



International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcse.com

Innovative Process for Macroscopic Requirement Gathering

Dr Nalini Chidambaram, Sengalvarayan Muthappillai
Bharath University,
Chennai, India.

Abstract: Requirement gathering is the practice of collecting the requirements of a system from users, customers and other clients. Requirement gathering process will identify and prioritize requirements. This process difficult in macroscopic software projects with many clients. Here will introduce an innovative method that uses social networks and collective filtering to manage the requirement gathering process in macroscopic. Collective filtering is done using an improved k -nearest neighbour algorithm where the existing system will be using k -nearest neighbour algorithm. In this method network formation process plays an important role, it identifies clients and the client needs to recommend other clients and client roles, by continuing this process we will obtain a social network where clients acts as a nodes and their recommendations act as links. In the social network the project influence is determined by using variety of social network measures. After that every client in network needs to rate their initial set of requirements, then this method will recommends other relevant requirements to them using collective filtering, and prioritizes their requirements using their ratings and their project influence. So this innovative method predicts client needs accurately, completely and accurately prioritized list of requirements compared to the existing methods, and this method will prioritize requirements within the time [1].

Keywords — Requirements Specifications, Gathering Methods, Requirements Prioritization, Recommender Systems, Social Network Analysis, Client Analysis.

I. Introduction

Software systems are growing. The increase in size extends beyond mere lines of code or number of modules. Today, projects to build macroscopic software systems involve vast numbers of clients the individuals or groups that can influence or be influenced by the success or failure of a software project. These clients 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. In macroscopic projects, these clients cut across divisions and organizations. They have diverse needs which may conflict. Requirement gathering is the software engineering activity in which client needs are understood. It aims to identify the purpose for which the software system is intended. It involves identifying clients and prioritizing them based on their influence in the project. It also involves identifying requirements [2] from these clients and prioritizing their requirements. Stake Rare is a method to identify and prioritize requirements using social networks and collective filtering. It aims to address three problems are omitted and their requirements overlooked that beset macroscopic requirements gathering: information overload, inadequate client input, and biased prioritization of requirements. Information overload is inevitable in big projects. These projects tend to have many clients and requirements [1]. Existing methods for requirements gathering require intensive interactions with the clients, for example, through face-to-face meetings, interviews, brainstorming sessions, and focus groups [1]. These methods lack the means to manage the information gathered from clients. As such, the methods fail to scale to big projects with hundreds, thousands, or even hundreds of thousands of clients. Practitioners struggle to use these methods in macroscopic projects. Inevitably, clients are omitted and their requirements overlooked. Users become frustrated when the software fails to meet their needs. Customers who pay for the project pay for the mistakes.

Problem Statement

In macroscopic software project many clients who must be heard, but are unable to meet, possibly due to sheer numbers, dispersed locations, or lack of time. Most requirements gathering methods require face-to-face meetings with the clients and hence is time consuming when there are many clients.

Existing methods for requirements gathering require intensive interactions with the clients, for example, through

face-to-face meetings, interviews, brainstorming sessions, and focus groups. These methods lack the means to manage the information gathered from clients. As such, the methods fail to scale to macroscopic projects with hundreds, thousands, or even hundreds of thousands of clients. Practitioners struggle to use these methods in macroscopic projects. Inevitably, clients are omitted and their requirements overlooked [1].

Objective of the Work

In macroscopic software project many clients who must be heard, but are unable to meet, possibly due to sheer numbers, dispersed locations, or lack of time. Most gathering methods require face-to-face meetings with the clients and hence is time consuming when there are many clients. Existing methods for Requirements gathering require intensive interactions with the clients, for example, through face-to-face meetings, interviews, brainstorming sessions, and focus groups. These methods lack the means to manage the information gathered from clients. As such, the methods fail to scale to big projects with hundreds, thousands, or even hundreds of thousands of clients. Practitioners struggle to use these methods in macroscopic projects. Inevitably, clients are omitted and their requirements overlooked.

II. Existing System

In macroscopic software project many clients who must be heard, but are unable to meet, possibly due to sheer numbers, dispersed locations, or lack of time. Most gathering methods require face-to-face meetings with the clients and hence is time consuming when there are many clients. Existing methods for requirements gathering require intensive interactions with the clients, for example, through face-to-face meetings, interviews, brainstorming sessions, and focus groups. These methods lack the means to manage the information gathered from clients. As such, the methods fail to scale to big projects with hundreds, thousands, or even hundreds of thousands of clients. Practitioners struggle to use these methods in macroscopic projects. Inevitably, clients are omitted and their requirements overlooked.

Demerits of Existing System

Information overload: These methods lack the means to manage the information gathered from clients. Clients are omitted and their requirements overlooked. Users become frustrated when the software fails to meet their needs. Customers who pay for the project pay for the mistakes. Biased prioritization of requirements: Occurs because current prioritization practices depend on individuals who may not have a global perspective in macroscopic projects. As a result, important requirements [1] known to only a few clients can be lost in the sea of information. Inadequate client input: Inadequate client input caused by inadequate client selection omitting clients is one of the most common mistakes in software engineering.

III. Proposed System

Proposed system introduces an innovative method that uses social networks and collaborative filtering to support requirements gathering in macroscopic software projects. Stake Rare uses social networks to identify [1] [2] and prioritize clients and their roles in the project. The development of Stake Rare uses social networks and collaborative filtering to support requirements gathering in macroscopic software projects. Stake Rare uses social networks to identify and prioritize clients and their roles in the project. Then, it asks the clients to rate an initial list of requirements, recommends other relevant requirements [3] to them using collaborative filtering, and prioritizes their requirements using their ratings weighted by their project influence derived from their position on the social network.

Metrics of Proposed System

Information overload: To recommend relevant requirements to clients, and prioritizing the clients and requirements. Inadequate client: To recommend other clients, and asking all clients to provide requirements. Biased prioritization: Using the clients' ratings on the requirements and their position on the social network.

IV. Proposed System In Detail

Identify and Prioritize Client

The first step to identify [2] and prioritize the network is social network analysis which is done by snowballing method. Snowball sampling begins with a set of actors each of these actors is asked to recommend other actors. A recommendation is a triple <Client, client role, salience> Salience-is a number in ordinal scale (1-5) 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 down a hill. The process continues until no new actors are identified. It applies various social network measures such as between ness centrality, degree centrality, and closeness centrality, to prioritize the clients in the network. The social network measures produce a score for each client. The client roles are prioritized by the highest score of their clients.

Collect Profile

The clients identified in previous stage are asked to provide their preferences on the initial requirements [3]. A preference is a triple: <client; requirement; rating> where rating is a number on an ordinal scale (e.g., 0-5) reflecting the importance of the requirement to the client (e.g., 0 is unimportant and 5 are very important). E.X <Alice; to combine library card with access card; 5> Clients can also indicate requirements that they actively do not want by rating the requirement an X. <Bob; to combine access card with bank card; X>

Requirement Gathering

Collaborative filtering is a technique to filter large sets of data for information and patterns. This technique is used in recommender systems to forecast a user's preference on an item by collecting preference information from many users. k-Nearest Neighbor algorithm: kNN [7] is used to identify like-minded users with similar rating histories in order to predict ratings for unobserved users-item pairs kNN first finds a unique subset of the community for each user by identifying those with similar interests.

Requirement Prioritization

Pair wise comparison: In this approach, requirements engineers compare two requirements to determine the more important one, which is then entered in the corresponding cell in the matrix. The comparison is repeated for all requirements pairs such that the top half of the matrix is filled. If both requirements are equally important, then they both appear in the cell. Then, each requirement is ranked by the number of cells in the matrix that contain the requirement. Pair wise comparison is simple. Prioritizing n requirements needs $n*(n-1)/2$ comparisons. BST: Binary search tree [9]: In this approach 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 one to the right of the tree. A prioritized list of requirements is generated by traversing the BST in order. The output is a prioritized list of requirements, with the most important requirements at the start of the list and the least important ones at the end.

V. KNN Algorithm

kNN is used to identify like-minded users with similar rating histories in order to predict ratings for unobserved users-item pairs kNN first finds a unique subset of the community for each user by identifying those with similar interests. To produce predictions, collaborative filtering systems use a variety of algorithms. One of the most well-known algorithms is the 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. 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 intersections of the pair of users' profiles. Then, a neighborhood is created for each user by selecting the k most similar users. The similarity between each pair of user profiles for users in the neighborhood 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.

Collaborative Filtering Algorithm

StakeRare uses collaborative filtering to recommend relevant requirements to a client. The evaluation uses mean absolute error, which measures the deviation between a Predicted rating and the actual rating provided by the client, rather than the relevance of the recommended requirements.

Collaborative filtering is used to predict other requirements a client may need based on the profile they provide in Step 2. This step was evaluated using the standard evaluation method in the collaborative filtering literature. The evaluation partitioned the clients' profiles into two subsets. The first subset was the training set, which the collaborative filtering algorithm learned from. The second subset was the test set, with rating values that were hidden from the algorithm. For the evaluation, the algorithm's task was to make predictions on all the items in the test set. The predictions were then compared to the actual hidden rating values. Using this method of evaluation, no additional input was required from the clients.

VI. System Architecture

The proposed system uses social networks and collaborative filtering to identify [2] and prioritize requirements in macroscopic software projects. StakeRare identifies clients and asks them to recommend other clients and client roles, builds a social network with clients as nodes and their recommendations as links, and prioritizes clients using a variety of social network measures to determine their project influence. It then asks the clients to rate an initial list of requirements, recommends other relevant requirements to them using collaborative filtering, and prioritizes their requirements using their ratings weighted by their project influence [1].

Step 1. Identify and Prioritize Clients

Step 1 identifies and prioritizes the clients based on their influence in the project. Clients have to be identified, as they are the source of requirements. They have to be prioritized, as their level of influence in the project affects the priority of their requirements. The output is a prioritized list of client roles and, for each role, a prioritized list of clients. StakeRare uses StakeNet for Step 1. StakeNet is a previously published client analysis method that produces such an output. StakeNet identifies an initial set of clients and asks them to recommend other clients and client roles. A recommendation is a triple $\langle \text{Client}, \text{client role}, \text{saliency} \rangle$, Where saliency is a number on an ordinal scale (e.g., 1-5). For example, in a software project to implement a university access control system, Alice, a client representing the role of Estates that manages the university's physical estate, can make a recommendation $\langle \text{Bob}, \text{Library}, 4 \rangle$. StakeNet then asks Bob to recommend other clients. Based on the clients' recommendations, StakeNet builds a social network with clients as nodes and their recommendations as links. Finally, StakeNet applies various social network measures, such as betweenness centrality, degree centrality, and closeness centrality, to prioritize the clients in the network. The social network measures produce a score for each client. The client roles are prioritized by the highest score of their clients.

Step 2. Collect Profile

Step 2 collects a profile from each client identified in Step 1. Existing gathering methods in the background section, such as interviews with a subset of clients or focus groups can be used to identify an initial list of requirements. As mentioned in the background section, requirements can be defined at different levels of abstraction and a high-level

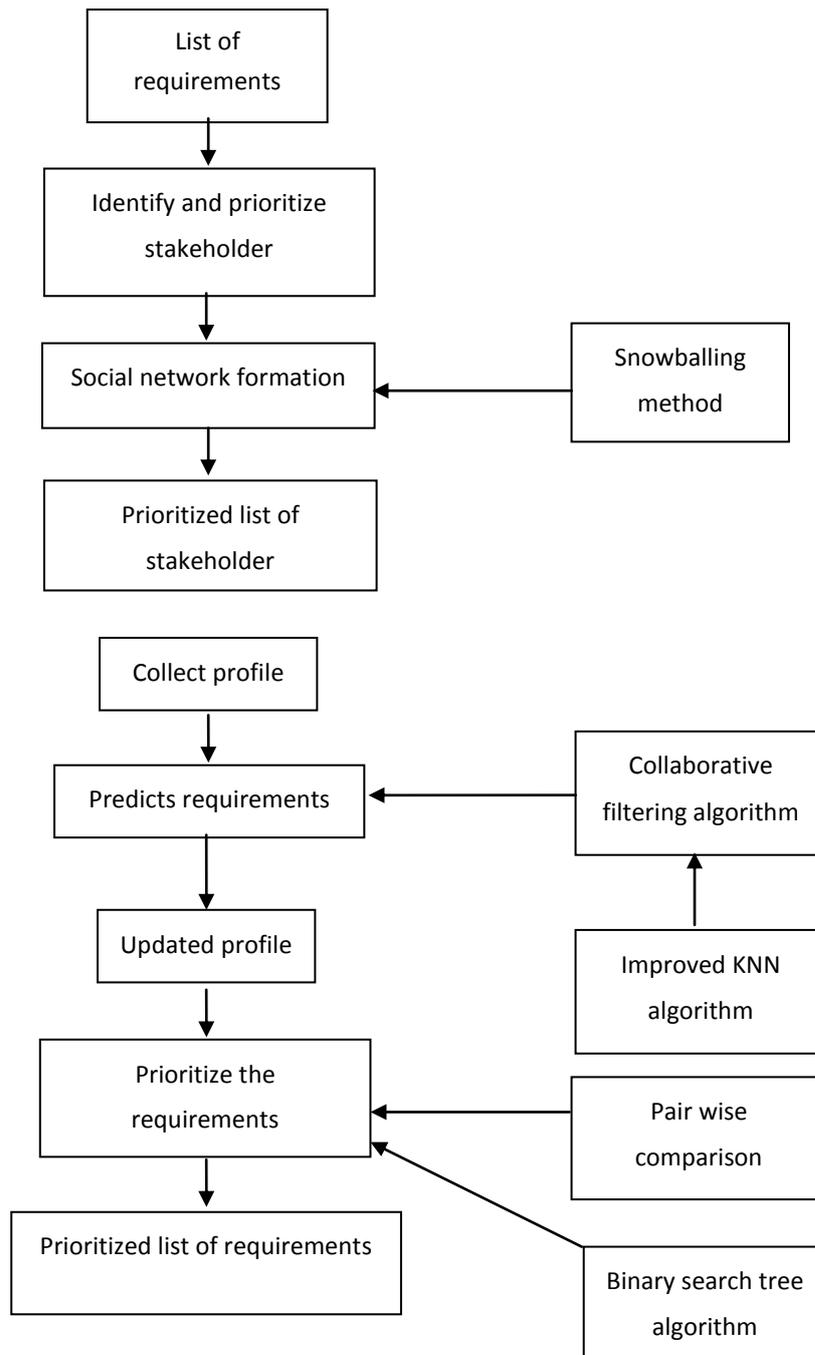
requirement can be refined into several specific requirements. In this example, the requirements are organized into a hierarchy of three levels: project objective, requirement, and specific requirement. Achieving all the specific requirements means that the parent requirement is achieved, and achieving all the parent requirements means that the project objective is achieved. For example, the requirement “all in one card” falls under the project objective “better user experience,” as it is easier to carry one card for all purposes then, combining the various cards are specific requirements under “all in one card.” The clients identified in Step 1 are asked to provide their preferences on the initial requirements. A preference is a triple: <client; requirement; rating>; where rating is a number on an ordinal scale (e.g., 0-5) reflecting the importance of the requirement to the client (e.g., 0 is unimportant and 5 is very important). For example, Alice provides a preference: <Alice; to combine library card with access card; 5>. Clients can also indicate requirements that they actively do not want (e.g., by rating the requirement an X). For example, Bob provides: <Bob; to combine access card with bank card; X>. Clients can also rate requirements not in the list by adding their own requirements. The requirements added are then available to be rated by other clients. If a requirement provided by a client does not have any specific requirements, specific requirements can be identified using existing gathering methods (e.g., interviews) and added to the list to be rated. Finally, StakeRare propagates the ratings of requirements to avoid missing values. Rating propagation enables StakeRare to make prioritizations and predictions at different levels of detail. If a client rates a high-level requirement but does not rate the lower level requirements, then his rating propagates down to the lower level requirements. This propagation assumes that specific requirements when unrated by the client have the same rating as their parent requirement, and the client agrees with the decomposition of the requirement into the specific requirements. Similarly, if a client 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 client cares about a specific requirement, they would care equally about the parent requirement. If more than one specific requirement is rated, then the maximum rating is propagated.

Step 3. Predict Requirements

Based on the clients’ profile, Step 3 uses collaborative filtering to predict other requirements that each client needs or actively does not want. StakeRare uses the k-Nearest Neighbor (kNN) algorithm described in the background section. Cross validation is used to find the optimal value for k. kNN finds similar clients by measuring the similarity between the clients’ profiles. Then, it generates the predicted level of interest that a client will have in a requirement which he has not yet rated. StakeRare returns requirements that may be relevant to the client (i.e., requirements with the highest predicted level of interest) as recommendations at all three levels. Clients can then rate the requirements that are recommended to them, provide new requirements, or rate other requirements. The new ratings by the clients 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 clients after one round of recommendations.

Step 4. Prioritize Requirements

For the final step, StakeRare aggregates all the clients’ profiles into a prioritized list of requirements. The ratings from the clients’ profiles, and the priority of the clients and their roles from Step 1 are used to prioritize requirements. Negative ratings (from a client actively not wanting a requirement) are excluded in the calculation, as their purpose is to highlight conflicts to the requirements engineers, rather than to prioritize the requirements. To calculate the importance of a requirement in a project, the influence of the stake holder’s role in the project is determined and then the influence of the clients in their roles is determined.



System Architecture Diagram

VII. Result

StakeRare identified the requirements in RALIC with a high level of completeness, with a 10 percent higher recall compared to the existing method used in the project. As the existing method mainly involved decision makers, the list omitted process related requirements such as “enable visual checking of cardholders’ roles” and “ease of installing new card readers.” In the StakeRare list, these requirements were identified by clients who were involved in the process. Hence, Stake Rare’s approach of asking clients with different perspectives increased the completeness of the gathered requirements, which is critical to build a system that meets the clients’ needs.

VIII. Conclusion

In macroscopic software projects, requirements’ gathering tends to be beset by three problems: information overload, inadequate client input, and biased prioritization of requirements. The main contribution of the work is the

development of the StakeRare method, which supports requirements gathering in macroscopic software projects. The method is one of filtering to identify [2] and prioritize clients and their requirements. A second important contribution of the work is the extensive empirical evaluation of the methods using a real macroscopic software project. This work pioneered three significant forms of evaluation: the comparison with existing gathering methods used in the project, the comparison with the ground truth built from post project knowledge, and the use of standard statistical measures from the information retrieval literature. This substantial empirical study using real data is one of the first in requirements gathering research. Approximately 200 face-to-face interviews were conducted with the project clients, and more than 1,000 pages of project documentation were reviewed.

IX. Future Work

To address the threats to validity, future work should evaluate StakeRare using different projects in different organizations, and apply the method at the start of the projects. In addition, although RALIC was a macroscopic project in terms of number of clients and client roles, it is not large in terms of the number of gathered requirements. StakeRare aims to predict the requirements the clients may need so that they need not go through a long list of requirements when they rate requirements. Hence, future work should evaluate StakeRare using projects with a large number of requirements. Future evaluations should also consider alternative ways of building the ground truth to increase its objectiveness, such as involving more than one researcher. Future work should improve the objectiveness and scalability of the data cleaning, for example, by crowd sourcing the clients to clean the data, enabling the clients to comment on the requirements engineers' data cleaning, or using natural language processing to identify [2] similar requirements.

References

- [1] A. Finkelstein, Soo Ling Lim, StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation, IEEE, May June 2012, pp: 707-735.
- [2] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," Proc. Conf. Future of Software Eng., pp. 35-46, 2000.
- [3] H.Sharp, G.H. Galal, and A. Finkelstein, "Stakeholder Identification in the Requirements Engineering Process," Proc. Database and Expert System Applications Workshop, pp. 387-391, 1999.
- [4] P. Zave, "Classification of Research Efforts in Requirements Engineering," ACM Computing Surveys, vol. 29, no. 4, pp. 315- 321, 1997
- [5] J.Cleland-Huang and B. Mobasher, "Using Data Mining and Recommender Systems to Scale Up the Requirements Process," Proc. Second Int'l Workshop Ultra-Large-Scale Software-Intensive Systems, pp. 3-6, 2008.
- [6] R.N. Charette, "Why Software Fails," IEEE Spectrum, vol. 42, no. 9, pp. 42-49, Sept. 2005.
- [7] D.C. Gause and G.M. Weinberg, Exploring Requirements: Quality before Design. Dorset House Publishing Company, Inc., 1989.
- [8] I. Alexander and S. Robertson, "Understanding Project Sociology by Modeling Stakeholders," IEEE Software, vol. 21, no. 1, pp. 23-27, Jan./Feb. 2004.
- [9] I. Alexander, "A Taxonomy of Stakeholders: Human Roles in System Development," Int'l J. Technology and Human Interaction, vol. 1, no. 1, pp. 23-59, 2005.
- [10] L. Lehtola, M. Kauppinen, and S. Kujala, "Requirements Prioritization Challenges in Practice," Proc. Int'l Conf. Product Focused Software Process Improvement, pp. 497-508, 2004.
- [11] I. Sommerville and P. Sawyer, Requirements Engineering: A Good Practice Guide. John Wiley & Sons, Inc., 1999