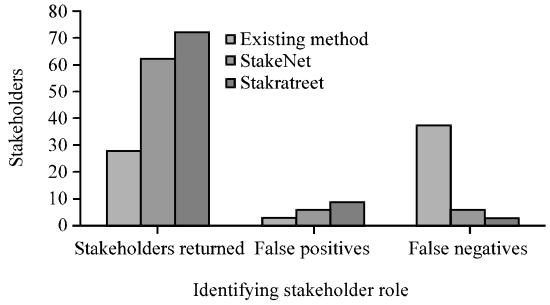

Fig. 3: Identifying stakeholder roles: precision and recall

Step 2 (collect profile): Step 2 collects a profile from each stakeholder identifiedin step 1 (Fig. 2). Existing elicitation methods in the background section such as interviews with a subset of stakeholders or focus groups can be used to identify an initial list of requirements. Using the university access control software project in step 1 as an example, a direct interview with John from Estates revealed that one of the software project objectives is to provide "better user experience". Smith representing the library reveals that his requirement is "to combine librarycard with access card", student Dave's requirement is "to combine access card with bank card" and John, representing the Estates, requests for "all in one card". As mentioned in the background section, requirements can be defined at different levels of abstraction and ahigh-level requirement can be refined into several specific requirements (Wood and Silver, 1995; Saaty, 1980). In this example, the requirements are organised into a hierarchy of three levels: project objective, requirement and specific requirements. Achieving all the specific requirements means that the parent requirementis achieved and achieving all the parent requirements means that the software project objective is achieved. For example, the requirement "all in one card" falls under the software project objective "better user experience" as it is easier to carry one card for all purposes (Fig. 3). Then, combining the various cards are specific requirements under "all in one card".

A preference is a triple<stakeholder, requirement, rating> where rating is a number on an ordinal scale (e.g., from -5 to +5) reflecting the importance of the requirement to the stakeholder (e.g., 0 is unimportant and 5 is very important). For example, John provides a preference<John to combine library card with access card, 5>. Stakeholders can also indicate requirements that they actively do not want (e.g., by rating the requirement an X). For example, Smith provides a preference<smith to combine access card with bank card, X>. Stakeholders can also rate requirements not in the list by adding their own requirements. The requirements added are then available to be rated by other stakeholders. If a requirement provided by a stakeholder does not have any specific requirements, specific requirements can be identified using existing elicitation methods (e.g., interviews) and added to the list to be rated. Finally, StakeRatreet propagates the ratings of requirements to avoid missing values. Rating propagation enables StakeRatreet to make prioritisations and predictions at different levels of detail. If a stakeholder rates a high-level requirement but does not rate the lower-level requirements then his rating propagates down to the lower-level requirements. For example, Cavin provides a preference<Cavin, all in one card, 4>.

Since, Smith and Dave provided specific requirements for this requirement, Cavin then implicitly provides two other preferences<Cavin to combine library card and access card, 4> and <Cavin to combine access card with bank card, 4>. This propagation assumes that specific requirements when unrated by the stakeholder have the same rating as their parent requirement and the stakeholder agrees with the decomposition of the requirement into the specific requirements. Similarly, if a stakeholder rates a lower-level requirement but does not rate the high-level requirement then his rating propagates up to the high level requirement. This propagation assumes that if a stakeholder cares about a specific requirement, they would care equally about the parent requirement (Table 1). If more than one specific requirement is rated then the maximum rating is propagated. Step 3: predict requirements based on the stakeholders' profile, step 3 uses collaborative filtering to predict other requirements that each stakeholder needs or actively does not want (Fig. 1).

StakeRatreet uses the k-Nearest Neighbour (kNN) algorithm described in the background section. Cross-validation is used to find the optimal value for k. kNN finds similar stakeholders by measuring the similarity between the stakeholders' profiles. Then, it generates the predicted level of interest that a stakeholder will have in a requirement that he has not yet rated. StakeRatreet returns requirements that may be relevant to the

Table 1: Example prioritised list of stakeholders

Prioritised stakeholder roles	Prioritised stakeholders
Estates	John
Students	Dave
	Cavin
Library	Smith

stakeholder (i.e., requirements with the highest predicted level of interest) as recommendations at all three levels. Stakeholders can then rate the requirements that are recommended to them, provide new requirements or rate other requirements. The new ratings by the stakeholders are then added to their profiles.

Then, step 3 is repeated with the updated profiles. Step 3 can be repeated until no new ratings and requirements are provided by stakeholders after one round of recommendations.

Step 4 (prioritise requirements): For the final step, StakeRatreet aggregates all the stakeholders' profiles into a prioritised list of requirements (Fig. 1). The ratings from the stakeholders' profiles and the priority of the stakeholders and their roles from step 1 are used to prioritise requirements. Negative ratings (from as takeholder actively not wanting a requirement) are excluded in the calculation as their purpose is to highlight conflicts to the requirements engineers, rather than to prioritise the requirements. To calculate the importance of a requirement in a project, the influence of the stakeholder's role in the project is determined and then the influence of the stakeholders in their roles is determined as follows. The influence of stakeholder i's role in the project is calculated using Eq. 1.

RESULTS

The method to evaluate each research question and the results are described as follows. The precision of identified stakeholder roles is the number of actual stakeholder roles in the set of identified stakeholder roles divided by the total number of identified stakeholder roles (Eq. 1):

$$\text{Precision} = \frac{|\{X\}\{\text{Ground truth}\}|}{|\{X\}|} \quad (1)$$

Where:

X = Set of stakeholder roles identified by Stakeratreet or the existing method

Ground truth = Set of stakeholder roles in the ground truth

The recall of identified stakeholder roles is the number of actual stakeholder roles in the set of identified stakeholder roles divided by the total number of actual stakeholder roles (Eq. 2):

Table 2: Identifying stakeholder roles (precision and recall)

Method	Stakeholders returned	False positives	False negatives
Existing method	28	3	37
StakeNet	62	6	6
Stakratreet	72	9	3

$$\text{Recall} = \frac{|\{X\}\{\text{Ground truth}\}|}{|\{\text{Ground truth}\}|} \quad (2)$$

Both precision and recall range from 0-1. Precision of 1 means all the identified roles are actual stakeholder roles. Recall of 1 means all the actual stakeholder roles are identified (Table 2). For example, the ground truth has 62 stakeholder roles and the existing method list has 28 stakeholder roles. The number of stakeholder roles in the existing method list that matches the stakeholder roles in the ground truth is 25.

Here, compare to other approach, StakeRatreet is better performance than other in terms of precision and recall.

CONCLUSION

StakRatreet uses social networks to identify and prioritize software project stakeholders and their roles. By applying the StakeRatreet to ralic dataset that Stakeratreet is better than other approach. StakeRatreet performs better in this project interms of recall and precision it identifying the stakeholders and their roles. Here in this work, developed a software tool that implemented in StakeRatreet.

REFERENCES

Albrecht, A.J. and J.E. Gaffney, 1983. Software function, source lines of code and development effort prediction: A software science validation. IEEE Trans. Software Eng., SE-9: 639-648.

Alexander, I. and S. Robertson, 2004. Understanding project sociology by modeling stakeholders. Software IEEE, 21: 23-27.

Alexander, I., 2007. A taxonomy of stakeholders: Human roles in system development. Int. J. Technol. Hum. Interact., 1: 23-59.

Anwar, F., R. Razali and K. Ahmad, 2011. Achieving effective communication during requirements elicitation: A conceptual framework. Commun. Comput. Inf. Sci., 181: 600-610.

Berander, P. and P. Jonsson, 2006. Hierarchical Cumulative Voting (HCV): Prioritization of requirements in hierarchies. Int. J. Software Eng. Knowl. Eng., 16: 819-849.

Charette, R.N., 2005. Why software fails. IEEE Spect., 42: 36-49.

Herlocker, J.L., J.A. Konstan, L.G. Terveen and J. Reidl, 2004. Evaluating collaborative filtering recommender systems. ACM Trans. Infrom. Syst., 22: 5-53.

Lehtola, L., M. Karuppinen and S. Kujala, 2004. Requirements Prioritization Challenges in Practice. In: Product Focused Software Process Improvement. Bomarius, F. and H. Iidapp (Eds.). Springer Berlin, Heidelberg, Germany, ISBN: 978-3-540-21421-2, pp: 497-508.

Liu, L. and E. Yu, 2004. Designing information systems in social context: A goal and scenario modelling approach. Inf. Syst., 29: 187-203.

Lopez-Fernandez, L., G. Robles and J.M. Gonzalez-Barahona, 2004. Applying social network analysis to the information in CVS repositories. Proceedings of the International Workshop on Mining Software Repositories, May 25, 2004, Edinburgh, UK., pp: 101-105.

Low, G.C. and D.R. Jeffery, 1990. Function points in the estimation and evaluation of the software process. Software Eng. IEEE Trans., 16: 64-71.

Macaulay, L., 1993. Requirements capture as a cooperative activity. Proceedings of IEEE International Symposium on Requirements Engineering, January 4-6, 1993, San Diego, CA., pp: 174-181.

Maguire, M. and N. Bevan, 2002. User Requirements Analysis. In: Usability. Hammond, J., T. Gross and J. Wesson (Eds.). Springer-Verlag, New York, USA., pp: 133-148.

Muller, H., W. Muller, D.M.G. Squire, S. Marchand-Maillet and T. Pun, 2001. Performance evaluation in content-based image retrieval: Overview and proposals. Pattern Recog. Lett., 22: 593-601.

Nas, T.F., 1997. Cost-benefit analysis: Theory and application. Eval. Program Plann., 3: 288-290.

Newman, M.E.J., 2005. A measure of betweenness centrality based on random walks. Social Networks, 27: 39-54.

Nurmuliani, N., D. Zowghi and S.P. Williams, 2004. Using card sorting technique to classify requirements change. Proceedings of the 12th IEEE International Requirements Engineering Conference, September 6-11, 2004, Kyoto, Japan, pp: 240-248.

Nuseibeh, B. and S. Easterbrook, 2000. Requirements engineering: A roadmap. Proceedings of the Conference on The Future of Software Engineering, June 4-11, 2000, Limerick, Ireland, pp: 35-46.

O'Neil, R.V., D.L. De Angelis, J.B. Waide and G.E. Allen, 1986. A Hierarchical Concept of Ecosystems. Princeton University Press, Princeton, New Jersey, ISBN-10: 0691084378.

Ohira, M., N. Ohsugi, T. Ohoka and K. Matsumoto, 2005. Accelerating cross project knowledge collaboration using collaborative filtering and social networks. ACMSIGSOFT Software Eng. Notes, 30: 1-5.

Price, J. and J. Cybulski, 2005. Consensus making in requirements negotiation: The communication perspective. Aust. J. Inf. Syst., 13: 209-224.

Roy, B., 1996. Multicriteria Methodology for Decision Aiding. Springer, New York, USA., ISBN-13: 9780792341666, Pages: 292.

Saaty, T.L., 1980. The Analytic Hierarchy Process: Planning Setting Priorities, Resource Allocation. 7th Edn., McGraw-Hill International Book Co., New York, USA., ISBN: 0070543712.

Sarma, A., L. Maccherone, P. Wagstrom and J. Herbsleb, 2009. Tesseract: Interactive visual exploration of socio-technical relationships in software development. Proceedings of the 31st International Conference on Software Engineering, May 16-24, 2009, Vancouver, BC., pp: 23-33.

Sarwar, B., G. Karypis, J. Konstan and J. Reidl, 2001. Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web, May 1-5, 2001, Hong Kong, China, pp: 285-295.

Sharp, H., A. Finkelstein and G. Galal, 1999. Stakeholder identification in the requirements engineering process. Proceeding of the Tenth International Workshop on Database and Expert Systems Applications. September 1, 1999, Florence, Italy, pp: 387-391.

Sommerville, I. and P. Sawyer, 1997. Requirements Engineering: A Good Practice Guide. 1st Edn., John Wiley and Sons Inc., Chichester, ISBN: 0471974447.

Stark, G.E., P. Oman, A. Skillicorn and A. Ameele, 1999. An examination of the effects of requirements changes on software maintenance releases. J. Software Maintenance, 11: 293-309.

Wasserman, S. and K. Faust, 1994. Social Network Analysis. Cambridge U. Press, Cambridge.

Well, A.D. and J.L. Myers, 2003. Research Design and Statistical Analysis. 2nd Edn., Taylor and Francis, USA., ISBN-13: 9781135641078, Pages: 736.

Wood, J. and D. Silver, 1995. Joint Application Development. 2nd Edn., John Wiley and Sons, Inc., New York, USA., Pages: 402.

Woolridge, R.W., D.J. McManus and J.E. Hale, 2007. Stakeholder risk assessment: An outcome-based approach. Software IEEE, 24: 36-45.

Zave, P., 1997. Classification of research efforts in requirements engineering. ACM Comput. Surv., 29: 315-321.