StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation

Soo Ling Lim and Anthony Finkelstein, *Member*, *IEEE*

Abstract—Requirements elicitation is the software engineering activity in which stakeholder needs are understood. It involves identifying and prioritizing requirements—a process difficult to scale to large software projects with many stakeholders. This paper proposes StakeRare, a novel method that uses social networks and collaborative filtering to identify and prioritize requirements in large software projects. StakeRare identifies stakeholders and asks them to recommend other stakeholders and stakeholder roles, builds a social network with stakeholders as nodes and their recommendations as links, and prioritizes stakeholders using a variety of social network measures to determine their project influence. It then asks the stakeholders 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. StakeRare was evaluated by applying it to a software project for a 30,000-user system, and a substantial empirical study of requirements elicitation was conducted. Using the data collected from surveying and interviewing 87 stakeholders, the study demonstrated that StakeRare predicts stakeholder needs accurately and arrives at a more complete and accurately prioritized list of requirements compared to the existing method used in the project, taking only a fraction of the time.

Index Terms—Requirements/specifications, elicitation methods, requirements prioritization, experimentation, human factors, recommender systems, social network analysis, stakeholder analysis.

1 Introduction

Software systems are growing. The increase in size extends beyond mere lines of code (LOC) or number of modules. Today, 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 [1]. 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 [1], [2]. In large projects, these stakeholders cut across divisions and organizations. They have diverse needs which may conflict.

Requirements elicitation is the software engineering activity in which stakeholder needs are understood [1]. It aims to identify the purpose for which the software system is intended [3]. It involves identifying stakeholders and prioritizing them based on their influence in the project. It also involves identifying requirements from these stakeholders and prioritizing their requirements.

StakeRare is a method to identify and prioritize requirements using social networks and collaborative filtering. It aims to address three problems that beset large-scale requirements elicitation: *information overload*, *inadequate stakeholder input*, and *biased prioritization of requirements*.

Information overload is inevitable in big projects. These projects tend to have many stakeholders and requirements. Existing methods for requirements elicitation require intensive interactions with the stakeholders, for example, through face-to-face meetings, interviews, brainstorming sessions, and focus groups [1]. These methods lack the means to manage the information elicited from stakeholders. As such, the methods fail to scale to big projects with hundreds, thousands, or even hundreds of thousands of stakeholders [4]. Practitioners struggle to use these methods in large projects. Inevitably, stakeholders 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 [5].

Inadequate stakeholder input is caused by inadequate stakeholder selection. Omitting stakeholders is one of the most common mistakes in software engineering [6]. Existing stakeholder analysis methods are likely to overlook stakeholders [7]. In addition, stakeholders are often sampled during requirements elicitation [8]. As requirements are elicited from stakeholders, omitting stakeholders results in missing requirements, which in turn leads to the wrong product being built.

Biased prioritization of requirements occurs because current prioritization practices depend on individuals who may not have a global perspective in large projects [4], [9]. Although the literature suggests that prioritizing from multiple stakeholders' viewpoints can reveal important requirements [10], the task is almost impossible to perform with many stakeholders and many requirements. As a result, important requirements known to only a few stakeholders

• *The authors are with the Department of Computer Science, University College London, Gower Street, London WC1E 6BT, United Kingdom. E-mail: {s.lim, a.finkelstein}@cs.ucl.ac.uk.*

Manuscript received 4 Sept. 2010; revised 26 Dec. 2010; accepted 27 Feb. 2011; published online 22 Mar. 2011.
Recommended for acceptance by T. Tamai.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2010-09-0271.
Digital Object Identifier no. 10.1109/TSE.2011.36.

can be lost in the sea of information. Those who attempt to get multiple viewpoints find it difficult to combine information from different sources [9]. Many practitioners avoid prioritizing requirements or resort to rough guesses when they prioritize requirements [9].

Above all, the existing requirements elicitation literature is largely qualitative [1], [11]. Without empirical evaluations using real projects, no one can be certain how well one method performs against another, or indeed whether the methods work at all!

To address these problems, this work proposes a method that uses social networks and collaborative filtering for requirements elicitation. In doing so, the work makes the following contributions:

- The development of StakeRare, a novel method that uses social networks and collaborative filtering to support requirements elicitation in large-scale software projects.
    - StakeRare stands for Stakeholder- and Recommender-assisted method for requirements elicitation. StakeRare supports requirements elicitation in projects where there are many stakeholders who must be heard, but are unable to meet, possibly due to sheer numbers, dispersed locations, or lack of time. It aims to be open and inclusive so that the stakeholders can participate in requirements elicitation.
    - StakeRare uses social networks to identify and prioritize stakeholders and their roles in the project. Then, it asks the stakeholders 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 derived from their position on the social network.
    - StakeRare addresses *information overload* by using collaborative filtering to recommend relevant requirements to stakeholders, and prioritizing the stakeholders and requirements. It addresses *inadequate stakeholder input* by asking stakeholders to recommend other stakeholders, and asking all stakeholders to provide requirements. It addresses *biased prioritization of requirements* by prioritizing requirements using the stakeholders' ratings on the requirements and their position on the social network.
- The evaluation of StakeRare using a real large-scale software project.
    - The evaluation is empirical and appears to be one of the first in requirements elicitation (as reviewed in [12]). It is substantial, using postproject knowledge to establish the ground truth of requirements. It uses measurements from the information retrieval literature, such as precision, recall, and mean absolute error (MAE), to determine the quality of the requirements returned by StakeRare. It also compares StakeRare to the existing methods used in the project.

TABLE 1
Project Size and Measures

Project Size	Measure		
	Lines of Code	Function Points	Number of Developers
Small	< 2,000^	< 100*	< 5†
Large	> 500,000*•#	> 5,000*	> 50#∩
Ultra-large	1,000,000,000~	> 100,000△	> 1,000⊥

Source: *[15], ^[25], †[26], #[23], •[24], ∩[27], ~[32], ⊥[28], △[18].

-
    - The evaluation provides clear evidence that StakeRare can identify a highly complete set of requirements and prioritize them accurately. In addition, it is straightforward to use and requires less time from the requirements engineers and stakeholders compared to the existing methods used in the project.

The rest of the paper is organized as follows: The next section reviews the existing literature. Section 3 describes the StakeRare method and Section 4 evaluates StakeRare. Section 5 identifies the limitations of the study, Section 6 describes future work, and Section 7 concludes.

2 Background

2.1 Large-Scale Software Projects

In this work, the definition of a large-scale software project is derived from the existing measures of project size and definitions of large-scale software projects. As requirements elicitation is the focus of this work, the definition measures the size of the requirements engineering tasks, rather than the size of the software system.

There are a number of existing measures to size a project, leading to different views on what constitutes large scale. Popular measures of project size include lines of code, function points (FP), number of developers, and man-hours [13], [14], [15], [16], [17], [18], [19]. LOC counts the number of nonblank, noncomment lines in the text of a software program's source code [20], [21], [22], [23], [24]. FP determines size by identifying the components of the system as seen by the end user, such as the inputs, outputs, interfaces to other systems, and logical internal files. Number of developers counts the number of developers involved in the project. Man-hour is the amount of work performed by an average worker for an hour [19].

These measures have been used to indicate the relative size of projects (Table 1) [25], [26], [27], [28]. But the numbers to indicate size are not absolute and may vary across different work. For example, McConnell [23] considered small projects to have 2,500 LOC, but Kruchten [29] considered them to have 10,000 LOC. McConnell [24] considered projects with 500,000 LOC as very large, but Kruchten [29] considered projects with 700,000 LOC as large.

These measures are more suitable for development [23], [24] and less so for elicitation. For example, a software project to solve a complicated set of differential equations may be very large in terms of LOC or man-hours, but may only have a small number of stakeholders [30]. Although the project is considered large in terms of development effort, it is small in terms of elicitation effort [30].

In requirements elicitation, the number of stakeholders is often used to size a project. Cleland-Huang and Mobasher define an ultra-large-scale project to have thousands or even hundreds of thousands of stakeholders [4]. Burstin and Ben-Bassat define a large software system as "a software system that has a large and diversified community of users, and entails a variety of human, organizational, and automated activities, and various, sometimes conflicting, aspects of different parts of its environment" [30]. Large, complex projects have multiple stakeholder groups that cut across many different agencies, divisions, and even organizations [31]. According to Northrop et al. [32] and Cheng and Atlee [11], the human interaction element makes requirements elicitation the most difficult activity to scale in software engineering. For example, the FBI Virtual Case File project is widely cited as a large-scale software project in the existing literature [4], [33], [34], [35], [36]. It had 12,400 users (FBI agents who would use the software). The project had more than 50 stakeholder groups, as the FBI consisted of 23 divisions which previously had their own IT budget and systems, and the agents worked out of 56 field offices [37].

This work defines a large-scale software project as a software project with dozens of stakeholder groups and tens of thousands of users, where users are members of the stakeholder groups and a stakeholder group contains one or more stakeholder roles. These stakeholders have differing and sometimes conflicting requirements. This definition measures the size of the project in terms of the requirements engineering task, and is based on the requirements engineering literature discussed in previous paragraphs.

2.2 Requirements Elicitation

2.2.1 Elicitation Techniques

In requirements elicitation, traditional techniques, such as interviews and focus groups, form the basis of existing practice [1], [38], [39], [40]. In interviews, the requirements engineers approach stakeholders with questions to gain information about their needs [41]. Focus groups bring stakeholders 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 catalyzing discussions and exploring the stakeholders' needs. These techniques include prototyping [42], metaphors [43], storyboards [44], [45], and model-driven techniques such as use cases [38], [46], scenarios [47], and goal models [48], [49], [50]. Nevertheless, similarly to traditional techniques, they require face-to-face meetings and hence do not scale well to large projects [4], [3].

2.2.2 Prioritization Techniques

Projects often have more requirements than time, resource, and budget allow for. As such, requirements should be prioritized and managed so that those that are critical and most likely to achieve customer satisfaction can be selected for implementation [36], [51], [52], [53].

A prioritization technique commonly used in practice is the numeral assignment technique [36], [53], [54]. In this technique, each requirement is assigned a value representing its perceived importance. For example, requirements can be classified as mandatory, desirable, or inessential [53]. Numeral assignment is straightforward, but a study by Karlsson [53] found that the participants' opinions about the numbers in the numeral assignment technique differ, and the scoring system is often inconsistent as different people make use of different personal scales. Nevertheless, this technique is widely used due to its simplicity.

Another popular technique is the pairwise comparison approach [53]. 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 [53], [55]. 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. Pairwise comparison is simple. However, since all unique pairs of requirements need to be compared, the effort is substantial when there are many requirements [55]. Prioritizing n requirements needs $n \times (n-1)/2$ comparisons [55], [56]. Hence, a project with 100 requirements would require 4,950 comparisons.

Many existing approaches, including those previously mentioned, prioritize requirements from an individual's perspective. Other similar approaches include the cost-value approach, which prioritizes requirements based on their relative value and implementation cost [53], [55], [56], and the value-oriented prioritization method, which prioritizes requirements based on their contribution to the core business values and their perceived risks [57]. As prioritizations involve a small subset of stakeholders, the results are biased toward the perspective of those involved in the process [4].

More sophisticated methods combine prioritizations from multiple stakeholders. In the 100-point test, each stakeholder is given 100 points that they can distribute as they desire among the requirements [58]. Requirements that are more important to a stakeholder are given more points. Requirements are then prioritized 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 [36], [59]. For example, stakeholders may distribute their points based on how they think others will do it [60]. In addition, it is difficult for stakeholders to keep an overview of a large number of requirements [54].

In the requirements triage method, Davis [61] proposed that stakeholders should be gathered in one location and group voting mechanisms be used to prioritize 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 that the relative priorities of requirements depend on the stakeholders who attended the prioritization meeting, and dominant participants may influence the prioritization [36].

In the win-win approach proposed by Boehm, stakeholders negotiate to resolve disagreements about candidate requirements [62], [63]. 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 toward achieving mutual gain, and use objective criteria to prioritize requirements. Nevertheless, the approach is labor intensive, particularly in large projects [59].

Another method that involves multiple stakeholders is the value, cost, and risk method proposed by Wiegers [64]. 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 favorable balance of the three elements. This method is limited by the individual's ability to determine the value, cost, and risk for each requirement [64].

Many existing prioritization methods consider requirements to have a flat structure and be independent of one another [65]. However, requirements are often defined at different levels of abstraction. For example, a high-level requirement can be refined into several specific requirements [48], [66]. Hierarchical cumulative voting (HCV), proposed by Berander and Jönsson [54], enables prioritizations to be performed at different levels of a hierarchy. Stakeholders perform prioritization using 100-point test within each prioritization 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 normalization. If several stakeholders have prioritized the requirements, their individual results are then weighted and combined. When doing so, different stakeholders may have different weights. Although the hierarchical prioritization in HCV makes it easier for the stakeholders to keep an overview of all the requirements, the prioritizations need to be interpreted in a rational way as stakeholders can easily play around with the numbers [54].

There is a plethora of methods to prioritize requirements, such as multi-attribute utility theory [67], top 10 requirements [41], outranking [68], minimal spanning tree [55], cost benefit analysis [69], and Quality Function Deployment [70]. Many of these methods have similar shortcomings: Significant effort is required when there are many requirements and the requirements' priorities are easy to manipulate [60]. For example, the Quality Function Deployment suggests a limit of 30 requirements [71]. Cost benefit analysis relies on the type of costs included in the analysis by the decision makers, which may be biased due to their vested interest [69].

One of the few methods that can scale to a large number of requirements is the binary search tree (BST) [72]. 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 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. This method is simple to implement but provides only a simple ranking of requirements as no priority values are assigned to the requirements [36].

For projects with many requirements, recent works by Laurent et al. [34] and Duan et al. [36] propose Pirogov, which uses data mining and machine learning techniques to support requirements prioritization. Pirogov uses various clustering techniques to organize requirements into different categories. The requirements engineers then prioritize the clusters and determine the importance of each clustering technique. Using the information, Pirogov generates a list of prioritized requirements. By automatically clustering the requirements into different categories, Pirogov reduces the number of manual prioritizations. It is a significant step toward large-scale requirements elicitation. But, at the moment, the results of prioritization depend on the requirements engineers' subjective prioritization of the clusters and clustering techniques [36].

2.3 Social Network Analysis

Social network analysis is the application of methods to understanding the relationships among actors and on the patterns and implications of the relationships [73]. 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 [73]. These actors are linked to one another by relational or social *ties*, such as evaluation of one person by another (e.g., friendship or respect), transfers of material resources (e.g., business transaction), and formal relations (e.g., authority) [73].

In social network analysis, the snowballing method proposed by Goodman [74] is used to sample social network data for large networks where the boundary is unknown [73], [75]. It is also used to track down "special" or "hidden" populations, such as business contact networks, community elites, and deviant subcultures [76]. Snowball sampling begins with a set of actors [73], [75], [76]. 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 down a hill [75]. 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 set under study [76].

A social network is a structure that consists of actors and the relation(s) defined on them [73], [75]. 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 [73], [75]. 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 visualized.

The centrality of actors in their social networks is of great interest to social network analysts [75]. Actors that are more central have a more favorable position in the network [76]. 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 [73], [75], [76].

In requirements engineering, Damian et al. used social network analysis to study collaboration, communication, and awareness among project team members [77], [78]. 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 analyze the collaboration behavior. For example, degree centrality indicated active members and betweenness centrality indicated members who control interactions between other members.

2.4 Collaborative Filtering

Collaborative filtering is a technique to filter large sets of data for information and patterns [79]. This technique is used in recommender systems to forecast a user's preference on an item by collecting preference information from many users [80]. For example, Amazon[1] uses collaborative filtering to recommend books to their customers and MovieLens[2] uses it to recommend movies [80], [81]. The underlying assumption is that users who have had similar taste in the past will share similar taste in the future [82].

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 [83]. A *rating* is a numerical 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 is then presented to the user as recommendations [82].

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 [80], [84], [85]. kNN is used to identify like-minded users with similar rating histories in order to predict ratings for unobserved users-item pairs [85]. 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 [80]. 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 [80].

In requirements engineering, Castro-Herrera et al. use collaborative filtering to facilitate online discussions for requirements identification [86], [87]. Their method, named Organizer 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 encourage stakeholders 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 paper 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 prioritize, while still ensuring they are aware of the requirements they may be interested in.

2.5 Summary

In this paper, a large-scale software project is defined as a software project with dozens of stakeholder groups and tens of thousands of users, where users are members of the stakeholder groups. These stakeholders have differing and sometimes conflicting requirements.

Existing methods to identify and prioritize requirements do not scale well to large projects. Most elicitation methods require face-to-face meetings with the stakeholders and hence are time consuming when there are many stakeholders. Existing requirements prioritization methods require substantial efforts from the requirements engineers when there are many requirements. Furthermore, requirements prioritization from an individual's perspective is likely to be biased, especially in large projects where no individual has the global perspective.

An ideal method in requirements elicitation should identify and prioritize stakeholders and their requirements from a global perspective. It should be independent of the individual doing the analysis and scalable to large projects. In doing so, it should not overload stakeholders with information or burden the requirements engineers. The aim of this work, described in the next section, is to develop such a method using the existing techniques in social networks and collaborative filtering described in this section.

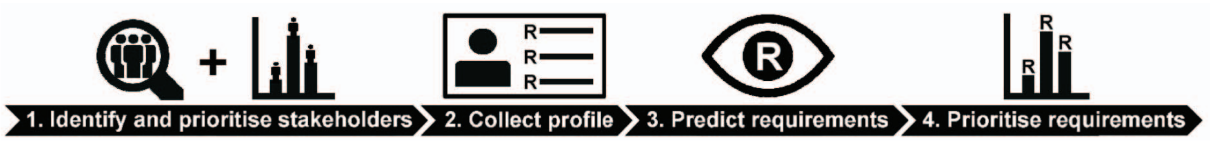
3 StakeRare

Large projects tend to be beset by three problems: information overload, inadequate stakeholder input, and biased prioritization of requirements. StakeRare is a method that uses social networks and collaborative filtering to elicit requirements in large projects.

To address the problem of inadequate stakeholder input, StakeRare aims to be open and inclusive so that a representative sample of stakeholders participates in the requirements elicitation process. As stakeholders are socially related to one another, they can be identified and prioritized using their relations. StakeRare exploits previous work [89] to do this. The previous work asks stakeholders to recommend other stakeholders, builds a social network with stakeholders as nodes and their recommendations as links, and prioritizes stakeholders from a global perspective using social network measures [89].

To avoid overloading stakeholders with information, StakeRare uses collaborative filtering to present only the

1. http://www.amazon.com/.
2. http://www.movielens.org/.


1. Identify and prioritise stakeholders
2. Collect profile
3. Predict requirements
4. Prioritise requirements

Step 1	Step 2	Step 3	Step 4
Input: None **Substeps**: 1. Requirements engineer provides initial stakeholders. 2. Initial and newly identified stakeholders provide recommendations. 3. StakeRare builds social network. 4. StakeRare prioritises stakeholders and roles using social network measures. **Output**: Prioritised list of stakeholders and roles.	**Input**: Stakeholder list from Step 1. **Substeps**: 1. Requirements engineer provides initial requirements. 2. Stakeholders in stakeholder list rate requirements and provide other requirements. 3. StakeRare propagates ratings. **Output**: Profiles of stakeholders who responded.	**Input**: Profiles from Step 2. **Substeps**: 1. StakeRare applies collaborative filtering algorithms on profiles to predict unrated requirements. 2. StakeRare recommends requirements that may be relevant to the stakeholders. 3. Stakeholders rate recommended requirements. Substeps 1-3 can be repeated using updated profiles. **Output**: Updated profiles.	**Input**: Stakeholder list from Step 1, and profiles from Step 3. **Substeps**: 1. StakeRare calculates project influence for each stakeholder. 2. StakeRare calculates score for each requirement. 3. StakeRare prioritises requirements based on score. **Output**: Prioritised list of requirements.

Fig. 1. StakeRare steps.

requirements that are relevant to them. StakeRare asks each stakeholder to rate an initial list of requirements and, based on the list, identifies a neighborhood of similar stakeholders for each stakeholder. Then, it predicts other relevant requirements for the stakeholder based on the requirements provided 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, StakeRare prioritizes stakeholders and their requirements.

Finally, to avoid biased prioritization of requirements, StakeRare produces a prioritized 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.

StakeRare has four steps (Fig. 1) and uses the concepts in Table 2.

3.1 Step 1. Identify and Prioritize Stakeholders

Step 1 identifies and prioritizes the stakeholders based on their influence in the project (Fig. 1). Stakeholders 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 stakeholder roles and, for each role, a prioritized list of stakeholders.

StakeRare uses StakeNet for Step 1. StakeNet is a previously published stakeholder analysis method that produces such an output [89]. StakeNet identifies an initial set of stakeholders and asks them to recommend other stakeholders and stakeholder roles. A recommendation is a triple <stakeholder, stakeholder role, salience>, where salience 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 stakeholder representing the role of Estates that manages the university's physical estate, can make a recommendation <Bob, Library, 4>. StakeNet then asks Bob to recommend other stakeholders.

Based on the stakeholders' recommendations, StakeNet builds a social network with stakeholders as nodes and their recommendations as links [89]. An example stakeholder network is illustrated in Fig. 2.

Finally, StakeNet applies various social network measures, such as betweenness centrality, degree centrality, and closeness centrality, to prioritize the stakeholders in the network [89]. The social network measures produce a score for each stakeholder. The stakeholder roles are prioritized by the highest score of their stakeholders. An example output is illustrated in Table 3. Fractional ranking or "1 2.5 2.5 4" ranking [90] 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.

3.2 Step 2. Collect Profile

Step 2 collects a profile from each stakeholder identified in Step 1 (Fig. 1). 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

TABLE 2
StakeRare Concepts

Concept	Definition
Salience	The level of influence a stakeholder has on the project [88]. Stakeholders with high salience are crucial to project success; stakeholders with low salience have marginal impact.
Scope	The work required for completing the project successfully [39].
Stakeholder	An individual or a group who can influence or be influenced by the success or failure of a project [1].
Stakeholder role	The stakeholder's position or customary function in the project [2].
Requirement	The real-world goals for, functions of, and constraints on software systems [3].
Rating	Numerical importance of a requirement to the stakeholder [80].
Profile	The set of requirements and their ratings provided by a stakeholder [80].

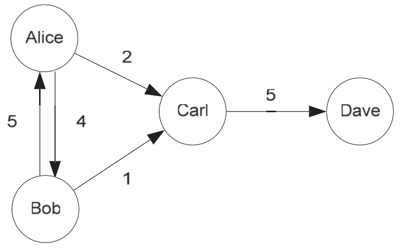


Fig. 2. Example stakeholder network.

control project as an example, an interview with Alice from Estates revealed that one of the project objectives is to provide "better user experience." Bob representing the library reveals that his requirement is "to combine library card with access card," student Dave's requirement is "to combine access card with bank card," and Alice, representing the Estates, requests "all in one card."

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 [48], [66]. In this example, the requirements are organized into a hierarchy of three levels: project objective, requirement, and specific requirement.[3] 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 (Fig. 3). Then, combining the various cards are specific requirements under "all in one card."

The stakeholders identified in Step 1 are asked to provide their preferences on the initial requirements. A preference is a triple:

$$<stakeholder, requirement, rating>,$$

where rating is a number on an ordinal scale (e.g., 0-5) reflecting the importance of the requirement to the stakeholder (e.g., 0 is unimportant and 5 is very important). For example, Alice provides a preference:

$$<\text{Alice, To combine library card with access card, } 5>.$$

Stakeholders can also indicate requirements that they actively do not want (e.g., by rating the requirement an X). For example, Bob provides:

$$<\text{Bob, To combine access card with bank card, } X>.$$

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.

3. Project objectives describe specific and measurable goals for the project, requirements describe what must be delivered to meet the project objectives, and specific requirements describe what must be delivered to meet the requirements. The number of levels and their classification are determined by the requirements engineer, and may be modified during the elicitation process (e.g., stakeholders may provide more details to a specific requirement, resulting in an additional level to the hierarchy).

TABLE 3
Example Prioritized List of Stakeholders

Prioritised Stakeholder Roles	Prioritised Stakeholders
(1) Estates	Alice
(2) Students	(1) Dave
	(2) Carl
(3) Library	Bob

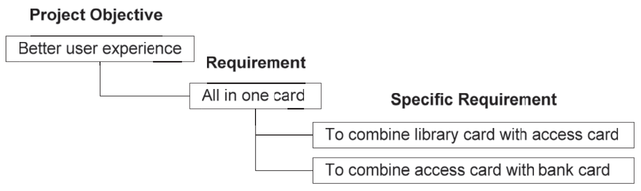


Fig. 3. The hierarchy of requirements. (A requirement such as "to combine library card with access card, unless access card is also bank card" should be placed under "to combine library card with access card" as it is more specific.)

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 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, Carl provides a preference:

$$<\text{Carl, All in one card}, 4>.$$

Since Bob and Dave provided specific requirements for this requirement, Carl then implicitly provides two other preferences:

$$<\text{Carl, To combine library card and access card}, 4>, \text{ and}$$
$$<\text{Carl, To combine access card with bank card}, 4>.$$

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 (Table 4 A1). 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 4 A2). If more than one specific requirement is rated, then the maximum rating is propagated.

3.3 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). 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 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 which he has not yet rated. StakeRare returns requirements that may be relevant to the stakeholder (i.e., requirements with the

TABLE 4
StakeRare Assumptions

ID	Assumption
A1	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. (This assumption does not always hold. For example, the specific requirements may be in conflict, the stakeholder may have unequal preference among the specific requirements, and the rating for specific requirements may be higher than the requirements.)
A2	If a stakeholder cares about a specific requirement, they would care equally about the parent requirement. (This assumption does not always hold. For example, a stakeholder who actively does not want a specific requirement may still rate the higher-level requirement positively.)
A3	A stakeholder's project influence is determined by their role in the project.
A4	The stakeholder represents only the role that gives him the highest weight.

highest predicted level of interest) as recommendations at all three levels (e.g., Fig. 4).

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.

3.4 Step 4. Prioritize Requirements

For the final step, StakeRare aggregates all the stakeholders' profiles into a prioritized 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 prioritize requirements. Negative ratings (from a stakeholder 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 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 (3-1):

$$Influence_{role(i)} = \frac{RRmax + 1 - rank(role(i))}{\sum_{j=1}^{n}(RRmax + 1 - rank(role(j)))}, \quad (3\text{-}1)$$

where $role(i)$ is stakeholder i's role in the project, $RRmax$ is the maximum rank of the roles in the list, $rank(role(j))$ is

StakeRare recommends the following requirements to you:
1. Card to have features to prevent sharing
2. To combine access card with fitness centre card

The recommendations are based on the requirements you have rated:
★★★★★ To combine library card with access card
★★★★ To combine ID card with session card

Fig. 4. StakeRare's output for Alice at the specific requirements level.

the fractional rank of role j, and n is the total number of roles in the list. Roles where none of the stakeholders provide ratings are excluded. As lower rank values correspond to higher influence, this calculation inverts the rank value by subtracting it from the upper bound of $maxrank_{Role} + 1$. The calculation also normalizes the influence of a role by dividing it with the sum of all role influences. An example prioritized list of stakeholder roles is Estates: Rank 1.5, Students: Rank 1.5, and Library: Rank 3 (fractional ranking is used where Estates and Students have the same rank). Using this example, Estates' influence is $\frac{(3+1)-1.5}{2.5+2.5+1} = 0.42$, Students' influence is the same as Estates' influence, and Library's influence is $\frac{(3+1)-3}{2.5+2.5+1} = 0.17$.

The influence of stakeholder i in the role is calculated the same way, using (3-2):

$$Influence_i = \frac{RSmax + 1 - rank(i)}{\sum_{k=1}^{n}(RSmax + 1 - rank(k))}, \quad (3\text{-}2)$$

where $RSmax$ is the maximum rank of all stakeholders with the same role, $rank(i)$ is the fractional rank of stakeholder i, and n is the total number of stakeholders with the same role. Stakeholders who do not provide any ratings are excluded. Again, as lower rank values correspond to higher influence, this calculation inverts the rank value by subtracting it from the upper bound of $maxrank_s + 1$, then it normalizes the influence by dividing it with the sum of all the influences of stakeholders with the same role. For roles with one stakeholder, the stakeholder's influence is its role's influence. For example, Alice's influence is 1 as she is the only stakeholder for the Estates role. The Student role has two stakeholders, Dave and Carl. Dave's influence is $\frac{(2+1)-1}{2+1} = 0.67$ and Carl's influence in his role is $\frac{(2+1)-2}{2+1} = 0.33$.

The influence of stakeholder i in a project is calculated using (3-3) as follows:

$$ProjectInfluence_i = Influence_{role(i)} \times Influence_i, \quad (3\text{-}3)$$

where $Influence_{role(i)}$ is the influence of the stakeholder's role in the project (3-1), and $Influence_i$ is the influence of the stakeholder in the role (3-2). The sum of all the stakeholders' project influence is equal to 1. From the previous example, Carl's influence in the Student role is 0.33 and the Student role's influence in the project is 0.42. Hence, Carl's influence in the project is $0.33 \times 0.42 = 0.1386$. This calculation of project influence assumes that a stakeholder's project influence is determined by their role in the project (Table 4 A3).

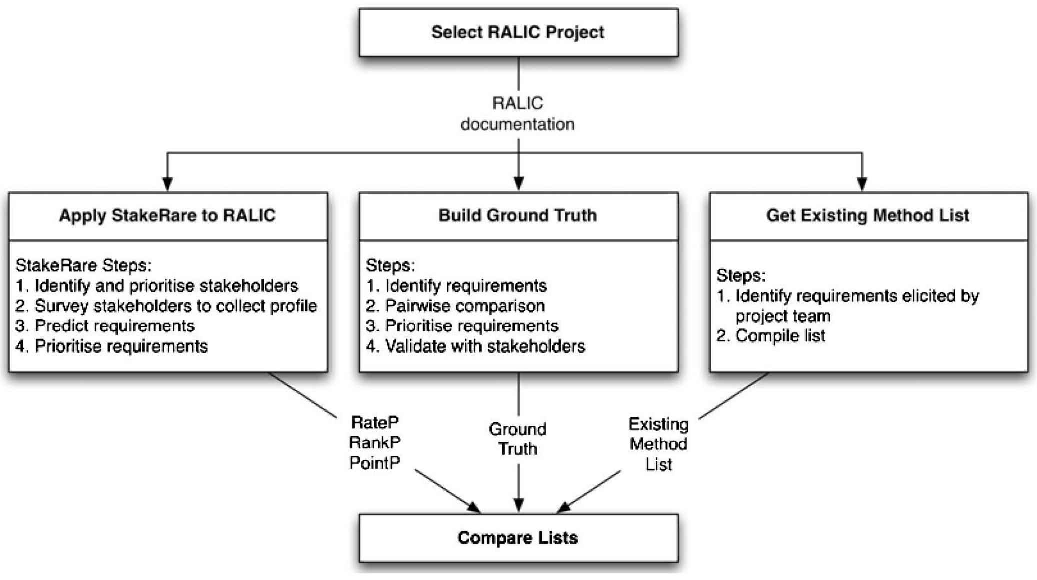


Fig. 5. StakeRare evaluation.

The importance of a requirement is calculated using (3-4) as follows:

$$Importance_R = \sum_{i=1}^{n} ProjectInfluence_i \times r_i, \tag{3-4}$$

where $ProjectInfluence_i$ is the stakeholder i's influence in the project (3-3), r_i is the rating provided by stakeholder i on requirement R, and n is the total number of stakeholders who rated on requirement R. Following the previous example, the requirement "To combine library card with access card" is rated 5 by Alice and 4 by Carl. Alice's influence in the project is 0.42, and Carl's influence in the project is 0.1386; hence, the requirement's importance is $(0.42 \times 5) + (0.1386 \times 4) = 2.6544$. If a stakeholder has more than one role, only the position in the role that gives him the highest weight is considered. By doing so, StakeRare assumes that the stakeholder represents only the role that gives him the highest weight (Table 4 A4).

Finally, the requirements are prioritized based on their importance, where requirements with higher importance values are ranked higher. The requirements are prioritized within their hierarchy so that the output is a ranked list of project objectives, for each project objective, a ranked list of requirements, and, for each requirement, a ranked list of specific requirements. This list is StakeRare's output for the requirements engineers.

4 Evaluation

StakeRare was evaluated by applying it to a real-world large-scale software project. The stakeholders were surveyed for their requirements. The resulting lists of requirements were empirically evaluated in terms of the quality of the requirements and accuracy of the prioritization, by comparing them with the ground truth—the actual complete and prioritized lists of requirements in the project. StakeRare was also compared to the existing methods used in the project in terms of quality of the requirements, accuracy of the prioritization, and the time spent using the methods. Finally, the stakeholders were interviewed and surveyed on the level of difficulty and effort in using StakeRare.

The rest of this section is organized as follows: Section 4.1 details the research questions. Section 4.2 describes RALIC, the large-scale software project used to evaluate this work. Section 4.3 describes the application of StakeRare to RALIC. Section 4.4 describes the construction of the ground truth list of requirements for RALIC and its validation. Section 4.5 describes the existing method list of requirements. Finally, Section 4.6 reveals the results. Fig. 5 summarizes the evaluation described in this section.

4.1 Research Questions

Step 1 of StakeRare identifies and prioritizes the stakeholders and their roles for RALIC. The evaluation of this step has been reported in [89] and [12]. The evaluation shows that social networks can be used to effectively identify and prioritize stakeholders. The method identified a highly complete set of stakeholders and prioritized them accurately, using less time compared to the existing method used in the project. The rest of this section describes the evaluation of the other StakeRare steps.

For Steps 2 and 4, the requirements lists produced by StakeRare were compared with the existing method and the ground truth lists of requirements.

RQ1 to RQ4 as follows are the research questions for Steps 2 and 4.

RQ1. Identifying requirements. The existing requirements elicitation methods described in the background section involve a subset of stakeholders. In contrast, StakeRare involves all the identified stakeholders. This research question assesses how well StakeRare identifies requirements as compared to the existing method used in the project by asking:

- How many requirements identified by StakeRare and the existing method used in the project are actual requirements as compared to the ground truth?
- How many of all the actual requirements in the ground truth are identified by StakeRare and the existing method used in the project?

RQ2. Prioritizing requirements. StakeRare prioritizes stakeholders using social network measures, and then uses the output to prioritize requirements. This research question asks:

- How accurately does StakeRare prioritize requirements as compared to the ground truth?

RQ3. Survey response and time spent. The quality of the requirements returned by StakeRare depends on the stakeholders' motivation to participate. Also, to provide effective support in requirements elicitation, StakeRare should take less time than existing methods. This research question asks:

- Are stakeholders motivated to provide requirements for StakeRare?
- How much time did stakeholders spend in identifying and prioritizing requirements as compared to the existing method in the project?

RQ4. Effective support for requirements elicitation. StakeRare aims to provide effective support for requirements elicitation by providing a predefined list of requirements for the stakeholders to rate (RateP). During the survey, two other elicitation methods were administered to explore the effectiveness of different methods. RankP asks stakeholders to enter their requirements without providing an initial list of requirements, and PointP asks stakeholders to allocate 100 points to the requirements they want in the same predefined list. In RateP and PointP, stakeholders can suggest additional requirements. This research question explores what kinds of support are effective for the requirements engineer and stakeholders by asking the following questions:

- Between the three elicitation methods RankP, RateP, and PointP, which produces the most complete list of requirements and most accurate prioritization for the requirements engineer? Are the results consistent regardless of the elicitation method used?
- Between the three elicitation methods RankP, RateP, and PointP, which do the stakeholders prefer?
- If stakeholders are provided with a list of all the requirements in the project, how prepared are they to rate them all?

In Step 3, collaborative filtering is used to predict other requirements a stakeholder 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 [80], [82], [91]. The evaluation partitioned the stakeholders' 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 stakeholders.

An alternative method to evaluate Step 3 was to make predictions based on the stakeholders' complete profiles and approach the stakeholders to approve the recommended requirements by getting them to rate the recommended requirements. This option was not selected as not all stakeholders were available to be interviewed more than once.

RQ5 to RQ7 as follows are the research questions for Step 3.

RQ5. Predicting requirements. To recommend requirements that may be of interest to the stakeholders, StakeRare uses the *k*NN algorithm in collaborative filtering to identify similar stakeholders and predict their requirements. This research question asks:

- How accurately can collaborative filtering predict stakeholder requirements?
- Are the results consistent regardless of the elicitation method used?

RQ6. Predicting requirements: enhanced profiles. In Castro-Herrera et al.'s work in recommending forums to stakeholders, the stakeholders' profiles are enhanced with stakeholder information, such as their roles in the project and their interest in different aspects of the system, to produce more accurate predictions of the stakeholders' interest in forums [86]. This research question asks:

- Does enhancing stakeholder profile by adding stakeholder information improve the accuracy of predicting stakeholder interest in requirements?
- Are the results consistent regardless of the elicitation method used?

RQ7. Predicting requirements: other algorithms. As mentioned in the background section, *k*NN is a simple machine learning algorithm. Other machine learning algorithms can also be used to predict stakeholders' interest [92]. This research question asks:

- Does using other algorithms and combinations of algorithms improve the prediction accuracy?
- Are the results consistent regardless of the elicitation method used?

4.2 The RALIC Project

The RALIC project was a software project at the University College London (UCL), initiated to replace the existing access control systems at UCL and consolidate the new system with library access and borrowing. RALIC stands for Replacement Access, Library and ID Card. It was a combination of development and customization of an off-the-shelf system. The project duration was two and a half years and the system has been in deployment for over two years.

RALIC was selected to evaluate this work from a list of approximately 40 software projects in UCL. The selection criteria were as follows:

- **Large scale.** The software project must be a large-scale software project following the definition of large scale provided in the background section.
- **Well documented.** The project must be very well documented in order to build the ground truth and existing method lists of requirements to evaluate the work.
- **Available stakeholders.** The stakeholders should be available for interviews.
- **Completed and deployed.** The project should be completed and the system should have been deployed at UCL for more than a year. This is necessary to allow sufficient time for missing

stakeholders and requirements to surface in order to build the ground truth of requirements.

- Requirements elicitation and analysis activities at the start of the project often produce a "complete enough" set of requirements [1]. Stakeholders and requirements that are omitted during the requirements phase are uncovered in later phases, such as design, development, and deployment. For example, one study described a project where all the change requests received during the first year the software was deployed were from stakeholder needs that were overlooked during the project [38].
- Ideally, for the least biased evaluation, the proposed method should be applied to a project when it was initiated and evaluated after the system is deployed so that postproject knowledge does not influence the results. But it is impractical to do so because big projects often take longer than the three-year duration allocated for the work.[4] Also, studies suggest that software projects are more likely to fail than complete successfully [94], [95]; hence, evaluating StakeRare using a project that has just started is risky because the project may fail before the ground truth can be built. Evaluation using a completed project comes with threats to validity, which will be discussed in Section 5.

Most of the projects in the list were either not large scale or lack documentation. RALIC was selected because it met the selection criteria.

- **Large scale.** RALIC had a large and complex stakeholder base with more than 60 stakeholder groups. Approximately 30,000 students, staff, and visitors use the system to enter buildings, borrow library resources, use the fitness center, and gain IT access. Besides all UCL faculties and academic departments, RALIC also involved other supporting departments such as the Estates and Facilities Division that manages UCL's physical estate, Human Resource Division that manages staff information, Information Services Division, Library Services, Security Services, and so on. These stakeholders have differing and sometimes conflicting requirements.
- **Well documented.** RALIC was a very well-documented project and the project documentation was available to build the ground truth and existing method lists of requirements.
- **Available stakeholders.** The majority of RALIC stakeholders were available for interviews. For stakeholders who were unavailable, other staff members were available to take their roles.
- **Completed and deployed.** RALIC was completed and the system was already deployed at UCL for more than a year.

4. A study of 214 software projects in Europe found the average project duration to be over two years [93]. The study investigated projects of all sizes, and the duration for large-scale projects is likely to be above the average.

4.3 Applying StakeRare to RALIC

Using Step 1 of StakeRare, the stakeholders and roles for the RALIC project were identified and prioritized. The application of this step on RALIC is described in previous work [12], [89]. A total of 127 stakeholders and 70 roles were identified.[5] This list of stakeholders and their roles served as input to Step 2 of StakeRare to collect stakeholder's profile.

Step 2 of StakeRare uses existing elicitation methods to identify an initial list of requirements. To reflect the actual initial list of requirements in the project, the initial list of requirements was taken from the earliest draft requirements produced by the project team. This initial list consists of three project objectives, 12 requirements, and 11 specific requirements.

Once the initial list of requirements was prepared, a survey was conducted to collect the profiles of RALIC stakeholders. To do so, all the stakeholders identified in Step 1 were contacted separately via e-mail for a survey.

Each survey took 30 minutes on average. In general, each stakeholder spent 10 minutes to learn about StakeRare and RALIC, and 20 minutes to complete the questionnaire. At the start of each survey, the respondent was provided with a cover sheet describing the survey purpose to elicit RALIC requirements. Then, StakeRare was introduced using a set of slides. After that, the respondent was provided with a description of RALIC and its project scope to familiarize the respondent with the project. The respondents were also asked to put themselves in the situation before RALIC was initiated when providing requirements. To prompt the respondent for recommendations, the respondent was provided with the definition of requirements, as well as the different types of requirements, examples for each type of requirement, and a template to guide the free text provided by the respondent (Fig. 6).

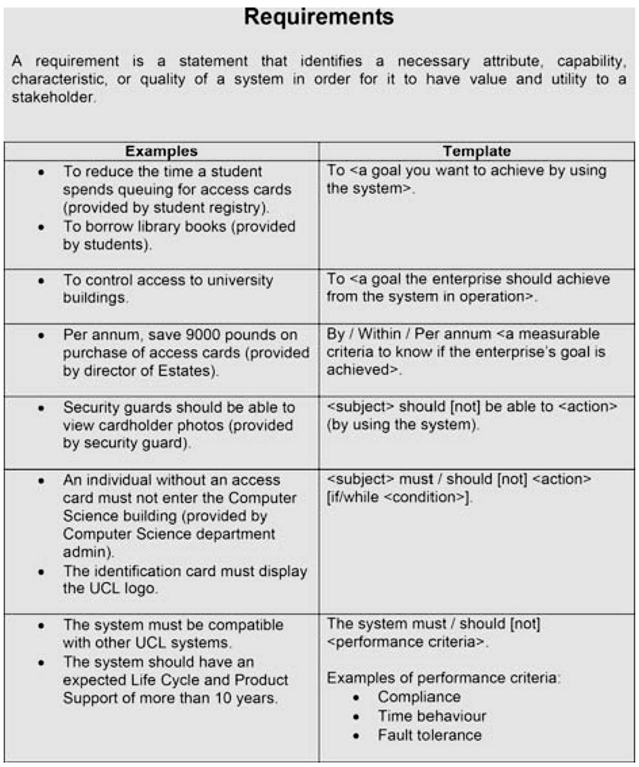
Step 2 of StakeRare collects stakeholders' profiles by asking them to rate a predefined list of requirements (the initial requirements) and provide other requirements not in the list. In addition to this elicitation method, the work also tested two other methods: 1) Without a predefined list, stakeholders provide a list of requirements and assign numeric ranks to the requirements based on their perceived importance [53], and 2) a 100-point test, where each stakeholder is given 100 points that they can distribute as they desire among the requirements [65].

In order to gather these different forms of information, a questionnaire comprised of the following parts[6] was used to gather the stakeholders' profile.

1. **Stakeholder details.** Respondents provide their name, position, department, and role in the project (Fig. 7a).
2. **Ranked profile (RankP).** Respondents provide their requirements with numeric priorities (1 for most important) and X for requirements they actively do not want (Fig. 7b). Then, respondents provide feedback on the elicitation method in terms of three criteria: 1) level of difficulty, 2) effort required, and

5. The 127 stakeholders and 70 roles were identified by applying StakeNet on RALIC. Refer to [12] and [89] for the precision and recall of this list. The 127 stakeholders include all kinds of stakeholders such as users, legislators, decision makers, and developers.

6. The complete questionnaire is available in [12] and at http://www.cs.ucl.ac.uk/research/StakeRare/.



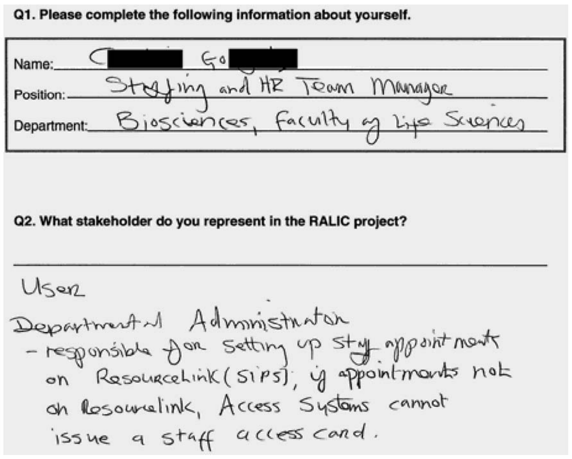
Requirements

A requirement is a statement that identifies a necessary attribute, capability, characteristic, or quality of a system in order for it to have value and utility to a stakeholder.

Examples	Template
• To reduce the time a student spends queuing for access cards (provided by student registry). • To borrow library books (provided by students).	To <a goal you want to achieve by using the system>.
• To control access to university buildings.	To <a goal the enterprise should achieve from the system in operation>.
• Per annum, save 9000 pounds on purchase of access cards (provided by director of Estates).	By / Within / Per annum <a measurable criteria to know if the enterprise's goal is achieved>.
• Security guards should be able to view cardholder photos (provided by security guard).	<subject> should [not] be able to <action> (by using the system).
• An individual without an access card must not enter the Computer Science building (provided by Computer Science department admin). • The identification card must display the UCL logo.	<subject> must / should [not] <action> [if/while <condition>].
• The system must be compatible with other UCL systems. • The system should have an expected Life Cycle and Product Support of more than 10 years.	The system must / should [not] <performance criteria>. Examples of performance criteria: • Compliance • Time behaviour • Fault tolerance

Fig. 6. Requirements: Examples and template.

(a)



Q1. Please complete the following information about yourself.

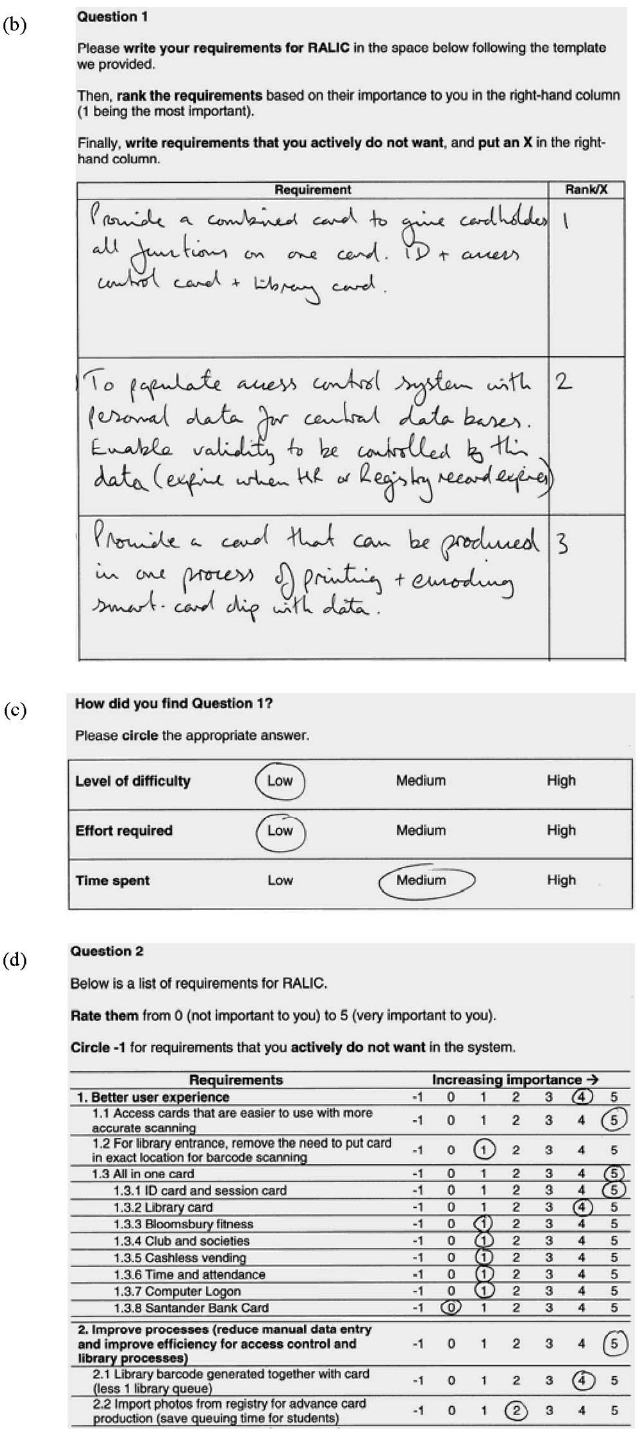
Name: C[redacted] Go[redacted]
Position: Staffing and HR Team Manager
Department: Biosciences, Faculty of Life Sciences

Q2. What stakeholder do you represent in the RALIC project?

User
Departmental Administrator
- responsible for setting up staff appointments on ResourceLink (SIPS); if appointments not on Resourcelink, Access Systems cannot issue a staff access card.

Fig. 7. StakeRare survey questionnaire.

(b)



Question 1

Please **write your requirements for RALIC** in the space below following the template we provided.

Then, **rank the requirements** based on their importance to you in the right-hand column (1 being the most important).

Finally, **write requirements that you actively do not want**, and **put an X** in the right-hand column.

Requirement	Rank/X
Provide a combined card to give cardholder all functions on one card. ID + access control card + library card.	1
To populate access control system with personal data for central data bases. Enable validity to be controlled by this data (expire when HR or Registry record expires)	2
Provide a card that can be produced in one process of printing + encoding smart-card chip with data.	3

(c)

How did you find Question 1?

Please **circle** the appropriate answer.

Level of difficulty	[✓ Low]	Medium	High
Effort required	[✓ Low]	Medium	High
Time spent	Low	[✓ Medium]	High

(d)

Question 2

Below is a list of requirements for RALIC.

Rate them from 0 (not important to you) to 5 (very important to you).

Circle -1 for requirements that you **actively do not want** in the system.

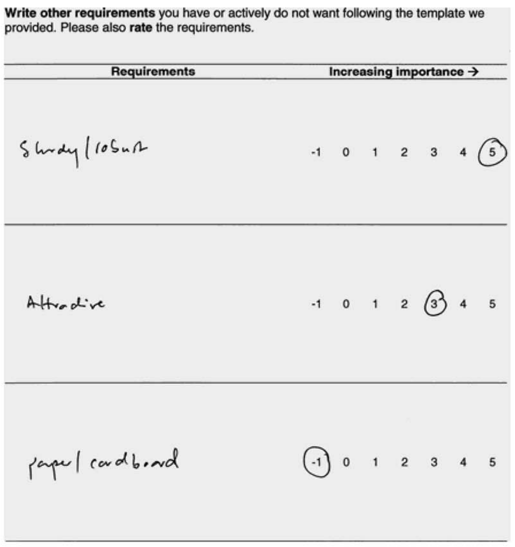
Requirements	Increasing importance →						
1. Better user experience	-1	0	1	2	3	[✓ 4]	5
1.1 Access cards that are easier to use with more accurate scanning	-1	0	1	2	3	4	[✓ 5]
1.2 For library entrance, remove the need to put card in exact location for barcode scanning	-1	0	[✓ 1]	2	3	4	5
1.3 All in one card	-1	0	1	2	3	4	[✓ 5]
1.3.1 ID card and session card	-1	0	1	2	3	4	[✓ 5]
1.3.2 Library card	-1	0	1	2	3	[✓ 4]	5
1.3.3 Bloomsbury fitness	-1	0	[✓ 1]	2	3	4	5
1.3.4 Club and societies	-1	0	[✓ 1]	2	3	4	5
1.3.5 Cashless vending	-1	0	[✓ 1]	2	3	4	5
1.3.6 Time and attendance	-1	0	[✓ 1]	2	3	4	5
1.3.7 Computer Logon	-1	0	[✓ 1]	2	3	4	5
1.3.8 Santander Bank Card	-1	[✓ 0]	1	2	3	4	5
2. Improve processes (reduce manual data entry and improve efficiency for access control and library processes)	-1	0	1	2	3	4	[✓ 5]
2.1 Library barcode generated together with card (less 1 library queue)	-1	0	1	2	3	[✓ 4]	5
2.2 Import photos from registry for advance card production (save queuing time for students)	-1	0	1	[✓ 2]	3	4	5

Fig. 7 (Continued).

3) time spent, by rating each criterion High, Medium, or Low (Fig. 7c). The respondents are required to complete this question before proceeding to the next to avoid the predefined list of requirements in the next question from influencing their answers.

3. **Rated profile (RateP).** Respondents rate a predefined list of requirements, from 0 (not important) to 5 (very important), and -1 for requirements they actively do not want (Fig. 7d). The predefined list consists of requirements extracted from the earliest draft requirements produced by the project team to reflect the actual initial requirements in RALIC. One extra requirement was added to the list, which is combining Santander Bank Card with UCL access card (Fig. 7d, Item 1.3.8). This requirement was being considered at the time of the survey and it was an opportunity to use the survey to elicit the stakeholders' views on the requirement. Respondents are also asked to add requirements not in the predefined list and rate those requirements (Fig. 7e). Once they start on RateP, they cannot return to RankP. As before, respondents provide feedback on the elicitation method after they have completed the question.
4. **Point test profile (PointP).** Respondents are allocated 100 points each to distribute among the requirements they want from RateP (Fig. 7f). The requirements include both the predefined ones and

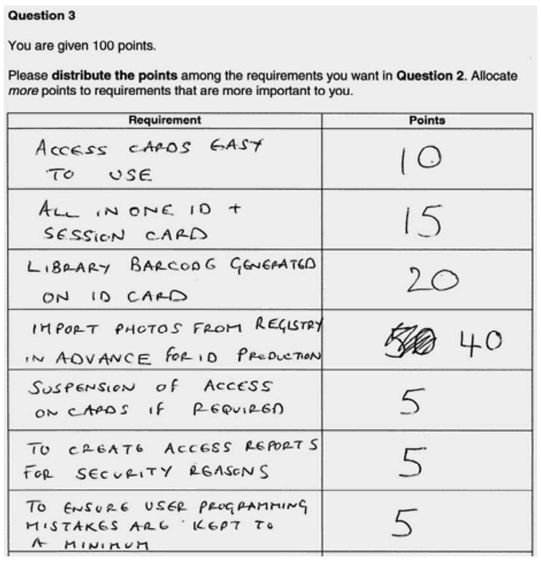
(e)



Write other requirements you have or actively do not want following the template we provided. Please also **rate** the requirements.

Requirements	Increasing importance →
Sturdy/robust	-1 0 1 2 3 4 [✓ 5]
Attractive	-1 0 1 2 [✓ 3] 4 5
paper/cardboard	[✓ -1] 0 1 2 3 4 5

(f)



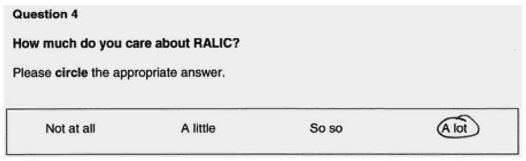
Question 3

You are given 100 points.

Please **distribute the points** among the requirements you want in **Question 2**. Allocate *more* points to requirements that are more important to you.

Requirement	Points
ACCESS CARDS EASY TO USE	10
ALL IN ONE ID + SESSION CARD	15
LIBRARY BARCODE GENERATED ON ID CARD	20
IMPORT PHOTOS FROM REGISTRY IN ADVANCE FOR ID PRODUCTION	~~50~~ 40
SUSPENSION OF ACCESS ON CARDS IF REQUIRED	5
TO CREATE ACCESS REPORTS FOR SECURITY REASONS	5
TO ENSURE USER PROGRAMMING MISTAKES ARE KEPT TO A MINIMUM	5

(g)



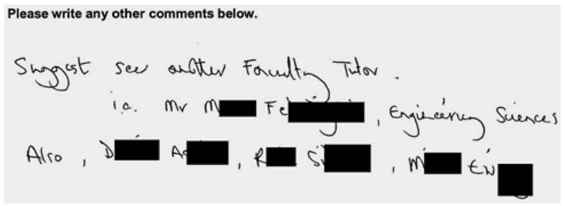
Question 4

How much do you care about RALIC?

Please **circle** the appropriate answer.

Not at all A little So so [✓ A lot]

(h)



Please write any other comments below.

Suggest see another Faculty Tutor .
i.e. Mr M[redacted] Fe[redacted], Engineering Sciences
Also, D[redacted] A[redacted], R[redacted] S[redacted], M[redacted] Ev[redacted]

Fig. 7 (Continued).

the additional ones they provide. Respondents are asked to allocate more points to the requirements that are more important to them. Again, respondents provide feedback on the elicitation method.

5. **Interest and comments.** Finally, respondents reveal their interest in the RALIC project in terms of "not at all," "a little," "so so," or "a lot" (Fig. 7g). Respondents also provide any other comments they have on the study (Fig. 7h). This part aims to learn more about the respondents and collect extra information to support the analysis of the results.

TABLE 5
Data Collected from 87 Respondents

Data	Amount
Stakeholder details	87 sets of details
RankP	
Ratings	415 ratings
Requirements	51 items
Specific Requirements	132 items
RateP	
Ratings	2,396 ratings
Requirements	48 items
Specific Requirements	104 items
PointP	
Ratings	699 ratings
Requirements	45 items
Specific Requirements	83 items
Feedback on elicitation method	783 ratings
Interest in RALIC	79 ratings

When the respondents had completed their questionnaire, they were interviewed for their survey experience and the rationale behind their answers. Similarly to the interviews in the StakeNet survey, these interviews were semistructured, allowing the questions to be modified and new questions to be brought up, depending on what the respondents say [96]. Some questions include:

- What do you think about the StakeRare method?
- Why is requirement R bad? (If the respondent actively does not want a requirement.)
- Why is requirement R important to you? (If the respondent allocated many points to a particular requirement.)
- Which elicitation method do you prefer and why?

A total of 87 stakeholders (out of the 127 stakeholders identified in Step 1) were surveyed. Table 5 summarizes the amount of data collected from the survey. RateP received the highest number of ratings, followed by PointP, and then RankP. The predefined list in RateP has the advantage of suggesting requirements that the respondents are unaware of, but has the disadvantage of enabling respondents to rate as many requirements as they like. PointP has fewer ratings than RateP, as the limitation of points encouraged the stakeholders to rate only the requirements that they needed most.

The data collected from the survey were processed and cleaned. The survey revealed that although all stakeholders provided short statements of their requirements (e.g., short clauses, or one to two sentences per requirement), very few stakeholders adhered to the requirements template provided to them at the start of the survey (Fig. 6). A respondent experienced in business analysis advised that in requesting input from stakeholders, restrictions and templates should be kept to a minimum to encourage response, and the requirements engineers should process the data after collection. As RateP and PointP used a predefined list of requirements, less cleaning was required. Data cleaning focused on RankP as follows:

- **Same requirement different wording.** Different statements describing the same requirement were merged. For example, in RALIC, "all in one card" was used interchangeably with "one card with multiple functionality"; hence, the two requirements were merged.
- **Same requirement different perspective.** Stakeholders with different perspective may express the same requirement in a different way; hence, these requirements were merged. For example, Library Systems had the requirement "to import barcode from Access Systems," but Access Systems had the requirement "to export barcode to Library Systems."
- **One statement many requirements.** Statements containing more than one requirement were split into their respective requirements.
- **Classification into the requirements hierarchy.** The requirements were grouped under their respective project objectives. Specific requirements were classified into their respective requirements.
- **Duplicate entries.** If a stakeholder provided a requirement more than once, only the requirement with the highest rank (i.e., assigned with the smallest number) was kept.
- **Missing fields.** A valid preference is in the form of *<stakeholder, requirement, rating>*. Three stakeholders provided only a requirement and did not rank their requirement. The rank of 1 was assigned to such requirements because if the stakeholder provided one requirement, then that requirement was assumed to be the most important to the stakeholder.
- **Tied ranks.** Some respondents provided tied ranks for their requirements (e.g., two requirements with the same rank). Fractional ranking was used to handle tied ranks. In RankP, the range of the ranking depends on the number of requirements provided by the stakeholders, and this variability affects the prediction and prioritization. Hence, normalization was done such that the sum of all the ranks from a stakeholder adds up to 1 in order to ensure that all rankings were within the same range of 0 to 1 for each stakeholder. The rating for "actively do not want" was converted to 0.

In RateP, if the respondents entered additional requirements, then the requirements were cleaned the same way as they were in RankP for the items "same requirement different wording," "same requirement different perspective," and "one statement many requirements." When providing additional requirements in RateP, some respondents indicated which project objective in the predefined list the requirements belong to, hence reducing the need for classification into project objectives. For duplicate entries in RateP, the requirement with the highest rating was kept.

For PointP, the ratings were normalized such that each stakeholder's allocated points added up to 100 to remove arithmetic errors during the survey. For duplicate entries in PointP, the requirement with the most points was kept.

In addition to the data cleaning, Step 2 of StakeRare involves propagating the ratings up and down the hierarchy. If a stakeholder rates a lower level requirement but does not

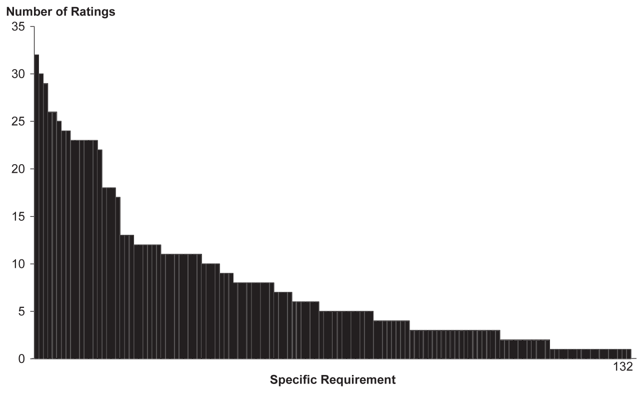


Fig. 8. RankP: Specific requirement versus number of ratings.

rate the high-level requirement, then StakeRare assumes the stakeholder provides the same rating to the high-level requirement. If a stakeholder rates a requirement but not its specific requirements, then StakeRare assumes the stakeholder provides the same rating to all the specific requirements. This propagation expanded the requirements' ratings into the ratings of their corresponding specific requirements, resulting in a total of 1,109 ratings on specific requirements for RankP, 3,113 for RateP, and 1,219 for PointP.[7]

When plotting the specific requirements in RankP against the number of positive ratings, the result resembles a power-law distribution (Fig. 8), with a few dominating requirements to the left receiving many ratings and a long tail to the right with many requirements receiving a few ratings [97]. A power-law graph is expected to emerge when there is a large population of stakeholders, a large number of ratings, and a high freedom of choice [97]. The graph for RateP has a large number of dominating requirements due to the predefined list (Fig. 9) and the graph for PointP has a long tail but does not have clear dominating requirements because of its point restriction (Fig. 10).

In Step 3, collaborative filtering is used to predict other requirements a stakeholder may need based on the profile they provide in Step 2. As discussed in Section 4.1, no additional input is required from the respondents.

The fourth and final step of StakeRare gathers all the stakeholders' initial ratings in Step 2 and their approved ratings from Step 3 to prioritize all the requirements for the project. This step uses the priority of stakeholders and their roles from Step 1. For this priority, the StakeNet list produced by the betweenness centrality measure was used because the evaluation from the previous work found the list to be the most accurate compared to the lists produced by the other social network measures [12], [89]. As three data sets, RankP, RateP, and PointP, were collected during the survey, three prioritized lists of requirements were produced.

4.4 Ground Truth

The ground truth of requirements in RALIC is the actual list of requirements implemented in the project, prioritized based on their importance in the project. The ground truth consists of 10 project objectives, 43 requirements, and 80 specific requirements.

7. The complete RankP, RateP, and PointP data sets are available at http://www.cs.ucl.ac.uk/staff/S.Lim/phd/dataset.html.

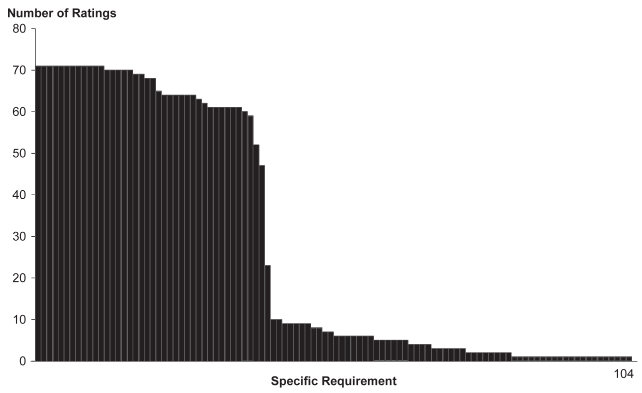


Fig. 9. RateP: Specific requirement versus number of ratings.

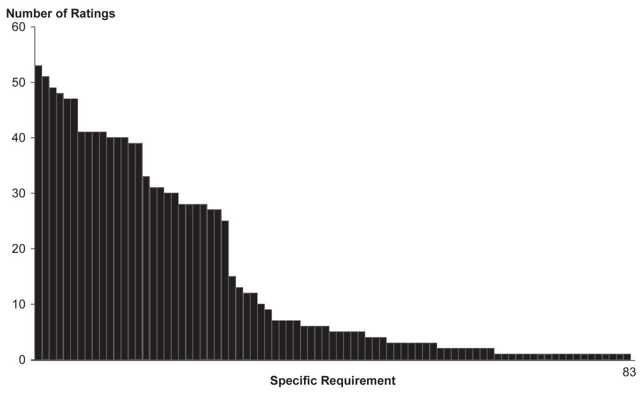


Fig. 10. PointP: Specific requirement versus number of ratings.

4.4.1 Construction

The ground truth was built by the author in three stages.

Stage 1: Identify requirements. The ground truth project objectives, requirements, and specific requirements were identified by analyzing project documentation such as the project plan, functional specification, meeting minutes, and post implementation report. For example, the project plan revealed that a project objective was "to improve security." A requirement to achieve the project objective was "to enable security/reception staff to validate the cardholder's identity." Two specific requirements were "to enable security/reception staff to check that the appearance of the cardholder matches the digitally stored photo" and "to enable security/reception staff to check the cardholder's role."

RALIC requirements were organized into three levels of hierarchy: project objectives, requirements, and specific requirements. A requirement that contributed toward a project objective was placed under the project objective, and a specific requirement that contributed toward the requirement was placed under the requirement. When placing requirements and specific requirements in the hierarchy, a requirement is placed under the objective that it influences most and, similarly, a specific requirement is placed under the requirement that it influences most.

Stage 2: Pairwise comparison. This stage uses pairwise comparison to prioritize the project objectives, requirements, and specific requirements from Stage 1. The pairwise comparison method discussed in the background section was used as it includes much redundancy and is thus less sensitive to judgmental errors common to techniques using absolute assignments [56]. The requirements from each level were considered separately.

- **Project objectives.** The project objectives were arranged in an $N \times N$ matrix where N is the total number of project objectives. For each row, the objective in the row was compared with respect to each objective in the rest of the row in terms of their importance to the project. The comparison is transitive. The project objective that contributes more to the success of the project is more important. For example, in RALIC, the objective "to improve security and access control in UCL" (labeled as Security in Fig. 11) was more important compared to the objective "to design the access card" (labeled as Card design in Fig. 11). The objective considered to be more important in each pairwise comparison was placed in the corresponding cell of the matrix (Fig. 11). If the two project objectives were equally important, they were both placed in the cell. For example, the objective "to design the access card" was equally important as the objective "to reduce cost" (labeled as Cost). Hence, in Fig. 11, they both appear in the corresponding cell of the matrix.
- **Requirements.** Requirements were prioritized the same way as project objectives. A requirement was more important if it contributed more toward the project objectives. For example, in RALIC, between the requirements "granting access rights" and "cashless vending," "granting access rights" was more important as it contributed highly toward the project objectives "to improve security" and "to improve processes," but "cashless vending" only contributed toward "extensible future features."
- **Specific requirements.** Specific requirements were prioritized the same way as requirements. A specific requirement was more important if it contributed more toward the requirements.

Stage 3: Prioritize requirements by their project objectives. This stage produced the ground truth of requirements, which consists of a prioritized list of project objectives, and within each project objective, a prioritized list of requirements, and within each requirement, a prioritized list of specific requirements. To produce the ground truth, project objectives were ranked by the number of cells in the matrix that contains the project objectives, from the most to the least. For each project objective, the requirements were prioritized from the requirements that appeared in the most cells to the least, and for each requirement, the specific requirements are prioritized from the specific requirements that appear in the most cells to the least.

4.4.2 Validation

The ground truth was validated by management-level stakeholders and stakeholders who were involved in a

	Security	Card design	Cost
Security	–	Security	Security
Card design	–	–	Cost, Card design
Cost	–	–	–

Fig. 11. 3×3 matrix for project objectives.

major part of the project. These stakeholders were interviewed to validate its completeness and accuracy in prioritization. The interviews were conducted after StakeRare was applied to RALIC. As StakeRare was evaluated by surveying the stakeholders, conducting the interviews after the surveys prevents the survey answers from being influenced by the interviews. The interviews increased the confidence that the ground truth is objective and accurate, and is representative of the actual requirements and their prioritizations in RALIC.

During the interviews, the stakeholders were provided with the description of the RALIC project and reminded about the purpose of the study, which was to identify and prioritize requirements. The stakeholders were also provided with the ground truth, and the purpose of the ground truth to evaluate StakeRare's output was explained to them. The stakeholders were asked to inspect the ground truth and were presented with the following questions:

- Are there any project objectives, requirements, and specific requirements that are missing from the ground truth?
- Are there any project objectives, requirements, and specific requirements that should not be included in the ground truth?
- Are any of requirements incorrectly prioritized? If so, why?

The feedback based on the questions was used to amend the ground truth. Missing requirements that were pointed out were confirmed with project documentation as this work only considered documented requirements. The interviews revealed that the ground truth of requirements was complete. However, some stakeholders pointed out that as the project was completed a while ago, they may not have remembered all the requirements, and the best way to check for completeness would be to consult the project documentation.

Disagreements in the prioritizations were confirmed with the other stakeholders and justified before amendments were made to the ground truth.

4.5 Existing Method List

The existing method list of requirements is an unprioritized list of requirements identified by the project team at the start of the project using existing methods. The project team used traditional elicitation techniques, which included meetings and interviews with key stakeholders (approximately 20 stakeholders representing 30 roles) determined by the project board. The existing method list consists of 10 project objectives, 43 requirements, and 56 specific requirements. These requirements were identified from the start of the project until the date the requirements were signed off by the project board.

The project team spent about 127 hours to produce the existing method list. This number is an approximation calculated from the total number of hours the stakeholders spent in meetings until the requirements were signed off.

4.6 Method and Results

This section describes the method to evaluate each research question in Section 4.1 and the results.

4.6.1 RQ1. Identifying Requirements

The first research question asks:

- How many requirements identified by StakeRare and the existing method used in the project are actual requirements as compared to the ground truth?
- How many of all the actual requirements in the ground truth are identified by StakeRare and the existing method used in the project?

Method. The list of requirements identified by StakeRare and the existing method was compared against the ground truth in terms of precision and recall, the two metrics widely used in the information retrieval literature [91].

The precision of identified requirements is the number of actual requirements in the set of identified requirements divided by the total number of identified requirements (4-1):

$$precision = \frac{|\{X\} \cap \{GroundTruth\}|}{|\{X\}|}, \tag{4-1}$$

where X is the set of requirements identified by StakeRare or the existing method, and $GroundTruth$ is the set of requirements in the ground truth.

The recall of identified requirements is the number of actual requirements in the set of identified requirements divided by the total number of actual requirements (4-2):

$$recall = \frac{|\{X\} \cap \{GroundTruth\}|}{|\{GroundTruth\}|}, \tag{4-2}$$

with X and $GroundTruth$ the same as for precision. Both precision and recall range from 0 to 1. A precision of 1 means all the identified requirements are actual requirements. A recall of 1 means that all the actual requirements are identified.

As explained in the StakeRare method, the requirements in StakeRare are organized into a hierarchy of project objectives, requirements, and specific requirements. To measure the precision and recall of the identified requirements, both the requirements and specific requirements were considered. Project objectives were not considered because the different methods share the same project objectives.

The requirements returned by the methods can be at a finer, equal, or coarser grain compared to the ground truth. If the returned requirements were at a finer grain, then the results were considered a match. For example, if ground truth returned "control access to departments," and StakeRare returned "control access to the Department of Computer Science, Department of Engineering, etc.," then it was considered that StakeRare returned a requirement that matched the ground truth. Otherwise, if the returned requirements were at a coarser grain than the ground truth, then the results were considered not a match. If they were of equal grain, the descriptions of the requirements were compared. Match was independent of exact details such as numeric parameters. For example, if StakeRare returned the requirement "a minimum of five years support from the software vendor," but the actual requirement in the ground truth was "a minimum of 10 years support from the software vendor," then it was considered that StakeRare returned a requirement that matched the ground truth. This is because in an actual project, specific details could be modified during the project as long as the requirement was identified.

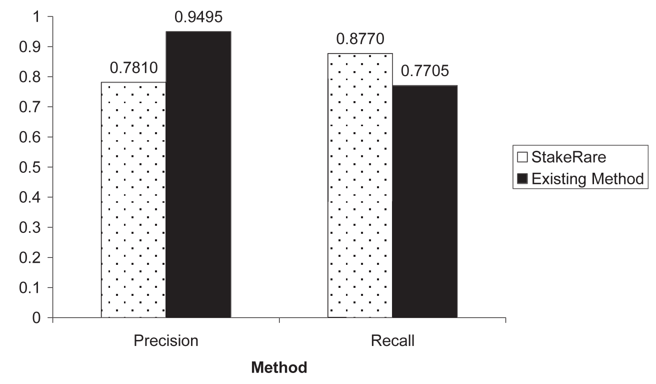


Fig. 12. Identifying requirements. (Comparison between RankP, RateP, and PointP can be found in RQ4 in Fig. 14.)

Results. 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 (Fig. 12). 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 stakeholders who were involved in the process. Hence, StakeRare's approach of asking stakeholders with different perspectives increased the completeness of the elicited requirements, which is critical to build a system that meets the stakeholders' needs.

The majority of the requirements missing in the StakeRare list were technical constraints such as "the database platform must support Microsoft SQL Server and/or Oracle Server" and "the system manufacturer must be a Microsoft Certified Partner," although technical stakeholders were involved in the survey. One reason could be the survey method of asking stakeholders to provide requirements on the spot, which may result in a bias to the type of requirements that can be identified using this approach. Future work should investigate this further.

StakeRare had a lower precision compared to the existing method. This is because in StakeRare, stakeholders are free to suggest requirements, which may not always be implemented. For example, some RALIC stakeholders wanted to replace the existing access control system with thumb readers, but this requirement was not implemented, hence lowering the precision. Other such requirements include enabling the tracking of UCL alumni, card to be cheap (10 pence per card), and combine access card with travel card. These requirements do not appear in the existing method and ground truth lists. Nevertheless, in requirements elicitation, it is better to be more complete but less precise (identify extra requirements which are not implemented), rather than to be precise (identify only the requirements that are implemented) but miss out on some requirements [1], [39]. Due to the wide coverage of requirements in StakeRare, the majority of missing requirements are of low priority.

Not all stakeholders had requirements. For example, some developers who had not provided any requirements explained that their job was to implement the requirements given to them; others highlighted prerequisites for their job. For example, when completing the survey, the maintenance team entered technical documentation as their requirement. The StakeRare lists were less precise partly due to these requirements, as they were not in the ground truth. These requirements do not appear in the existing method list.

By asking stakeholders to indicate requirements they actively did not want, StakeRare uncovered conflicts. For example, security guards preferred not to have UCL's logo on the ID card for security reasons had the card been lost. Fitness center users preferred not to have too many features on the ID card as they had to exchange their cards for their locker keys for the duration of their use of the gym lockers. If the ID card had also served as a cash or bank card, then the fitness center users would prefer using a separate gym card to exchange for locker keys, which meant that the requirement "all in one card" would not be achieved.

Stakeholders recommended for a role may provide requirements for another role, as it is common for a stakeholder to have more than one role in a project. For example, a RALIC developer was also a user of the access cards. According to this stakeholder, "as a developer, I only care about system requirements. But as a member of staff, I want to be able to access required buildings, and by rating these [access related] requirements, I am answering the questionnaire as a member of staff."

During the surveys, some respondents recommended other stakeholders to be surveyed regarding a specific requirement (e.g., the comment in Fig. 7h). This suggests that the stakeholder analysis step should overlap with the requirements elicitation step to improve the quality of the stakeholder list and requirement list. This finding is consistent with the existing requirements engineering literature. According to Robertson and Robertson [39], the identifications of scope, stakeholders, and requirements are dependent on one another, and should be iterated until the deliverables have stabilized.

Stakeholders supported StakeRare's idea of being open and inclusive in requirements elicitation. When asked to represent all UCL students in RALIC, the student representative clarified that "I do not represent all 20 thousand students, even though they voted for me. Their views on data management, for example, can be very different from mine," and suggested a wider body of students to be surveyed using StakeRare to ensure a more representative view. A management-level stakeholder commented that StakeRare provides the opportunity to elicit different views from a large number of stakeholders, which can increase stakeholder buy-in, a vital element for project success.

4.6.2 RQ2. Prioritizing Requirements

The second research question asks: How accurately does StakeRare prioritize requirements as compared to the ground truth?

Method. StakeRare's output for the requirements engineers was measured against the ground truth of requirements in Section 4.4, in terms of accuracy of prioritization. The accuracy of prioritization for project objectives is the similarity between the prioritization of the project objectives by StakeRare and the prioritization in the ground truth. Pearson's correlation coefficient, ρ, was used to determine the similarity [91]. Values of ρ range from $+1$ (perfect correlation), through 0 (no correlation), to -1 (perfect negative correlation). A positive ρ means that high priorities in the ground truth list are associated with high priorities in the list of identified requirements. The closer the values are to -1 or $+1$, the stronger the correlation. The statistics software by Wessa[8] is used to calculate ρ in this work.

The computation of ρ requires the lists to be of the same size. Therefore, the measurement of ρ takes the intersection between the lists: Each list is intersected with the other, and fractional ranking [90] is reapplied to the remaining elements in that list. Missing requirements and additional requirements in the StakeRare list were accounted for when answering RQ1 on the completeness of the returned requirements. For requirements, ρ was measured for each list of requirements per project objective and the results were averaged, and the standard deviation was calculated for each average. The same was done for specific requirements by measuring ρ for each list of specific requirements per requirement, averaging the results, and calculating the standard deviation for each average.

As a control, prioritized lists were produced using unweighted stakeholders (referred to as Unweighted in the results), i.e., each stakeholder's weight is 1. The existing method list was not compared as it is unprioritized.

Results. StakeRare prioritized requirements accurately compared to the ground truth (Fig. 13). In prioritizing project objectives and requirements, StakeRare had a high correlation with the ground truth ($\rho = 0.8$ and $\rho = 0.7$, respectively). It was less accurate in prioritizing specific requirements ($\rho = 0.5$). Weighting stakeholders by their influence increased the accuracy of prioritizing project objectives and requirements by over 10 percent, but not for specific requirements. This influence is produced by applying social network measures to the stakeholder network. As such, the results show that using social networks generally improves the accuracy of prioritizing requirements. Nevertheless, the low accuracy in prioritizing specific requirements should be further investigated, as these requirements are crucial to system development.

StakeRare's output was presented to interested stakeholders after this work was completed. The director of Information Systems, who was experienced in options rankings for major decisions, commented that StakeRare's prioritization is "surprisingly accurate."

Interestingly, analysis of the meeting minutes and post implementation report revealed that the project team spent a disproportionate amount of time discussing less important requirements during project meetings. According to the minutes and interviews with the stakeholders, a significant amount of time was spent discussing card design. However, the project objectives "better user experience" and "improve processes" have a higher priority than "card design" in the ground truth. The post implementation report identified a key area relating to user experience and processes that was not given adequate attention. According to the report, "A student's first experience of UCL should not be spent all day in a queue. . . . the queuing arrangements should be revised and a more joined up approach with Registry taken. Last year, the queue for the RALIC card meant students were queuing outside—we were fortunate with the weather."

StakeRare accurately prioritized the project objectives "better user experience" and "improve processes" as having higher priority than "card design." Had StakeRare been used, the project team would have realized that "card design" had a lower priority compared to the other two project objectives, and might have spent less time discussing card design and more time discussing various ways to improve processes and user experience.

8. Wessa, P. (2009) Free Statistics Software, Office for Research Development and Education, version 1.1.23-r4, URL http://www.wessa.net/.

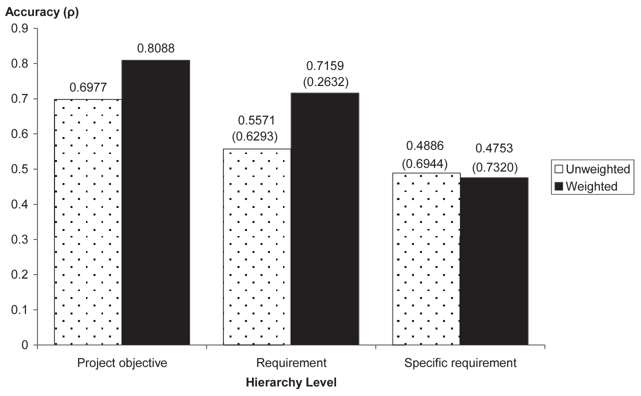


Fig. 13. Prioritizing requirements (standard deviation in parentheses).

4.6.3 RQ3. Survey Response and Time Spent

The third research question asks:

- Are stakeholders motivated to provide requirements for StakeRare?
- How much time did stakeholders spend in identifying and prioritizing requirements as compared to the existing method in the project?

Method. To determine the stakeholders' motivation to provide ratings, the response rate of the survey was calculated as the number of stakeholders who responded over the total number of stakeholders who were contacted, expressed as a percentage. In the calculation, "nonstakeholders" and stakeholders who have left but have yet to be replaced were excluded. For stakeholders who responded, their level of interest in the project, which was collected during the survey, was also investigated.

The time spent using StakeRare was calculated using (4-3):

$$time_{StakeRare} = time_{predefined\_list} + \frac{time_{questionnaire}}{3}, \qquad (4\text{-}3)$$

where $time_{predefined\_list}$ is the time spent building the predefined list of requirements. As the predefined list of requirements was taken from the first draft requirements, $time_{predefined\_list}$ is the number of hours the stakeholders spent in meetings until the draft requirements were produced on 8 August 2005, which was 61 hours. $time_{questionnaire}$ is the total time spent answering the questionnaire, which is the total survey time minus the time spent introducing StakeRare and interviewing the respondents after the survey, which was approximately 10 minutes per respondent. $time_{questionnaire}$ is divided by 3, to consider only one elicitation method out of the three.

$time_{StakeRare}$ is the time spent using StakeRare from the stakeholders' perspective. Only the respondents' time spent was calculated, as this researcher's presence while they were completing the questionnaires was just to observe them for research purposes. The calculation was an approximation that assumed the elicitation methods take an equal amount of time. The time spent using StakeRare was compared with the time spent using the existing method in the project, which was 127 hours as reported in Section 4.5.

Results. Stakeholders were motivated to provide ratings. The survey response rate was 79 percent, about 30 percent higher than the weighted average response rate without regard to technique[9] [98]. Most of the stakeholders who responded were very interested in RALIC; only 13 percent indicated that they have little interest and 3 percent no interest. Those with little or no interest may not have responded in tool-based implementations of the survey.

The time spent using StakeRare was 71 hours, 56 hours less than the time spent using the existing method in the project. While the existing method in the project returned an unprioritized list of requirements, StakeRare's list of requirements was accurately prioritized and highly complete, as shown in the previous research questions. These findings indicate that a more adequate involvement of stakeholders in this case produced better requirements.

Finally, many stakeholders preferred using StakeRare to provide requirements rather than attend lengthy meetings. In line with the existing literature, elicitation meetings used in existing methods can be time consuming and ineffective [6]. One stakeholder commented, "I was only interested in one issue, but had to sit through hours of meetings, where unrelated items were discussed. What a waste of time. With this method, I could just write it down and get on with my work!"

4.6.4 RQ4. Effective Support for Requirements Elicitation

StakeRare's default elicitation method is RateP, but the evaluation administered two other elicitation methods, RankP and PointP, to explore the effectiveness of different methods. The fourth research question asks:

- **Best elicitation method.** Between the three elicitation methods RankP, RateP, and PointP, which produces the most complete list of requirements and most accurate prioritization for the requirements engineer? Are the results consistent regardless of the elicitation method used?
- **Stakeholders' preference.** Between the three elicitation methods RankP, RateP, and PointP, which do the stakeholders prefer?
- **Rating all requirements.** If stakeholders are provided with a list of all the requirements in the project, how prepared are they to rate them all?

Method. *Best elicitation method.* Three elicitation methods were administered to explore the effectiveness of different methods. In RankP, stakeholders were asked to enter their requirements without providing an initial list of requirements. In RateP, stakeholders were asked to rate a predefined list of requirements and provide additional requirements. In PointP, stakeholders were asked to allocate 100 points to the requirements they want in the same predefined list as RateP. To evaluate the different elicitation methods, the lists produced from RankP and PointP were compared against the ground truth, and RQ1 and RQ2 were answered using precision, recall, and accuracy as described previously. Then, the results from RankP and PointP were compared with the results from RateP.

Stakeholders' preference. To determine the elicitation method preferred by the stakeholders, the respondents' feedback on the elicitation methods was investigated. During the survey, each respondent rated RankP, RateP, and PointP in terms of the criteria: level of difficulty, effort required, and time spent. Each criterion was rated High, Medium, or Low. The ratings were converted into numeric values $(\mathrm{High} = 3,\ \mathrm{Medium} = 2,\ \mathrm{Low} = 1)$ to be averaged. The average rating on each criterion for each elicitation method and the standard deviation for each average were calculated for RankP, RateP, and PointP. The interviews conducted with stakeholders after the surveys provided the rationale behind the stakeholders' preference.

Rating all requirements. To determine how prepared stakeholders were to rate all the requirements, an alternative RateP questionnaire, which consisted of the predefined list of requirements as well as the requirements provided by other stakeholders, was prepared. An experiment was conducted with four stakeholders to rate this alternative questionnaire. The initial plan to involve more stakeholders in the experiment fell through as the stakeholders were reluctant to rate a long list of requirements.

Results. *Best elicitation method.* In identifying requirements (RQ1), RateP had the best overall results, RankP had the highest recall, and PointP had the highest precision (Fig. 14). RankP had the lowest precision because stakeholders were free to express what they want. PointP had the highest precision as limited points encouraged stakeholders to only suggest requirements they really needed. Although RateP collected the highest number of ratings, it has a lower recall compared to RankP. This is because in RateP, some stakeholders do not provide additional requirements after they have rated the requirements in the predefined list. Requirements missing from RateP that are in RankP are mostly specific requirements. For example, RateP has the requirement "import data from other systems" but omitted the specific systems to import the data from.

In prioritizing requirements (RQ2), RateP produced the most accurate prioritization for project objectives and requirements (Fig. 15). The results in all three data sets showed that weighting stakeholders generally increased the accuracy of prioritization. The most significant improvement was RateP requirements with an increase of 16 percentage points. Only for RankP requirements and RateP specific requirements was there no improvement.

The elicitation method influenced the prioritization. For example, the project objective "extensible for future features" was prioritized disproportionately high in RateP, and disproportionately low in PointP and RankP. As RateP allowed stakeholders to rate as many requirements as they wanted, "nice to have" features were rated high. In PointP, stakeholders were given limited points, hence they allocated the points for requirements they really need rather than future features. In RankP, developer related project objectives such as "compatibility with existing UCL systems" and "project deliverables"[10] were prioritized disproportionately high. This was because developers who participated in the survey listed development requirements (e.g., the maintenance team needed the requirements

9. In the study by Yu and Cooper [98], the sample sizes for 497 response rates from various survey methods (e.g., mail surveys, telephone surveys, and personal interviews) varied from 12 to 14,785. As such, the response rate averages are weighted by the number of contacts underlying the response rate.

10. This project objective contains development related requirements such as requirements documentation, technical documentation, and change management.

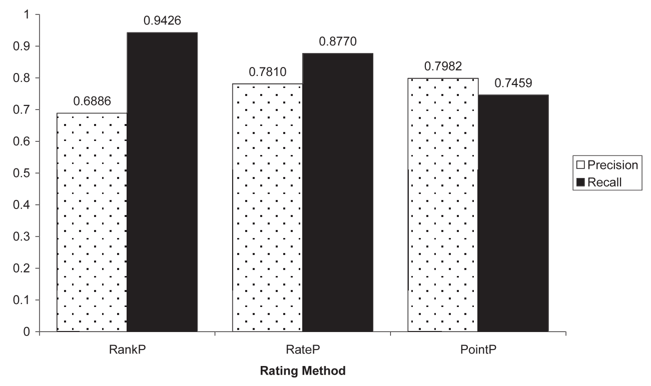


Fig. 14. Identifying requirements: RankP, RateP, and PointP.

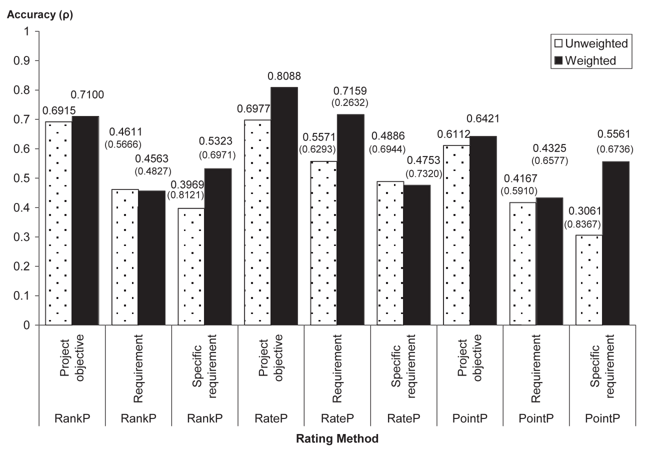


Fig. 15. Prioritizing requirements: RankP, RateP, and PointP (standard deviation for requirements and specific requirements in parentheses).

documentation for their work) rather than system requirements (e.g., easier to use access cards), and the other stakeholders provided relatively fewer requirements in RankP compared to RateP and PointP.

RateP had the advantage of suggesting requirements that the respondents were unaware of, which improved the accuracy of prioritization. Upon looking at the predefined requirements in RateP, some stakeholders commented that they were reminded about requirements which did not cross their minds while they were completing RankP. For example, the requirement "centralized management of access and identification information" had high priority in the ground truth. But in RankP, only one respondent provided this requirement, resulting in a biased overall prioritization. The accuracy of prioritization for the list of requirements under the same project objective was $\rho = -0.1$, indicating that the list was negatively correlated with the ground truth. In contrast, this requirement, which was provided in the predefined list in RateP, received positive ratings from 68 respondents, resulting in a prioritization that was highly correlated with the ground truth ($\rho = 0.7$).

Stakeholders found it easy to point out requirements they actively do not want from the predefined list of requirements in RateP. However, they found it more difficult to do so in RankP where no requirements were provided. Although RateP suggested requirements to stakeholders, stakeholders did not blindly follow the suggestions. For example, although the "Santander bank card" requirement was in the predefined list, the majority of stakeholders were against it. The requirement received a rating of 0 from 25 respondents and a rating of -1 (actively do not want) from 20 respondents. Only 23 respondents rated it positively, suggesting that if UCL were to implement it, the card would not be well received.

Stakeholders' preference. The majority of stakeholders preferred RateP as they found it to require the least effort. Nevertheless, as they had to go through a predefined list of requirements, they found it more time consuming than RankP and PointP, where they only entered the requirements they wanted. For example, a security guard found the RateP list too student focused, and commented that it was "tedious to go through a list of requirements that are mostly unrelated." In general, stakeholders found all three elicitation methods easy to complete, requiring little time and effort (Fig. 16). The average responses sat between Low and Medium for all three elicitation methods in all criteria.

Different types of stakeholders preferred different elicitation methods. Many decision makers preferred PointP, as they were used to making decisions under constraints. System users such as students and gym users preferred RateP, where options were provided. RankP was challenging to some as they found it difficult to articulate their needs without prompts. Nevertheless, stakeholders with specific requirements in mind preferred RankP, where they could freely articulate their requirements. Some stakeholders found the predefined list of requirements constraining. For example, a respondent had trouble rating the requirement "enable the gathering and retrieval of the time which individuals enter and leave buildings." He explained that the requirement should be worded as two requirements because he would provide a negative rating for gathering the time individuals enter buildings (he did not want the time he arrived at work to be recorded), but a high positive rating for gathering the time individuals leave buildings for security reasons.

Finally, many stakeholders found the arithmetic exercise in PointP distracting, especially those who allocated points at a fine level of granularity. As such, PointP was rated the highest in terms of effort and difficulty. Future implementations of PointP should provide automatic point calculations.

Rating all requirements. Although stakeholders rated most of the initial requirements, they were not prepared to rate the full list of requirements. The four stakeholders involved in rating the alternative questionnaire found the task tedious and time consuming. They preferred to only rate a subset of requirements that were relevant to them. One complained that she was bored and wanted to stop halfway. This suggests that it is useful to recommend relevant requirements to stakeholders when the list of requirements is long.

4.6.5 RQ5. Predicting Requirements

The fifth research question asks:

- How accurately can collaborative filtering predict stakeholder requirements?
- Are the results consistent regardless of the elicitation method used?

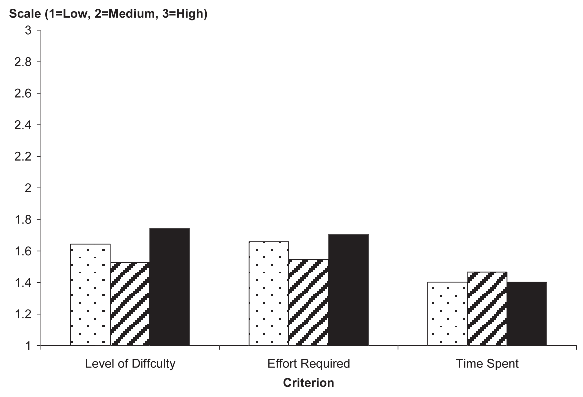


Fig. 16. Stakeholder feedback on elicitation method.

Method. To evaluate StakeRare's prediction accuracy, 10-fold cross validation [92] was used to predict the stakeholders' ratings for project objectives, requirements, and specific requirements. Tenfold cross validation is commonly used in machine learning research for assessing how the results of a statistical analysis will generalize to an independent data set [99], [100], [101]. Using 10-fold cross validation, the original sample is randomly partitioned into 10 subsamples. Of the 10 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining nine subsamples are used as training data [92]. The cross-validation process is then repeated 10 times (the folds), with each of the 10 subsamples used exactly once as the validation data. The 10 results from the folds are then averaged to produce a single estimation. The advantage of this method over repeated random subsampling is that all observations are used for both training and validation, and each observation is used for validation exactly once [92], [102].

The Weka[11] data mining software was used for the kNN algorithm and the evaluation. Cross validation was used to find the optimal value for k. This was done using the built-in cross validation in Weka by setting k to the total number of stakeholders who provided more than one rating. The resulting optimal value was corroborated through an exhaustive search using all possible values of k (from 1 to the total number of stakeholders who provided more than one rating) during preliminary experiments.

The mean absolute error (MAE) metric was used to evaluate the accuracy of the predictions [91]. MAE is a measure of the deviation of recommendations from their true user-specified values widely used in the collaborative filtering literature [103], [104], [105], [106]. The MAE is computed by first summing these absolute errors of the n corresponding rating-prediction pairs and then computing the average. Formally,

$$MAE = \frac{\sum_{i=1}^{n} |r_i - p_i|}{n}, \tag{4-4}$$

where n is the number of tested ratings and $<r_i, p_i>$ is a rating-prediction pair. The lower the MAE, the more accurately the recommendation system predicts user ratings. Stakeholders must provide at least one rating before their preference can be predicted; hence, stakeholders who provided only one rating were removed.

11. Weka version 3.6 http://www.cs.waikato.ac.nz/ml/weka/.

The results produced from using kNN with optimal k were compared to the following controls:

- **Random.** Random predictions for the ratings were produced with a uniform distribution within the rating range, which was different for RankP, RateP, and PointP (Table 6). The experiment was repeated 50 times and the average MAE was computed.
- **Max k.** All stakeholders were assumed to be the same by running kNN with $k =$ the total number of stakeholders who provided more than one rating.

To check if the result was consistent regardless of the elicitation method, the data sets for RankP and PointP were evaluated using the same experiment.

Results. StakeRare predicted the stakeholders' preference with high accuracy using the default RateP data set (Fig. 17). As expected, random predictions were inaccurate, with an MAE of about 2.5 for all three hierarchy levels. kNN with maximum k improved the prediction accuracy by more than half, and kNN with optimal k performed the best in all three levels.[12] This showed that identifying similar stakeholders improved prediction accuracy. The average MAE after applying kNN was about 1 (Fig. 17). This meant that if a stakeholder rated a requirement as 4, StakeRare's prediction of her rating was between 3 and 5. This result was comparable to that reported in the literature for standard collaborative filtering applications, such as movie rating [104]. In Jin et al.'s [104] experiments with two data sets (one with five ratings and one with six ratings), the MAE ranged between 0.8 and 1.3. The StakeRare RateP data set had six ratings (i.e., actively do not want, 1, 2, 3, 4, and 5).

The prediction results were consistent regardless of the elicitation method (Fig. 18). Random predictions were the least accurate, followed by kNN with maximum k, and kNN with optimal k produced the most accurate prediction. The MAE for RankP was small as the rank was normalized. The MAE for PointP was large, with random guessing having an MAE as large as 44.6, as possible ratings ranged between 0 and 100. kNN managed to reduce the MAE to as low as 3.9 for that data set.

4.6.6 RQ6. Predicting Requirements: Enhanced Profiles

The sixth research question asks:

- Does enhancing stakeholder profile by adding stakeholder information improve the accuracy of predicting stakeholder interest in requirements?
- Are the results consistent regardless of the elicitation method used?

Method. For enhancing stakeholders' profiles, two attributes were added to each stakeholder's profile: role (referred to as Role in the Results section) and number of ratings (referred to as #Rtgs in the Results section). The attributes were first added separately, and then together (referred to as Both in the Results section). The experiment as before was used to predict stakeholders' ratings using the enhanced profiles. Tenfold cross validation [92] was used to predict the stakeholders' ratings for project objectives, requirements, and specific requirements. The mean absolute error evaluation

12. The differences in MAE observed in the results reported in this paper may not always be significant.

TABLE 6
Data Characteristics

	RankP	RateP	PointP
Project Objectives			
Number of Stakeholders Providing > 1 Rating	66	75	71
Number of Items	10	10	10
Number of Ratings	249	438	270
Requirements			
Number of Stakeholders Providing > 1 Rating	71	75	71
Number of Items	51	48	45
Number of Ratings	461	1513	664
Specific Requirements			
Number of Stakeholders Providing > 1 Rating	76	75	75
Number of Items	132	104	83
Number of Ratings	1106	3112	1217
Rating Range*	$0 \leq x < 1$	$-1 \leq x \leq 5$	$0 < x \leq 100$

** As mentioned in Section 4.3 on data cleaning, the ratings for RankP have been normalized between 0 and 1, where the rating for actively do not want is 0. The ratings for RateP are integers. The ratings for PointP are real numbers.*

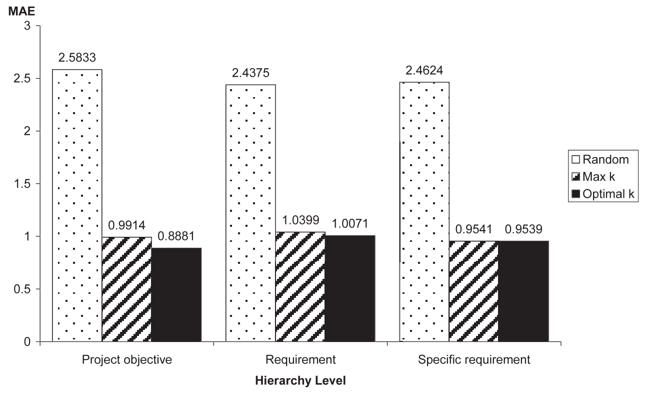
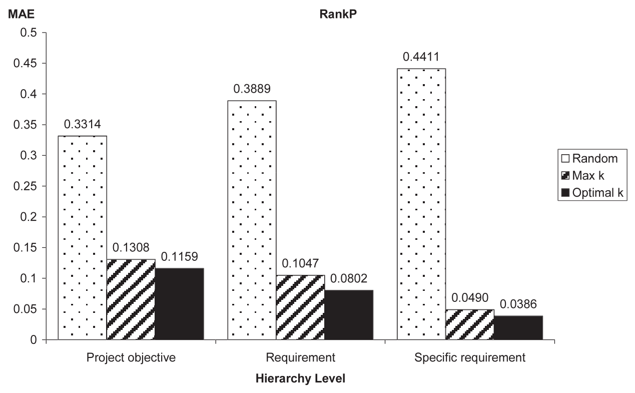


Fig. 17. RateP: Predicting requirements (smaller MAE indicates higher prediction accuracy).

metric was used to evaluate the accuracy of the predictions [91]. The control was kNN with optimal k from the previous research question (referred to as Basic in the Results section).

To check if the result was consistent regardless of the elicitation method, the data sets for RankP and PointP were evaluated using the same experiment.

Results. Enhancing the stakeholders' profiles improved the accuracy of predicting their requirements for the default RateP data set (Fig. 19). Adding stakeholder role improved prediction accuracy because stakeholders with the same roles tend to have similar requirements. For example, members of the UCL Development and Corporate Communications Office required the card to have UCL branding but security guards preferred otherwise for security reasons in case the cards were lost. In requirements and specific requirements, adding each attribute separately significantly improved the prediction accuracy, and adding both attributes produced the most accurate prediction. In project objectives, the improvement was less obvious, and adding each attribute separately produced better prediction than adding both attributes together.



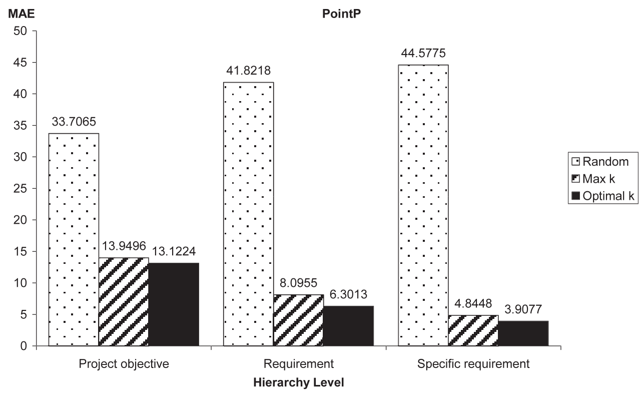


Fig. 18. Predicting requirements: RankP and PointP (smaller MAE indicates higher prediction accuracy).

The results were less consistent for RankP and PointP (Fig. 20). For PointP, prediction accuracy improved after adding roles to the profiles, but prediction accuracy became worse when both attributes were added. For RankP, no significant improvements could be observed after enhancing profiles. Hence, different attributes may be needed for different data sets to improve prediction accuracy.

4.6.7 RQ7. Predicting Requirements: Other Algorithms

The final research question asks:

- Does using other algorithms and combinations of algorithms improve the prediction accuracy?
- Are the results consistent regardless of the elicitation method used?

Method. To answer the question about using other algorithms and combinations of algorithms, linear regression (LR) was used to predict the stakeholders' preferences [92]. Principle Component Analysis (PCA) was also used to preprocess the data before the prediction algorithms (i.e., kNN or linear regression) were applied [92], [107], [108]. PCA is widely used in exploratory data analysis and for making predictive models [107]. It involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components [107]. The experiment in the previous research question was used to predict stakeholders' ratings using four permutations as follows:

- kNN with optimal k (kNN),
- PCA and kNN with optimal k (P+kNN),

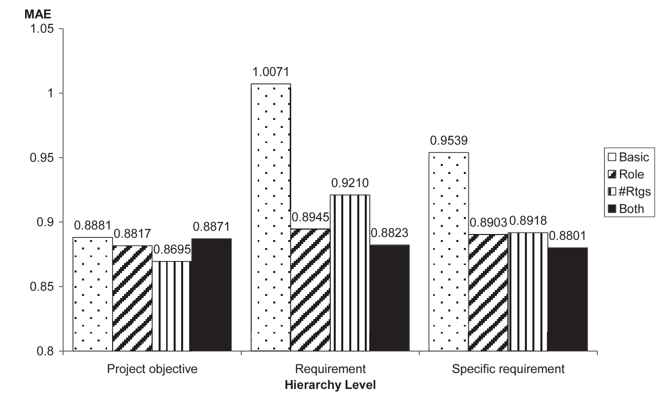


Fig. 19. RateP: Enhanced profiles (smaller MAE indicates higher prediction accuracy).

- linear regression (LR), and
- PCA and linear regression (P + LR).

To check if the result was consistent regardless of the elicitation method, the data sets for RankP and PointP were evaluated using the same experiment.

Results. The use of other algorithms improved the accuracy of prediction in general (Fig. 21). Applying PCA before *k*NN improved the prediction accuracy for project objectives and requirements, but not for specific requirements. As PCA finds the principle components, it potentially discards some information and focuses on the variables that make classification easiest. The lower prediction accuracy for specific requirements suggested that the specific requirements were more complex and needed more information for better classification. Using linear regression instead of *k*NN improved the prediction accuracy significantly for all three hierarchy levels. However, applying PCA before linear regression did not produce better results in general. For project objectives and specific requirements, it performed slightly worse.

The best algorithm differed across different data sets. For RankP, apart from the objective level, *k*NN was the most accurate (Fig. 22). For PointP, apart from the objective level, linear regression was slightly more accurate than the others. Applying PCA before *k*NN or linear regression consistently produced lower accuracy for specific requirements in all three data sets. This confirmed that specific requirements were complex and required more information for better classification.

4.6.8 Summary

The evaluation of StakeRare on RALIC provides clear evidence that StakeRare effectively supports requirements elicitation as follows:

- The first step of StakeRare, which uses the StakeNet method, identifies a highly complete set of stakeholders and prioritizes them accurately based on their influence in the project. The evaluation is described in [89] and [12].
- StakeRare identifies a highly complete set of requirements compared to the existing method used in the project. By eliciting requirements from various perspectives, StakeRare detects conflicts and has the potential of increasing stakeholder buy-in.

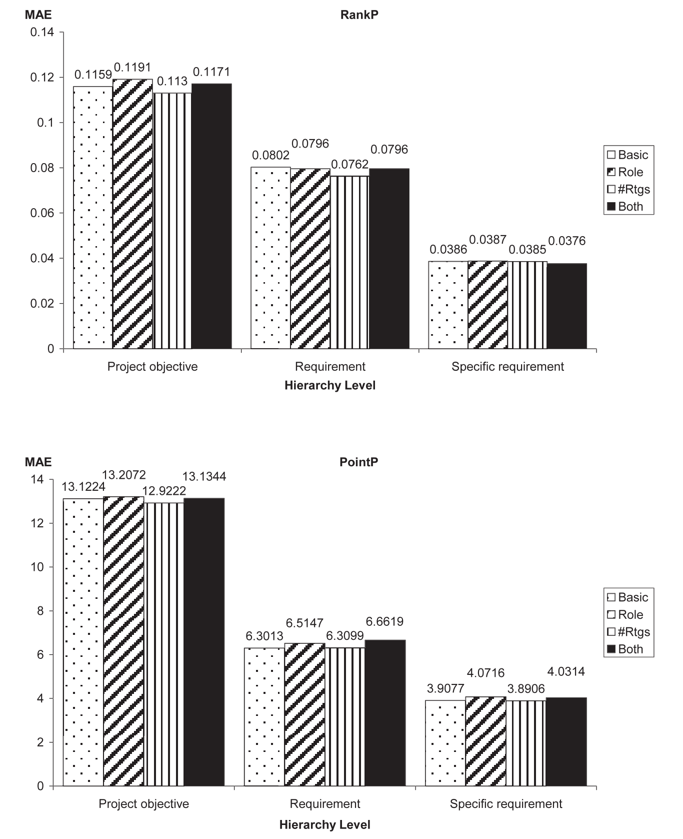


Fig. 20. Enhanced profiles: RankP and PointP (smaller MAE indicates higher prediction accuracy).

- Stakeholders are motivated to provide their requirements using StakeRare. StakeRare requires less time from the requirements engineers and the stakeholders as compared to the existing method used in the project.
- StakeRare prioritizes requirements accurately using the stakeholders' influence produced by the social network measures. The director of Information Systems, experienced in options rankings for major decisions, commended StakeRare's prioritization as "surprisingly accurate."
- The investigation of different elicitation methods, such as RankP and PointP, shows that StakeRare's elicitation method, RateP, which provides stakeholders with a predefined list of requirements as well as allowing them to add new requirements, is rated by stakeholders as low difficulty and requiring little effort. It also produces the most accurate prioritization of requirements. Nevertheless, stakeholders prefer not to be overloaded by information, which happens when there is a long list of requirements for them to rate.
- StakeRare handles information overload by drawing stakeholders' attention to only the relevant requirements that they are unaware of. The recommendations by StakeRare that are approved by the stakeholders will then improve global prioritization. The *k*NN collaborative filtering algorithm accurately predicts a stakeholder's requirements based on the requirements provided by similar stakeholders.

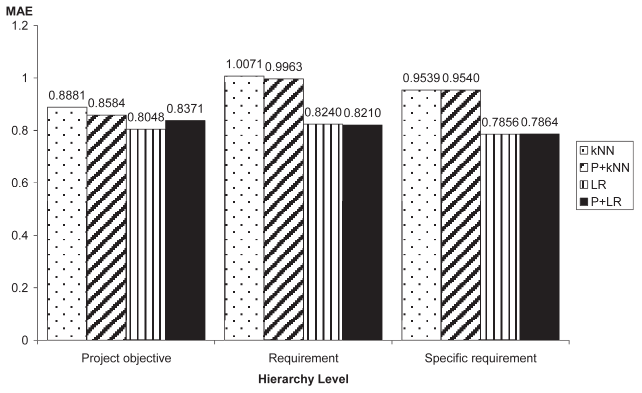


Fig. 21. Other algorithms (smaller MAE indicates higher prediction accuracy).

- Adding stakeholder profiles can increase prediction accuracy, and using other collaborative filtering algorithms can also improve prediction accuracy.

5 Threats to Validity

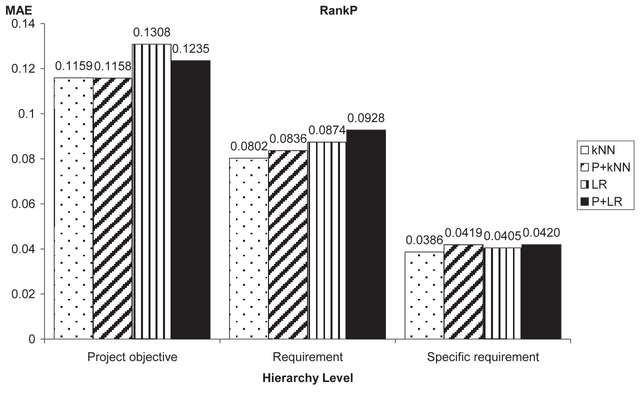
5.1 Single Completed Project

The main threat to validity in the evaluation is the use of one project to evaluate StakeRare, and that the project was already completed before StakeRare was applied. As such, there must be some caution in generalizing the results to other projects and organizations.

The StakeRare survey was conducted after the RALIC project was completed; hence, postproject knowledge may influence the results. In the survey, respondents were asked to articulate requirements for the existing system by imagining the situation before the RALIC project. Nevertheless, it is difficult for the respondents to do so without bias, as they had already been using the system and may not be aware of the difficulties before the RALIC system was implemented. More importantly, the respondents may be aware of the requirements due to their involvement in the project, and their knowledge may skew the evaluation results. In addition, the comparison of StakeRare with the existing method used in the project could be regarded as unfair because the existing method was used at the start of the project, when the stakeholders had no prior knowledge about the requirements.

Further analysis was conducted to investigate the effect of this threat on the evaluation results. Due to staff turnover and department restructuring, only 15 percent of the respondents were involved in decision making, stakeholder analysis, and requirements elicitation during the project; hence, their influence on the overall prioritization of stakeholder roles and requirements is low. The requirements provided by these respondents have a recall of 39 percent, which was approximately 50 percent lower than the recall of the requirements returned by StakeRare. As the respondents who were also involved in the actual project were mainly decision makers, their requirements missed out on process related requirements such as "faster card issue." In the StakeRare list, these requirements were identified by stakeholders who were involved in the process themselves.

Finally, despite this threat to validity to some of the results, the evaluation contains much more than a comparison against the existing method and the ground truth. They include a systematic comparison of various elicitation methods, prioritization methods, and collaborative filtering algorithms.



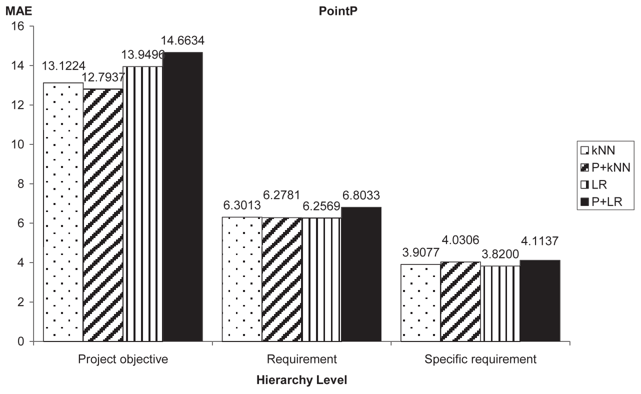


Fig. 22. Other algorithms: RankP and PointP (smaller MAE indicates higher prediction accuracy).

5.2 Ground Truth Construction

The ground truth of requirements in a software project is difficult to establish definitely, as the decision of which requirements to implement may involve politics and power struggles. One may claim that the ground truth is biased in the perspective of this researcher, thus affecting the results of the study. Nevertheless, it is argued that the ground truth is representative of the actual stakeholders and requirements in the project because the global perspective of the project was acquired from reviewing project documentation, observing the stakeholders' engagement with the project, and interviewing them. In addition, to increase the confidence that the ground truth is objective and accurate, the ground truth was validated by management-level stakeholders and stakeholders who were involved in a major part of the project (Section 4.4).

As the ground truth validation involved stakeholders who were also involved in the survey, one may claim that their input in the validation could bias the ground truth. Nevertheless, the total number of these stakeholders was less than 10 (out of a total of 87 respondents), and they had different levels of project influence; hence, their overall influence on the requirements prioritization in StakeRare was low. In addition, disagreements with the ground truth must be backed by justifications and evidence before the ground truth was amended. The total number of amendments made from the validation was less than 5.

5.3 Data Cleaning

The responses provided by respondents during the survey were cleaned by this researcher. Additional requirements provided by the respondents were classified into their respective project objectives, synonymous requirements were merged, and statements containing more than one requirement were split. Manual merging of requirements and classification of requirements into their respective project objectives can be subjective.

The data cleaning on the RankP data set has the highest risk of being subjective among all the data sets in this work. StakeRare uses the RateP data set where an initial list of requirements is provided. RankP was a data set used solely to evaluate StakeRare. In RankP, there were no initial requirements; stakeholders provided their own requirements, which were then cleaned by the researchers by manually classifying the requirements into their respective project objectives.

To determine the objectiveness of the classification, a group of 16 Master of Science students from the UCL Department of Computer Science were requested to classify the raw text provided by the respondents into the relevant project objectives. The students' classifications were then compared to the classification by this researcher. These students were enrolled in the Systems Requirements Engineering course[13] during the time of this study. Their familiarity with the RALIC project comes from the brief description of the project in the StakeNet paper [89] and the use of their own access cards. Each student was given approximately 25 requirements to classify into project objectives. The requirements are raw text as provided by the stakeholders.

The classification survey revealed that the classification in the work appears to be valid, but dependent on project knowledge and interaction with the stakeholders. The student's classification shows agreement with the classification in this work, with a 68 percent match. The discrepancy was partly due to the students' lack of project knowledge about RALIC. The students found the classification easy, but felt that they required more knowledge about RALIC to be certain about some classifications.

5.4 Evaluation of the Collaborative Filtering Algorithm

StakeRare uses collaborative filtering to recommend relevant requirements to a stakeholder. The evaluation uses mean absolute error, which measures the deviation between a predicted rating and the actual rating provided by the stakeholder, rather than the relevance of the recommended requirements. In addition, MAE may be less appropriate for tasks where a ranked result is returned to the user, who then only views items at the top of the ranking. The accuracy of predictions for items that the system correctly knows the user will have no interest in are less important, but using MAE, the predictions for all items are evaluated equally [80], [91].

Nevertheless, MAE is used to evaluate this work as it captures the quality of the collaborative filtering system because more accurate predictions of ratings generally correspond to more relevant recommendations [80]. In addition, MAE is a standard measure for evaluating predictive accuracy in collaborative filtering widely used in the field (e.g., for predicting the accuracy of movie ratings) [80], [91], [103], [104], [105], [106]. Finally, measures that are more appropriate to evaluate a ranked list of recommendations have their own drawbacks (e.g., the half-life utility metric is less standard, and different researchers can use significantly different values making it difficult to compare results and easy to manipulate) [91]. More accurate means to evaluate the relevance of the recommendations, such as interviewing stakeholders to find out the relevance of the recommendations, should be investigated.

5.5 Respondent Feedback

Due to social niceties, the respondents' feedback (e.g., their ratings on the ease of use of StakeRare, and feedback on the accuracy of StakeRare's prioritization) may be positively biased. Nevertheless, these stakeholders have little incentive to make socially desirable remarks, and they have been quite frank in pointing out deficiencies in the methods (e.g., the arithmetic exercise in PointP was distracting and they disliked rating all the requirements). In addition, it was made clear to the stakeholders that the main objective of their feedback was to improve the work.

The respondents were requested to evaluate each elicitation method sequentially. For a more effective comparison among the three elicitation methods, the stakeholders should evaluate the methods all at once at the end of the survey, for example, using pairwise comparison among the elicitation methods.

6 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 large project in terms of number of stakeholders and stakeholder roles, it is not large in terms of the number of elicited requirements. Step 3 of StakeRare aims to predict the requirements the stakeholders 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 (e.g., hundreds or thousands).

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 crowdsourcing the stakeholders to clean the data, enabling the stakeholders to comment on the requirements engineers' data cleaning, or using natural language processing to identify similar requirements [109].

The evaluation also highlighted limitations in the StakeRare method that should be addressed in future work as follows:

- The quality of the requirements identified by StakeRare may depend on the initial set of requirements, especially in projects where stakeholders are less aware of their requirements. Future work should investigate the effect of the initial requirements on the result, for example, by using a control project

13. http://www.cs.ucl.ac.uk/teaching/syllabus/mscsse/gs01.htm.

with a lower quality set of initial requirements. The initial requirements in StakeRare are identified using the existing elicitation methods discussed in the background section. Future work can also explore ways to improve the quality of the initial requirements, for example by selecting suitable existing elicitation methods for different projects.

- The quality of the requirements depends on the stakeholders' response. Future work should investigate methods to encourage their response. StakeRare was designed for projects where stakeholders are likely to use the resulting system. In such projects, it is likely that stakeholders are better motivated to respond. In projects where stakeholders are not obliged to use the system or are unclear about their requirements, focus groups and workshops may be more suitable to encourage stakeholders to articulate and discover requirements. Hence, future work should investigate the types of projects suitable for StakeRare to be used.
- StakeRare assumes that stakeholders provide recommendations and ratings honestly. However, malicious stakeholders may provide responses for their personal benefit, such as recommend "nonstakeholders," exclude some stakeholders, or manipulate the requirements ratings. This, in turn, affects the quality of stakeholders and requirements returned by the method. Nevertheless, the betweenness centrality measure makes it more difficult for stakeholders to manipulate their influence because it considers the whole network in the prioritization (i.e., to manipulate their influence in the project, the stakeholders have to influence the other stakeholders' recommendations). Still, future work should develop more sophisticated methods that account for malicious stakeholders who manipulate recommendations and ratings for personal gains.
- StakeRare's rating propagation in Step 2 makes assumptions that may not always be true. Future work should investigate alternate methods of propagation and the effect of these methods on the results.
- Requirements that stakeholders actively do not want are rated with an X. Future work should investigate 1) enabling stakeholders to indicate the levels they do not want a requirement (e.g., -1 to -5, where -1 is mildly do not want and -5 is strongly do not want), 2) incorporating negative ratings into requirements prioritization, and 3) integrating conflict negotiation methods into StakeRare.
- In StakeRare, requirements are weighted by the stakeholders' total influence over the project life cycle. To improve the accuracy of prioritization, future work should consider the following factors:
  - Stakeholders have different influence across different issues, such as funding, development, and usage.
  - A stakeholder's influence in a project may change over time.
  - Requirements and their importance in the project change over time.
  - Knowledge about the implementation cost of each requirement may influence the stakeholders' rating on the requirement.
  - The dependencies among requirements may influence the prioritization. For example, if an essential requirement depends on a trivial requirement to be realized, then the trivial requirement deserves high priority.
- During the evaluation, various lessons were learned from analyzing different elicitation methods. For example, RateP allowed stakeholders to rate as many requirements as they wanted, which caused them to include many "nice to have" features, stakeholders find it difficult to point out requirements they actively do not want in RankP where no requirements were provided, and PointP had the highest precision as limited points encouraged stakeholders to only suggest requirements they really needed. Future work should analyze these insights systematically and investigate their suitability for different kinds of projects and stakeholders.
- The diversity of results for the different elicitation methods suggests that future work should investigate the combination of elicitation methods to produce better results. In this work, only three elicitation methods (i.e., RateP, RankP, and PointP) are compared. Future work should also conduct comparisons with more elicitation methods.

Finally, changes in the stakeholders' recommendations (e.g., modification of salience value or new recommendations) or changes in the requirements' ratings (e.g., modification or addition of ratings) mean that the requirements have to be reprioritized. StakeRare's effectiveness in supporting requirements elicitation can be significantly improved by automating the requirements engineer's task, such as face-to-face surveys with stakeholders, data entry, and requirements prioritization. StakeSource,[14] a software tool described in previous work [110], has been developed to automate Step 1 of StakeRare. StakeSource *crowdsources*[15] the stakeholders themselves for recommendations about other stakeholders and aggregates their answers using social network analysis. StakeSource has been used in real projects by practitioners. Tool support for the other StakeRare steps, such as rating collection, automated requirements prioritization, and collaborative filtering computations, has also been developed [113].

7 Conclusion

In large software projects, requirements elicitation tends to be beset by three problems: information overload, inadequate stakeholder input, and biased prioritization of requirements.

The main contribution of the work is the development of the StakeRare method, which supports requirements elicitation in large software projects. The method is one of the first applications of social networks and collaborative

14. http://www.cs.ucl.ac.uk/research/StakeSource/.

15. Crowdsourcing is a concept that harnesses the knowledge contained in diverse communities of people [111], [112]. In this case, StakeSource harnesses the knowledge of stakeholders to identify and prioritize stakeholders.

filtering to identify and prioritize stakeholders and their requirements.

A second important contribution of the work is the extensive empirical evaluation of the methods using a real large-scale software project. This work pioneered three significant forms of evaluation: the comparison with existing elicitation methods used in the project, the comparison with the ground truth built from postproject 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 elicitation research. Approximately 200 face-to-face interviews were conducted with the project stakeholders, and more than 1,000 pages of project documentation were reviewed.

Using these substantial data, the work has demonstrated that social networks and collaborative filtering provide effective support for requirements elicitation in large-scale software projects.

In a broader context, this work proposes a new methodology in requirements elicitation that shifts the emphasis from requirements elicitation by the requirements engineers to a collaborative approach in which a representative sample of stakeholders has a say. Doing so reduces the requirements engineers' workload and the likelihood of omitting stakeholders and their requirements. This methodology for supporting requirements elicitation is one of the first scalable solutions for future large projects. Using methods such as StakeRare, it is hoped that one day software projects will no longer fail from information overload, inadequate stakeholder input, and biased prioritization of requirements.

ACKNOWLEDGMENTS

The authors wish to thank the RALIC stakeholders for their survey participation, Peter Bentley for his machine learning expertise and for reading the numerous revisions of this paper, Jun Wang for his advise on collaborative filtering, Daniele Quercia and Neal Lathia for their idea of using collaborative filtering in this context, Emmanuel Letier for his requirements engineering advice, and his Systems Requirements Engineering students for participating in the requirements classification survey, and the anonymous reviewers for their feedback on the paper.

REFERENCES

[1] B. Nuseibeh and S. Easterbrook, "Requirements Engineering: A Roadmap," *Proc. Conf. Future of Software Eng.*, pp. 35-46, 2000.
[2] 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.
[3] P. Zave, "Classification of Research Efforts in Requirements Engineering," *ACM Computing Surveys*, vol. 29, no. 4, pp. 315-321, 1997.
[4] 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.
[5] R.N. Charette, "Why Software Fails," *IEEE Spectrum*, vol. 42, no. 9, pp. 42-49, Sept. 2005.
[6] D.C. Gause and G.M. Weinberg, *Exploring Requirements: Quality before Design.* Dorset House Publishing Company, Inc., 1989.
[7] I. Alexander and S. Robertson, "Understanding Project Sociology by Modeling Stakeholders," *IEEE Software*, vol. 21, no. 1, pp. 23-27, Jan./Feb. 2004.
[8] 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.
[9] 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.
[10] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide.* John Wiley & Sons, Inc., 1997.
[11] B.H.C. Cheng and J.M. Atlee, "Research Directions in Requirements Engineering," *Proc. Conf. Future of Software Eng.*, pp. 285-303, 2007.
[12] S.L. Lim, "Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation," PhD thesis, Univ. of New South Wales, //www.cs.ucl.ac.uk/staff/S.Lim/phd/thesis_soolinglim.pdf, 2010.
[13] A.J. Albrecht, "Measuring Application Development Productivity," *Proc. Joint SHARE, GUIDE, and IBM Application Development Symp.*, pp. 83-92, 1979.
[14] G.C. Low and D.R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process," *IEEE Trans. Software Eng.*, vol. 16, no. 1, pp. 64-71, Jan. 1990.
[15] C. Jones, "Patterns of Large Software Systems: Failure and Success," *Computer*, vol. 28, no. 3, pp. 86-87, Mar. 1995.
[16] C.F. Kemerer, "Reliability of Function Points Measurement: A Field Experiment," *Comm. ACM*, vol. 36, no. 2, pp. 85-97, 1993.
[17] D. Galin, *Software Quality Assurance: From Theory to Implementation.* Addison-Wesley, 2004.
[18] C. Jones, *Applied Software Measurement: Global Analysis of Productivity and Quality*, third ed. McGraw-Hill Osborne Media, 2008.
[19] F.P. Brooks Jr., *The Mythical Man-Month*, anniversary ed. Addison-Wesley Longman Publishing Co., Inc., 1995.
[20] M.E. Fagan, "Design and Code Inspections to Reduce Errors in Program Development," *IBM Systems J.*, vol. 15, no. 3, pp. 182-211, 1976.
[21] D.R. Jeffery and M.J. Lawrence, "An Inter-Organisational Comparison of Programming Productivity," *Proc. Fourth Int'l Conf. Software Eng.*, pp. 369-377, 1979.
[22] A.J. Albrecht and J.E. Gaffney Jr., "Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation," *IEEE Trans. Software Eng.*, vol. 9, no. 6, pp. 639-648, Nov. 1983.
[23] S. McConnell, *Rapid Development: Taming Wild Software Schedules.* Microsoft Press, 1996.
[24] S. McConnell, *Code Complete.* Microsoft Press, 2004.
[25] C. Jones, *Estimating Software Costs.* McGraw-Hill, 1998.
[26] J.J. Rakos, *Software Project Management.* Prentice Hall, 1990.
[27] M. Fischer and H. Gall, "Visualizing Feature Evolution of Large-Scale Software Based on Problem and Modification Report Data," *J. Software Maintenance and Evolution Research and Practice*, vol. 16, no. 6, pp. 385-403, 2004.
[28] C. Sauer, A. Gemino, and B.H. Reich, "The Impact of Size and Volatility on IT Project Performance," *Comm. ACM*, vol. 50, no. 11, pp. 79-84, 2007.
[29] P. Kruchten, "Architectural Blueprints—The '4+ 1' View Model of Software Architecture," *IEEE Software*, vol. 12, no. 6, pp. 42-50, Nov. 1995.
[30] M. Burstin and M. Ben-Bassat, "A User's Approach to Requirements Analysis of a Large Software System," *Proc. ACM Ann. Conf. Fifth Generation Challenge*, pp. 133-145, 1984.
[31] M. Cross, "Special Report: Public Sector IT Failures," *PROSPECT*, pp. 48-52, 2005.
[32] L. Northrop, P. Feiler, R.P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, and K. Sullivan, *Ultra-Large-Scale Systems: The Software Challenge of the Future.* Software Eng. Inst., 2006.
[33] R. Schaefer, "A Rational Theory of System-Making Systems," *ACM SIGSOFT Software Eng. Notes*, vol. 31, no. 2, pp. 1-20, 2006.
[34] P. Laurent, J. Cleland-Huang, and C. Duan, "Towards Automated Requirements Triage," *Proc. 15th IEEE Int'l Conf. Requirements Eng.*, pp. 131-140, 2007.
[35] R. Schaefer, "A Systems Analysis of Systems Integration," *ACM SIGSOFT Software Eng. Notes*, vol. 11, no. 1, 2008.
[36] C. Duan, P. Laurent, J. Cleland-Huang, and C. Kwiatkowski, "Towards Automated Requirements Prioritization and Triage," *Requirements Eng.*, vol. 14, no. 2, pp. 73-89, 2009.
[37] H. Goldstein, "Who Killed the Virtual Case File?" *IEEE Spectrum*, vol. 42, no. 9, pp. 24-35, Sept. 2005.

[38] A. Cockburn, *Writing Effective Use Cases.* Addison-Wesley Professional, 2000.

[39] S. Robertson and J. Robertson, *Mastering the Requirements Process.* Addison-Wesley Professional, 2006.

[40] A. Davis, O. Dieste, A. Hickey, N. Juristo, and A.M. Moreno, "Effectiveness of Requirements Elicitation Techniques: Empirical Results Derived from a Systematic Review," *Proc. 14th IEEE Int'l Conf. Requirements Eng.,* pp. 179-188, 2006.

[41] S. Lauesen, *Software Requirements: Styles and Techniques.* Addison-Wesley Professional, 2002.

[42] A.M. Davis, "Operational Prototyping: A New Development Approach," *IEEE Software,* vol. 9, no. 5, pp. 70-78, Sept. 1992.

[43] B. Potts, "Metaphors of Intent," *Proc. Fifth IEEE Int'l Symp. Requirements Eng.,* pp. 31-38, 2001.

[44] S.J. Andriole, *Storyboard Prototyping: A New Approach to User Requirements Analysis.* QED Information Sciences, Inc., 1989.

[45] D. Leffingwell and D. Widrig, *Managing Software Requirements: A Unified Approach.* Addison-Wesley Longman Publishing Co., Inc., 1999.

[46] I. Jacobson, *Object-Oriented Software Engineering.* ACM Press, 1991.

[47] N.A.M. Maiden, "CREWS-SAVRE: Scenarios for Acquiring and Validating Requirements," *Automated Software Eng.,* vol. 5, no. 4, pp. 419-446, 1998.

[48] A. Dardenne, A. van Lamsweerde, and S. Fickas, "Goal-Directed Requirements Acquisition," *Science of Computer Programming,* vol. 20, nos. 1/2, pp. 3-50, 1993.

[49] A. van Lamsweerde, "Goal-Oriented Requirements Engineering: A Guided Tour," *Proc. Fifth IEEE Int'l Symp. Requirements Eng.,* pp. 249-262, 2001.

[50] E.S.K. Yu, "Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering," *Proc. Third IEEE Int'l Symp. Requirements Eng.,* pp. 226-235, 1997.

[51] H. Holbrook III, "A Scenario-Based Methodology for Conducting Requirements Elicitation," *ACM SIGSOFT Software Eng. Notes,* vol. 15, no. 1, pp. 95-104, 1990.

[52] A.M. Davis, *Software Requirements: Objects, Functions, and States.* Prentice-Hall, Inc., 1993.

[53] J. Karlsson, "Software Requirements Prioritizing," *Proc. Second Int'l Conf. Requirements Eng.,* pp. 110-116, 1996.

[54] P. Berander and P. Jonsson, "Hierarchical Cumulative Voting (HCV)—Prioritization of Requirements in Hierarchies," *Int'l J. Software Eng. and Knowledge Eng.,* vol. 16, no. 6, pp. 819-849, 2006.

[55] J. Karlsson, C. Wohlin, and B. Regnell, "An Evaluation of Methods for Prioritizing Software Requirements," *Information and Software Technology,* vol. 39, nos. 14/15, pp. 939-947, 1998.

[56] J. Karlsson and K. Ryan, "A Cost-Value Approach for Prioritizing Requirements," *IEEE Software,* vol. 14, no. 5, pp. 67-74, Sept./Oct. 1997.

[57] J. Azar, R.K. Smith, and D. Cordes, "Value-Oriented Requirements Prioritization in a Small Development Organization," *IEEE Software,* vol. 24, no. 1, pp. 32-37, Jan./Feb. 2007.

[58] D. Leffingwell and D. Widrig, *Managing Software Requirements: A Use Case Approach.* Pearson Education, 2003.

[59] N.R. Mead, *Requirements Prioritization Introduction.* Software Eng. Inst., Carnegie Mellon Univ., 2006.

[60] B. Regnell, M. Höst, J.N. och Dag, P. Beremark, and T. Hjelm, "An Industrial Case Study on Distributed Prioritisation in Market-Driven Requirements Engineering for Packaged Software," *Requirements Eng.,* vol. 6, no. 1, pp. 51-62, 2001.

[61] A.M. Davis, "The Art of Requirements Triage," *Computer,* vol. 36, no. 3, pp. 42-49, Mar. 2003.

[62] B.W. Boehm and R. Ross, "Theory-W Software Project Management Principles and Examples," *IEEE Trans. Software Eng.,* vol. 15, no. 7, pp. 902-916, July 1989.

[63] J. Park, D. Port, and B. Boehm, "Supporting Distributed Collaborative Prioritization for Win-Win Requirements Capture and Negotiation," *Proc. Int'l Third World Multiconf. Systemics, Cybernetics and Informatics,* vol. 2, pp. 578-584, 1999.

[64] K. Wiegers, "First Things First: Prioritizing Requirements," *Software Development,* vol. 7, no. 9, pp. 48-53, 1999.

[65] A. Herrmann and M. Daneva, "Requirements Prioritization Based on Benefit and Cost Prediction: An Agenda for Future Research," *Proc. 16th IEEE Int'l Conf. Requirements Eng.,* pp. 125-134, 2008.

[66] A. van Lamsweerde, *Requirements Engineering: From System Goals to UML Models to Software Specifications.* John Wiley & Sons, Inc., 2009.

[67] R.L. Keeney and H. Raiffa, *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* Cambridge Univ. Press, 1993.

[68] B. Roy, *Multicriteria Methodology for Decision Aiding.* Springer-Verlag, 1996.

[69] T.F. Nas, *Cost-Benefit Analysis: Theory and Application.* Sage, 1996.

[70] Y. Akao, *Quality Function Deployment: Integrating Customer Requirements into Product Design.* Productivity Press, 2004.

[71] F. Moisiadis, "The Fundamentals of Prioritising Requirements," *Proc. System Eng., Test and Evaluation Conf.,* 2002.

[72] V. Ahl, "An Experimental Comparison of Five Prioritization Methods," master's thesis, School of Eng., Blekinge Inst. of Technology, 2005.

[73] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications.* Cambridge Univ. Press, 1994.

[74] L.A. Goodman, "Snowball Sampling," *The Annals of Math. Statistics,* vol. 32, no. 1, pp. 148-170, 1961.

[75] J. Scott, *Social Network Analysis: A Handbook.* Sage, 2000.

[76] R.A. Hanneman and M. Riddle, *Introduction to Social Network Methods.* Univ. of California, Riverside, 2005.

[77] C. Damian, S. Marczak, and I. Kwan, "Collaboration Patterns and the Impact of Distance on Awareness in Requirements-Centred Social Networks," *Proc. 15th IEEE Int'l Conf. Requirements Eng.,* pp. 59-68, 2007.

[78] D. Damian, I. Kwan, and S. Marczak, "Requirements-Driven Collaboration: Leveraging the Invisible Relationships between Requirements and People," *Collaborative Software Engineering,* Springer, 2010.

[79] E. Goldberg, D. Nichols, B.M. Oki, and D. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Comm. ACM,* vol. 35, no. 12, pp. 61-70, 1992.

[80] J. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative Filtering Recommender Systems," *The Adaptive Web: Methods and Strategies of Web Personalization,* pp. 291-324, Springer, 2007.

[81] F. Linden, B. Smith, and J. York, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering," *IEEE Internet Computing,* vol. 7, no. 1, pp. 76-80, Jan./Feb. 2003.

[82] N. Lathia, "Computing Recommendations with Collaborative Filtering," *Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling,* Information Science Reference, 2008.

[83] T. Segaran, *Programming Collective Intelligence.* O'Reilly Media, 2007.

[84] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An Algorithmic Framework for Performing Collaborative Filtering," *Proc. 22nd Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 230-237, 1999.

[85] R.M. Bell and Y. Koren, "Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights," *Proc. Seventh IEEE Int'l Conf. Data Mining,* pp. 43-52, 2007.

[86] C. Castro-Herrera, J. Cleland-Huang, and B. Mobasher, "Enhancing Stakeholder Profiles to Improve Recommendations in Online Requirements Elicitation," *Proc. 17th IEEE Int'l Conf. Requirements Eng.,* pp. 37-46, 2009.

[87] D. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "A Recommender System for Requirements Elicitation in Large-Scale Software Projects," *Proc. ACM Symp. Applied Computing,* pp. 1419-1426, 2009.

[88] R.K. Mitchell, B.R. Agle, and D.J. Wood, "Toward a Theory of Stakeholder Identification and Salience: Defining the Principle of Who and What Really Counts," *Academy of Management Rev.,* vol. 22, no. 4, pp. 853-886, 1997.

[89] S.L. Lim, D. Quercia, and A. Finkelstein, "StakeNet: Using Social Networks to Analyse the Stakeholders of Large-Scale Software Projects," *Proc. 32nd ACM/IEEE Int'l Conf. Software Eng.,* vol. 1, pp. 295-304, 2010.

[90] M.F. Triola, *Elementary Statistics.* Addison Wesley, 1992.

[91] J.L. Herlocker, J.A. Konstan, L.G. Terveen, and J.T. Riedl, "Evaluating Collaborative Filtering Recommender Systems," *ACM Trans. Information Systems,* vol. 22, no. 1, pp. 5-53, 2004.

[92] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann Publishers, Inc., 2005.

[93] J. McManus and T. Wood-Harper, "Understanding the Sources of Information Systems Project Failure," *Management Services,* vol. 51, no. 3, pp. 38-43, 2007.

[94] The Standish Group, *The CHAOS Report in the Standish Group Report,* p. 8, 1994.

[95] The Standish Group, *CHAOS Summary 2009 in The Standish Group Report,* 2009.

[96] T.R. Lindlof and B.C. Taylor, *Qualitative Communication Research Methods.* Sage, 2002.

[97] C. Anderson, *The Long Tail: Why the Future of Business Is Selling Less of More.* Hyperion Books, 2008.

[98] J. Yu and H. Cooper, "A Quantitative Review of Research Design Effects on Response Rates to Questionnaires," *J. Marketing Research,* vol. 20, no. 1, pp. 36-44, 1983.

[99] P. Zhang, "Model Selection via Multifold Cross Validation," *The Annals of Statistics,* vol. 21, no. 1, pp. 299-313, 1993.

[100] R. Kohavi, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection," *Proc. Int'l Joint Conf. Artificial Intelligence,* pp. 1137-1145, 1995.

[101] U.M. Braga-Neto and E.R. Dougherty, "Is Cross-Validation Valid for Small-Sample Microarray Classification?" *Bioinformatics,* vol. 20, no. 3, pp. 374-380, 2004.

[102] S. Alag, *Collective Intelligence in Action.* Manning Publications, 2008.

[103] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-Based Collaborative Filtering Recommendation Algorithms," *Proc. 10th Int'l Conf. World Wide Web,* pp. 285-295, 2001.

[104] R. Jin, J.Y. Chai, and L. Si, "An Automatic Weighting Scheme for Collaborative Filtering," *Proc. 27th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 337-344, 2004.

[105] G.R. Xue, C. Lin, Q. Yang, W.S. Xi, H.J. Zeng, Y. Yu, and Z. Chen, "Scalable Collaborative Filtering Using Cluster-Based Smoothing," *Proc. 28th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 114-121, 2005.

[106] J. Wang, A.P. De Vries, and M.J.T. Reinders, "Unifying User-Based and Item-Based Collaborative Filtering Approaches by Similarity Fusion," *Proc. 29th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval,* pp. 501-508, 2006.

[107] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *Philosophical Magazine Series 6,* vol. 2, no. 11, pp. 559-572, 1901.

[108] T. Howley, M.G. Madden, M.L. O'Connell, and A.G. Ryder, "The Effect of Principal Component Analysis on Machine Learning Accuracy with High Dimensional Spectral Data," *Knowledge Based Systems,* vol. 19, no. 5, pp. 209-222, 2006.

[109] K. Zachos, N. Maiden, X. Zhu, and S. Jones, "Discovering Web Services to Specify More Complete System Requirements," *Proc. 19th Int'l Conf. Advanced Information Systems Eng.,* pp. 142-157, 2007.

[110] S.L. Lim, D. Damian, and A. Finkelstein, "StakeSource: Harnessing the Power of Crowdsourcing and Social Networks in Stakeholder Analysis," *Proc. 32nd ACM/IEEE Int'l Conf. Software Eng.* vol. 2, pp. 239-242, 2010.

[111] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations.* Doubleday Books, 2004.

[112] J. Howe, *Crowdsourcing: Why the Power of the Crowd Is Driving the Future of Business.* Three Rivers Press, 2009.

[113] S.L. Lim, D. Damian, and A. Finkelstein, "StakeSource2.0: Using Social Networks of Stakeholders to Identify and Prioritise Requirements," *Proc. 33rd ACM/IEEE Int'l Conf. Software Eng.,* 2011.

Soo Ling Lim received the bachelor's degree in software engineering with first class honors from the Australian National University in 2005. She is working toward the PhD degree in the School of Computer Science and Engineering, The University of New South Wales in Sydney, Australia. She is also a visiting researcher at the Department of Computer Science, University College London. She also won the Australian Computer Society prize and the Dean's prize. Before her PhD work, she worked as an SAP consultant at the Computer Sciences Corporation and as a software engineer at CIC Secure in Australia. Her research interests are in the area of requirements engineering, specifically in the areas of stakeholder analysis, requirements elicitation, prioritization, and modeling, and requirements change management. She is a member of the British Computer Society.

Anthony Finkelstein received the BEng degree in systems engineering, the MSc degree in systems analysis, and the PhD degree in design theory. He is a professor of software systems engineering and the dean of the Faculty of Engineering Sciences at University College London. Formerly, he was a professor of computer science at The City University, London, and the head of the Department of Computer Science. Prior to that, he was a member of the academic staff at Imperial College of Science, Technology & Medicine. His research interests are in the area of software systems engineering and, in particular, in requirements engineering. He has contributed to software specification methods, software development processes, tool, and environment support for software development. He has published more than 200 papers in these areas and held research grants totaling in excess of £20m. In 2003, he was a winner of the prestigious International Conference on Software Engineering "most influential paper" prize for work on "viewpoints" and in 2004 was a winner of the Requirements Engineering "most influential paper" prize for work on traceability. In 2005, he was a member of the winning team of the first Times Higher Education Supplement "Research Project of the Year." In 2009, he received the Oliver Lodge Medal of the Institution of Engineering and Technology for outstanding achievement in the area of Information Technology. In 2010, he won the special award conferred by the International Conference on Software Engineering for his "outstanding contribution in designing the highly influential Future of Software Engineering track and editing the highly cited volume with the collection of papers presented in the track." He is a chartered engineer, a fellow of the IEE and BCS. He is a founding member of IFIP WG 2.9 Software Requirements Engineering. He is a member of the IEEE.