Using Genetic Algorithms to Search for Key Stakeholders in Large-Scale Software Projects

Soo Ling Lim
s.lim@cs.ucl.ac.uk
Department of Computer Science, University College London, United Kingdom

Mark Harman
m.harman@cs.ucl.ac.uk
Department of Computer Science, University College London, United Kingdom

Angelo Susi
susi@fbk.eu
Fondazione Bruno Kessler, Trento, Italy

ABSTRACT
Large software projects have many stakeholders. In order for the resulting software system and architecture to be aligned with the enterprise and stakeholder needs, key stakeholders must be adequately consulted and involved in the project. This work proposes the use of genetic algorithms to identify key stakeholders and their actual influence in requirements elicitation, given the stakeholders' requirements and the actual set of requirements implemented in the project. The proposed method is applied to a large real-world software project. Results show that search is able to identify key stakeholders accurately. Results also indicate that many different good solutions exist. This implies that a stakeholder has the potential to play a key role in requirements elicitation, depending on which other stakeholders are already involved. This work demonstrates the true complexity of requirements elicitation – all stakeholders should be consulted, but not all of them should be treated as key stakeholders, even if they appear to be significant based on their role in the domain.

INTRODUCTION
Large software projects are complex. They involve thousands, or even hundreds of thousands of stakeholders (Cleland-Huang & Mobasher, 2008). These stakeholders range from customers who pay for the software system, to users who interact with the system, developers who build and maintain the system, and legislators who impose rules on the development and operation of the system. They have different and sometimes conflicting requirements. They also have different degrees of influence on the project (Glinz & Wieringa, 2007).

In order for the resulting software system and architecture to be aligned with the enterprise and stakeholder needs, the stakeholders' voices must be heard (Lim & Finkelstein, 2011). In addition, key stakeholders – stakeholders who contribute significant and valuable information to the requirements elicitation process – must be identified and engaged throughout the project (Gause & Weinberg, 1989; Glinz & Wieringa, 2007). Nevertheless, stakeholders can lack knowledge, interest or time to be

adequately involved in the project (Alexander & Robertson, 2004). Previous studies have found that most developers experience difficulty identifying and engaging with key stakeholders (Alexander & Robertson, 2004; Gause & Weinberg, 1989). As a result, software project often fail due to the omission of these stakeholders and lack of input from them (Lim, 2010).

This paper focuses on discovering key stakeholders who influence the outcome of requirements elicitation. In large projects, this is a known challenge (Cheng & Atlee, 2007). To consider all stakeholders would be impractical and often infeasible. However, considering only a subset of stakeholders is risky. As requirements are elicited from stakeholders, involving the wrong set of people gives rise to incomplete requirements, which leads to the development of the wrong product. Previous work proposes the use of social network analysis – a technique increasingly being used in software engineering – to identify and prioritise stakeholders (Lim, Quercia, & Finkelstein, 2010). However, such approaches make the assumption that the stakeholders' influence in the project is determined solely by their position in the network, which does not always hold (Lim et al., 2010). Exactly how to determine if an individual is a key stakeholder is not well understood.

This work proposes the use of genetic algorithms to identify key stakeholders and their *actual* influence in a project, given the stakeholders' requirements priorities and the actual priorities of the requirements that are implemented in the project. The aim of the work is to increase the understanding of key stakeholders using search-based techniques. In addition, the proposed method can be used in an incremental development process in which the analyst applies the method to the data collected from the previous increment, and use the results to inform elicitation in the current increment. This is one of the first applications of search-based techniques to analyse software project stakeholders. Stakeholder analysis is an ideal candidate for genetic algorithms, because the direct calculation of influence is not feasible as the search space is vast, noisy and multimodal, yet the evaluation of candidate solutions is fast (Goldberg, 1989). The proposed method is applied to data collected from a large real-world software project. The results are analysed using quantitative data collected from project documentation and stakeholder interviews.

The rest of the paper is organised as follows. The next section describes the background and related work, and the section after that introduces the proposed search-based technique for stakeholder analysis. The evaluation section describes the research questions, the large-scale software project used in the study, and the experiments. It also reports the results of the study. The final sections discuss future work and conclude.

BACKGROUND

Social Networks for Stakeholder Analysis

In requirements elicitation, recent work proposed the use of social network analysis to identify and prioritise stakeholders in large software projects (Lim et al., 2010). A web-based tool has been developed to support the method and has been used in more than ten software projects (Lim, Damian, & Finkelstein, 2011). In this method, stakeholders are asked to recommend other stakeholders who should be involved in the project. A recommendation is a triple:

<stakeholder name, stakeholder role, salience>

where stakeholder role is the stakeholder's position or customary function in the project (Sharp, Galal, & Finkelstein, 1999), salience is a number on an ordinal scale (e.g., 1–5) indicating the stakeholder's influence in the project (e.g., 1 is low influence and 5 is high influence), as perceived by the stakeholder who made the recommendation. An example recommendation from Bob is <Alice, data protection officer, 4>.

The recommendations are used to build a social network in which the nodes are the stakeholders and the links are their recommendations. Then, various social network measures (Table 1) are used to prioritise the stakeholders. Each social network measure produces a score for each stakeholder. Stakeholders with the same roles are grouped together. Roles are ranked based on the highest scores of their constituent stakeholders, and within roles, stakeholders are ranked based on their scores (Lim et al., 2010). The output is a prioritised list of stakeholder roles, and for each role, a prioritised list of stakeholders, ordered by decreasing score. The previous work has found that the betweenness centrality measure produced the most accurate result (Lim et al., 2010), and subsequent previous work has used the measure to weight stakeholders for requirements prioritisation (Lim & Finkelstein, 2011). This existing method is used as the baseline approach in this work for comparison purposes.

Table 1. Social Network Measures

Measure	**Prioritisation of Stakeholder *S* in a Network (Lim et al., 2010)**	**Assumptions**
Betweenness centrality (Brandes, 2001)	Sums the number of shortest paths between pairs of stakeholders that pass through *S*.	Stakeholders who are widely recommended by disparate groups of stakeholders are more influential.
Load centrality (Brandes, 2008)	Sums the amount of information passing through *S*.	The links between stakeholders represent information flow, and stakeholders with more information flowing through them are more influential.
Closeness centrality (Scott, 2000)	Sums the inverse average shortest-path distance from *S* to all reachable stakeholders.	The faster it is for stakeholders' recommendation to reach *S* the more influential *S* is.
PageRank (Page, Brin, Motwani, & Winograd, 1999)	A stakeholder strongly recommended by many highly ranked stakeholders is ranked higher, and the recommendations of a highly ranked stakeholder have more weight, which, in turn, raises the ranking of their recommended stakeholders.	Stakeholders who receive more recommendations are more influential and their recommendations are more accurate.
Degree centrality (Scott, 2000)	Ranks *S* using the number of recommendations that *S* makes and receives.	Stakeholders who make and receive a lot of recommendations are more influential.
In-degree centrality (Scott, 2000)	Ranks *S* using the number of recommendations that *S* receives.	Stakeholders who receive a lot of recommendations are more influential.
Out-degree centrality (Scott, 2000)	Ranks *S* using the number of recommendations that *S* makes.	Stakeholders who make a lot of recommendations are more influential.

Social network analysis has been widely used in software engineering. For example, Lopez-Fernandez, Robles, and Gonzalez-Barahona (2004) used social networks to analyse the information in software revision control repositories; Damian, Marczak, and Kwan (2007) used social network analysis to study collaboration, communication, and awareness among project team members. Previous work has also used social network analysis to predict software failure. For example, Zimmermann and Nagappan (2008) used social networks to predict programs that are more likely to have defects. Wolf, Schroter, Damian, and Nguyen (2009) used social network analysis to predict whether a software integration will fail based on

the communication structures of software project teams. Meneely, Williams, Snipes, and Osborne (2008) modelled developer collaboration relationships using social networks and showed that it can help to predict software failures at the file level.

Search-Based Software Engineering

Search-based software engineering is a field that applies search-based optimisation techniques to address software engineering problems. A wide range of optimisation techniques has been used, such as local search, simulated annealing, genetic algorithms, and genetic programming, as reviewed in (Harman, 2007). These techniques have been applied to various software engineering activities such as requirements analysis, project management, software refactoring, test data generation, and bug fixing, as reviewed in (Harman, 2007). A common characteristic in these works is that perfect solutions are impractical or impossible in many software engineering problems (Harman, 2007). As such, metaheuristic search-based optimisation techniques are used to search for good solutions. The search is guided by a fitness function, which serves to differentiate good solutions from poor ones. This shifts the emphasis from solution construction to solution description, and the researcher defines what is required rather than how the solution should be constructed. The search-based methods are then left to generate creative solutions for the defined fitness function.

This work uses a genetic algorithm (GA) as the search-based technique. A GA is an algorithm that mimics biological evolution as a problem-solving strategy (Goldberg, 1989). Given a specific problem, the input to the GA is a set of potential solutions to that problem, and a metric called a fitness function that allows each candidate to be quantitatively evaluated. Candidate solutions are often randomly generated, but they can also be known solutions, and the aim of the GA is to improve them. The algorithm is as follows:

1. Randomly generate an initial population
2. Compute and save the fitness for each individual in the current population
3. Define selection probabilities for each individual in the current population so that the selection probabilities is proportional to the fitness for each individual
4. Generate a new population by probabilistically selecting individuals from the current population to produce offspring via genetic operators
5. Repeat Step 2 until the maximum number of generations is reached or until an acceptable fitness score is obtained

In requirements engineering, the application of search-based techniques focuses on requirements optimisation and prioritisation. For example, in software release planning, Greer and Ruhe (2004) used genetic algorithm based approaches to optimise requirements allocation, and assess and optimise the extent to which the ordering conflicts with stakeholder priorities. Tonella, Susi, and Palma (2010) proposed prioritising requirements using interactive genetic algorithms and constraint handling. Zhang (2010) used multi-objective search-based techniques to optimise requirements in different contexts, such as balancing requirements and resources, and balancing the extent of requirements satisfaction for each stakeholder. Finkelstein, Harman, Mansouri, Ren, and Zhang (2009) proposed using search to analyse the tradeoffs between different stakeholders' notions of fairness in requirements allocation, where there are multiple stakeholders with potentially conflicting requirement priorities and different views of what constitutes a fair solution.

Despite the recent interest in search-based requirements optimisation, little work has been conducted on using search-based techniques to analyse stakeholders. The only previous work related to this topic is the recent work which explored the use of evolutionary computation to analyse the relationship between stakeholders' social networks and their involvement in the project (Lim & Bentley, 2011). The work builds different model social networks to represent various types of stakeholder activity in a project.

Then, it uses Cartesian Genetic Programming to correlate the social network of a real software project against each model.

SEARCHING FOR KEY STAKEHOLDERS

This work proposes the use of search as a method to identify key software project stakeholders. Given:

(1) a list of requirements and corresponding requirement priorities for each stakeholder,
(2) the actual priorities of the list of requirements implemented in the project (based on post project knowledge), and
(3) the assumption that more influential stakeholders are more likely to make the priorities of the project requirements resemble their personal priorities,

this work proposes to use a genetic algorithm (GA) to search for the real influence of each stakeholder in the project.

The work assumes that the project team has used existing requirements elicitation methods (e.g., use cases or goal modelling (van Lamsweerde, 2009)) to produce a set of requirements for the project. Requirements are defined at different levels of abstraction, and a high-level requirement can be refined into several more specific requirements (van Lamsweerde, 2009). As such, the requirements are organised in a hierarchy (Figure 1). Achieving all the lower-level requirements belonging to the same parent requirement means that the parent requirement is achieved. An example hierarchy of requirements for an access control system can be seen in Figure 2.

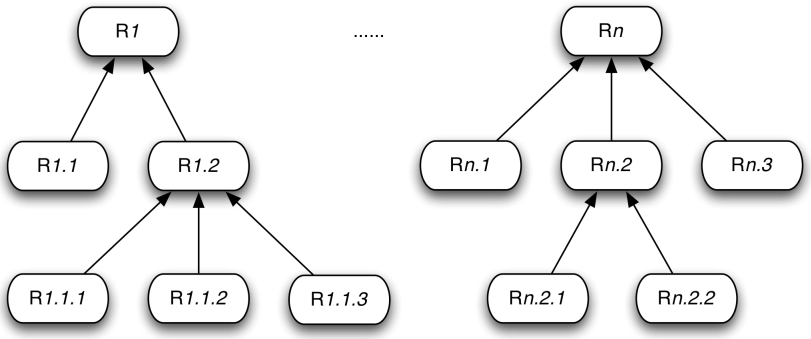

Figure 1. Hierarchy of requirements.

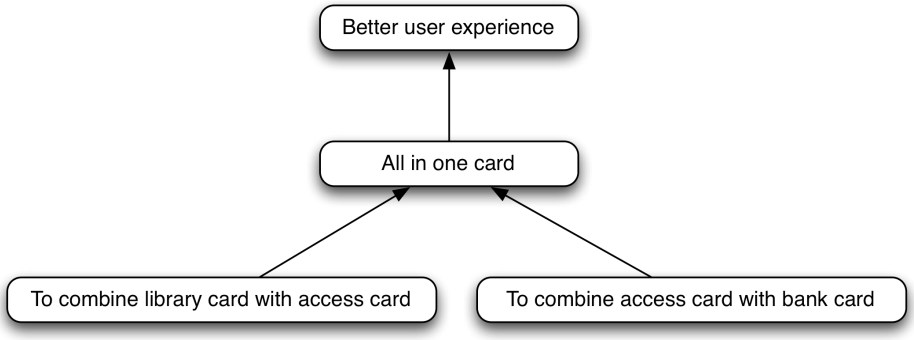

Figure 2. Example hierarchy of requirements.

Each stakeholder assigns ratings to the set of requirements identified by the project team. A rating is a number on an ordinal scale (e.g., 1–5) reflecting the importance of the requirement to the stakeholder (e.g., 1 means that the requirement is not very important to the stakeholder; 5 means that the requirement is very important). Different stakeholders can have different ratings on the same requirements (Greer &

Ruhe, 2004; Lim & Finkelstein, 2011). The set of requirements identified by the project team may be incomplete. As such, stakeholders can also provide other requirements not already in the set and rate the requirements. Stakeholders are asked to rate requirements rather than rank them, because previous work has shown that large projects can have hundreds of requirements, and stakeholders experienced difficulty providing a rank order of requirements when there are many requirements (Lim & Finkelstein, 2011).

Each requirement is assigned an *Importance* value, calculated using Equation 1.

$$Importance_R = \sum_{i=1}^{n} ProjectInfluence_i \times r_i \qquad \textbf{(Equation 1)}$$

where $ProjectInfluence_i$ is a value denoting stakeholder i's influence in the project, 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 (Lim & Finkelstein, 2011). (In the existing method, the *ProjectInfluence* value is calculated using social networking measures of stakeholder recommendation networks sorted by the stakeholders' role in the project, as described in the background section, with details of the calculation available in Lim and Finkelstein (2011).)

The requirements are ranked based on their *Importance* value, whereby requirements with higher *Importance* values are ranked higher. In assigning the ranks, fractional ranking (also known as "1 2.5 2.5 4" ranking) is used such that if a tie in ranks occurs, the mean of the ranks involved is assigned to each of the tied items. The requirements are prioritised within their hierarchy, so that the output is a ranked list of high-level requirements, for each high-level requirement, a ranked list of requirements, and for each requirement, a ranked list of specific requirements.

Using this method, the project team arrives with a prioritised list of requirements. Nevertheless, the *ProjectInfluence* values are an *approximation* of the stakeholders' real influence in the project, especially in large projects where no individuals have the global perspective (Cleland-Huang & Mobasher, 2008). Errors in these values produce incorrect prioritisation of the requirements. Throughout the project, incorrectly prioritised requirements will surface, and be rectified as the project continues. For example, stakeholders who are overlooked will voice their requirements during the project, and in the worst case, after the system has been deployed (Lim & Finkelstein, 2011). Hence, after the system has been deployed, post project knowledge reveals the actual priority of the requirements in the project. This actual prioritisation of requirements project is known as the ground truth and can be collated from project documentation (in a well-documented project) (Lim, 2010).

With the ground truth and the stakeholders' ratings on the requirements, the stakeholders' actual *ProjectInfluence* can be derived. In this work, key stakeholders are determined by the actual *ProjectInfluence* values – the higher the value, the more influential the stakeholders. The GA is used to evolve the *ProjectInfluence* values in order to search for good solutions, i.e., a set of *ProjectInfluence* values such that the resulting prioritised list of requirements (using Equation 1) is close to the ground truth. A simple binary-coded elitist GA with tournament selection, single point crossover and bitwise mutation is used (Bentley & Wakefield, 1997; Goldberg, 1989). The phenotype is defined as an array of weights, one for each stakeholder in the project who has provided ratings. Weights are coded in the genotype as 8 bit binary numbers, each corresponding to a positive integer with the range from 0 to 255. Using this method, key stakeholders – those who positively influence the end result of the prioritisation – will be identified, with their project influence values determined by the weights. Stakeholders whose requirements are not needed can be removed entirely by setting their weight to zero.

The fitness of the solution corresponds to the similarity between the priority of the requirements produced from the evolved *ProjectInfluence* values and their priority in the ground truth. The aim of the GA is to maximise the similarity. This similarity is determined using Spearman's rank correlation coefficient, ρ, in Equation 2.

$$\rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \qquad \textbf{(Equation 2)}$$

where n is the total number of requirements, x_i is the rank for requirement i in the prioritised list of requirements produced using the evolved *ProjectInfluence* values and y_i is the rank for requirement i in the ground truth. 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. As the requirements are organised in a hierarchy (Figure 1), the fitness is calculated differently depending on the level of hierarchy being evaluated. For the top-level requirements (R_1 to R_n in Figure 1), the fitness is simply the rank correlation for requirements R_1 to R_n (using Equation 2). For the second-level requirements ($R_{1.1}$ to $R_{n.3}$ in Figure 1), rank correlation is measured for each list of second-level requirements with the same parent (ρ_1 for $R_{1.1}$ and $R_{1.2}$ … ρ_n for $R_{n.1}$ to $R_{n.3}$). Then, the fitness is the average correlation for all the lists calculated using Equation 3.

$$\rho_{average} = \frac{\rho_1 + \rho_2 ... + \rho_n}{n} \qquad \textbf{(Equation 3)}$$

The fitness for the third-level requirements is calculated the same way as the second-level requirements. Finally, if the ground truth has a rank order, but the rank returned by the GA has equal values, ρ is mapped to zero to avoid divide by zero errors. This hierarchical based fitness measure has been used in previous work (Lim & Finkelstein, 2011).

EVALUATION

The experiments investigate the ability of the GA to search for the stakeholder weights in a real-world software project, given their ratings on the requirements and the ground truth. The evaluation asks the following research questions and aims to increase our understanding about key stakeholders.

- **RQ1: GA vs. existing method**. As mentioned in the previous section, the existing method calculates project influence using the output from betweenness centrality sorted by the stakeholders' roles in the project (Lim & Finkelstein, 2011). This measure was used in the existing method because previous work (Lim & Finkelstein, 2011; Lim et al., 2010) found it to be most accurate in prioritising stakeholder roles among the measures described in the background section. Using this method, a stakeholder with a low score may be ranked high, if he shares the same role with a stakeholder with a high score. RQ1 asks: How does the solution from the existing method compare to the solutions identified by the GA?

- **RQ2: Social network measures**. The use of social network measures sorted by stakeholder roles assumes that stakeholders' influence is determined by their role in the project (Lim & Finkelstein, 2011). RQ2 investigates the assumption by asking: How optimal are the raw outputs from social network measures (not sorted by roles) compared to the output from the existing method (sorted by roles)? Can the GA optimise the raw outputs from the social network measures further?

- **RQ3: All or nothing**. Traditional requirements engineering methods involve a small subset of stakeholders, but recent work has proposed inclusive methods that involve all stakeholders (Cleland-Huang & Mobasher, 2008). RQ3 asks: Can the GA find better results if it begins with the assumption that all stakeholders should be involved, or if it begins with no stakeholders involved?

- **RQ4: Rating effect**. Besides the project influence value, the importance value for each requirement also depends on the stakeholders' ratings. Previous work has shown that it is not always possible to elicit complete rating values from the stakeholders, and the data cleaning used to populate missing data fields may distort the stakeholders' actual intentions, causing a bias in the results (Lim & Finkelstein, 2011). RQ4 asks: How does data cleaning of rating values affect the results?

RALIC Datasets

RALIC is the acronym for Replacement Access, Library and ID Card. RALIC was a large-scale software project to replace the existing access control system at University College London and consolidate the new system with library access and borrowing. RALIC has a large and complex stakeholder base with more than 60 stakeholder groups and approximately 30,000 users. Besides students, staff and visitors who use the system, the project stakeholders include faculty and academic departments, as well as administrative divisions such as the estates and facilities division, human resource division that manages staff information, library systems, security systems, and so on. These stakeholders have different and sometimes conflicting requirements. The project duration was 2.5 years, and the system has already been deployed at UCL.

In this work, the following datasets from the RALIC project were used (Lim, 2010). The RALIC datasets are available at: http://www.cs.ucl.ac.uk/staff/S.Lim/phd/dataset.html. For reasons of privacy, the names of stakeholders were annonymised in this paper.

- **Recommendations**. The RALIC stakeholders were surveyed to collect their recommendations on other stakeholders. A total of 68 stakeholders provided recommendations and 127 stakeholders were identified. Recommendation data is used to build the social network of stakeholders.

- **Raw ratings**. The stakeholders rated a predefined list of requirements, from 0 (not important) to 5 (very important), and –1 for requirements they actively do not want. Stakeholders also added requirements not in the predefined list and rated those requirements. The requirements were organised into a hierarchy of three levels: project objectives, requirements, and specific requirements. A total of 76 stakeholders provided their ratings. All unrated requirements at any level of the hierarchy are given a rating of zero.

- **Propagated ratings**. The previous work (Lim & Finkelstein, 2011) cleaned the raw ratings data to eliminate missing data as follows. 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. 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, and if more than one lower-level requirement is rated, then the maximum rating is propagated. To answer RQ4, the raw ratings data and propagated ratings data were compared.

- **Ground truth**. This ground truth was built using post project knowledge, reflecting the list of requirements implemented in the project, prioritised based on their actual importance (Lim & Finkelstein, 2011).

Experiments

The following experiments were conducted to answer the research questions described previously, with variations made to the initial population and datasets. Variations were made in order to investigate how the initial population affects the trajectory of subsequent evolution in the search space. Experiments 1 to 3 used the propagated ratings dataset, and Experiment 4 used the raw ratings dataset.

- **Experiment 1: GA vs. existing method**. To answer RQ1, Experiment E1.1 initialised the GA with a random population; Experiment E1.2 initialised the GA with the output from the existing method (i.e., betweenness centrality sorted by stakeholder role) to investigate if the existing *ProjectInfluence* values are already optimal or whether they can be further optimised.

- **Experiment 2: Social network measures**. To answer RQ2, this experiment initialised the population with the output from the social network measures in Table 1 (not sorted by stakeholder roles): betweenness centrality (E2.1), load centrality (E2.2), closeness centrality (E2.3), PageRank (E2.4), degree centrality (E2.5), in-degree centrality (E2.6) and out-degree centrality (E2.7).

- **Experiment 3: All or nothing**. To answer RQ3, Experiment E3.1 initialised the population with zero weights and E3.2 initialised it with maximum weights.

- **Experiment 4: Rating effect**. Previous experiments used propagated ratings data. To answer RQ4, this experiment repeated Experiments 1 to 3 using the raw ratings data. The experiments were labelled E4.1, E4.2, and E4.3 respectively.

All experiments were run 20 times. The GA used a tournament size of 5 and elitism of 2 individuals. To discourage premature convergence, crossover was used with a probability of 1.0 and mutation was used with a probability of 0.1 per individual. The population size was 200 and the maximum number of generations was 1000. (The source code for the GA is written by Bentley and Wakefield (1997) and is available at: http://www.cs.ucl.ac.uk/staff/p.bentley/sourcecode.html. We added tournament and elitism to the code. The code was written in C, with typical execution times of about 10 seconds per run, on a 2.4GHz dual core MacBook.)

The termination criteria were either when the maximum number of generations was reached or when evolution obtained a perfect fitness value. The weights produced by the social network measures were normalised values between 0 and 1, while the GA produced integer values from 0 to 255. As such, the weights from the social network measures used to initialise the population in Experiments 1, 2 and 4 were scaled such that the maximum weight was equal to 100, in order to allow the GA to increase the weights if needed. In Experiment E3.2 (maximum weights), the initial weight for each stakeholder was 255.

RESULTS AND DISCUSSION

Table 2 summarises the results for the existing method and social network measures calculated in this work to provide a baseline comparison for the experiment results. For the existing method and each measure in Table 1, this table reports the correlation for the three hierarchy levels (project objectives, requirements, and specific requirements) and their average. The existing method produced the best average correlation for all three hierarchy levels ($\rho = 0.6186$) as shown in the first row of Propagated Ratings Data in Table 2. Overall, propagated data produced higher average correlation, demonstrating that propagation was able to clean the data to fit the current social network measures. Data propagation increased the correlation of objectives and requirements, with $\rho = 0.8088$ and 0.5825 respectively, but decreased the correlation for specific requirements, with $\rho = 0.4645$. For specific requirements, raw data

produced higher correlation with the ground truth with the best correlation of $\rho = 0.8105$ from the existing method (see first row of Raw Ratings Data in Table 2).

Table 2. Original Correlation on Propagated Ratings Data (P) and Raw Ratings Data (R)

Data	Measure	Objectives	Requirements	Specific Requirements	Average
P	Existing method	0.8088	0.5825	0.4645	0.6186
P	Betweenness	0.8829	0.2568	0.4267	0.5221
P	Load	0.8829	0.2449	0.4267	0.5182
P	Closeness	0.6791	0.3571	0.4518	0.4960
P	PageRank	0.7717	0.3492	0.4891	0.5367
P	Degree	0.6977	0.3460	0.5027	0.5155
P	In-degree	0.7162	0.3492	0.4916	0.5190
P	Out-degree	0.6791	0.3421	0.4518	0.4910
R	Existing method	0.6497	-0.1043	0.8105	0.4520
R	Betweenness	0.6497	-0.0741	0.7358	0.4371
R	Load	0.6497	-0.0741	0.7358	0.4371
R	Closeness	0.6497	-0.1537	0.6852	0.3937
R	PageRank	0.6688	0.1108	0.7541	0.5112
R	Degree	0.6497	0.1155	0.7412	0.5021
R	In-degree	0.6688	0.1308	0.7301	0.5099
R	Out-degree	0.6497	-0.1577	0.7296	0.4072

Table 3 summarises the results for all four experiments, reporting, for each experiment, the dataset, the initial population, and the correlation for the three hierarchy levels. The GA was always able to find better weights than those produced by the existing method and social network measures (Table 2). This indicates that the assumptions made by the social network measures to produce weights were not always valid for identifying key stakeholders, because their weights were evidently sub-optimum. The rest of this section discusses the results for each experiment.

Table 3. Results for Experiments 1 to 4 (standard deviation in brackets).

E	Data	Initial Population	Objectives	Requirements	Specific Requirements
1.1	P	Random	0.8947 (0.0187)	0.7028 (0.0224)	0.7648 (0.0543)
1.2	P	Existing method	0.9451 (0.0238)	0.7463 (0.0179)	0.7160 (0.0246)
2.1	P	Betweenness	0.9864 (0.0015)	0.8348 (0.0295)	0.8155 (0.0125)
2.2	P	Load	0.9847 (0.0068)	0.8470 (0.0300)	0.8255 (0.0052)
2.3	P	Closeness	0.8835 (0.0230)	0.7483 (0.0134)	0.7965 (0.0160)
2.4	P	PageRank	0.9581 (0.0102)	0.7844 (0.0305)	0.8055 (0.0251)
2.5	P	Degree	0.8934 (0.0143)	0.7403 (0.0178)	0.7964 (0.0141)
2.6	P	In-degree	0.9130 (0.0313)	0.7505 (0.0122)	0.8113 (0.0121)
2.7	P	Out-degree	0.9070 (0.0246)	0.7486 (0.0164)	0.8057 (0.0198)
3.1	P	Minimum (0)	0.9890 (0.0035)	0.9228 (0.0223)	0.8345 (0.0036)
3.2	P	Maximum (255)	0.7717 (0.0000)	0.6240 (0.0043)	0.6207 (0.0000)
4.1	R	Random	0.6914 (0.0011)	0.3056 (0.0237)	0.8764 (0.0079)
	R	Existing method	0.6902 (0.0000)	0.3231 (0.0099)	0.8717 (0.0212)
4.2	R	Betweenness	0.6918 (0.0011)	0.3681 (0.0113)	0.8842 (0.0173)
	R	Load	0.6934 (0.0048)	0.3710 (0.0120)	0.8830 (0.0192)
	R	Closeness	0.6902 (0.0007)	0.3279 (0.0132)	0.8621 (0.0155)
	R	PageRank	0.6903 (0.0005)	0.3256 (0.0144)	0.8779 (0.0187)
	R	Degree	0.6902 (0.0000)	0.3081 (0.0194)	0.8719 (0.0133)
	R	In-degree	0.6905 (0.0008)	0.3220 (0.0119)	0.8710 (0.0195)

	R	Out-degree	0.6902 (0.0000)	0.3232 (0.0107)	0.8663 (0.0145)
4.3	R	Minimum (0)	0.8242 (0.0000)	0.4820 (0.0530)	0.9135 (0.0078)
	R	Maximum (255)	0.6880 (0.0000)	0.1962 (0.0227)	0.8633 (0.0260)

Result 1: GA vs. Existing Method

The GA found comparative, and in some cases better solutions than the solution from the existing method (RQ1).

When the GA was initialised with a random population (E1.1), it was able to find comparatively good solutions, with correlation values up to 0.8947 (Table 3 E1.1). For specific requirements, it managed to find better solutions than when initialised with values from the existing method (E1.2). In the existing method, the allocation of project influence based on roles added the assumption that stakeholders' knowledge in the project is determined by their role in the project. Interviews with the stakeholders revealed that this was not always the case. Some stakeholders (such as Song the gym manager, Hicks the developer, Faulk the head of Research Department of Speech, Hearing and Phonetic Sciences, and Holmes the system maintainer) had marginal roles in the project, but had significant knowledge and interest about the project. The GA significantly increased their project influence weights (by a value of 100 or more out of a total of 255) to produce better overall requirements prioritisation.

Stakeholders with high project influence in objectives were not always influential in requirements and specific requirements. In E1.1, good solutions for different hierarchy levels have low correlation with one another. In E1.2, all three hierarchy levels were initialised with the same set of weights from the existing method. The GA made different corrections to the weights in the initial population depending on the hierarchy of requirements that was being measured. The correlation between the magnitude of corrections made to the stakeholders' weights for objectives, requirements and specific requirements was less than 0.1.

The GA showed that it was possible to find good sets of requirements from a large number of different combinations of stakeholders, sometimes not including those with significant roles as identified by the project team. Different runs of the same experiment produced best solutions that consisted of different combinations of stakeholders and weights. The best solutions between runs had low correlation with one another, and in the experiment initialised with a random initial population, the correlation between two best solutions are often around 0, indicating no correlation. This result suggests that in a practical application of the proposed solution, the analyst can select one of these equivalent solutions, based on the availability of the stakeholders.

Result 2: Social Network Measures

The raw outputs from the social network measures were sub-optimal but better than the output from the existing method, and the GA was able to optimise the raw outputs significantly (e.g., an improvement in correlation from 0.24 to 0.85 for load centrality requirements) (RQ2).

As mentioned in RQ1, the existing method used the output from betweenness centrality sorted by the stakeholders' roles to produce their *ProjectInfluence* values. When the GA was initialised with betweenness centrality weights that are not sorted by roles (E2.1), it produced a better correlation than when initialised with output from the existing method (E1.2), for all levels of requirements (e.g., for requirements, correlation of 0.8348 as compared to 0.7463 in Table 3). Again, this demonstrates that prioritising stakeholders by their roles in the project adds assumptions that do not always hold, hence producing a sub-optimal solution.

Among all the social network measures, the GA initialised with output from betweenness centrality (E2.1) and load centrality (E2.2) produced prioritised lists of requirements that had the highest correlation with the ground truth, with load centrality having a better correlation for requirements and specific requirements (Table 3). These two measures also had the highest original correlation before the GA was applied (Table 2). This showed that among the social network measures evaluated in this work, the assumptions made by these two measures produced stakeholder weights that were most valid for identifying key stakeholders.

Evolution corrected the assumptions made by the social network measures that did not hold for a particular stakeholder. For betweenness centrality, initial stakeholders who made recommendations but were not recommended, and stakeholders who did not make recommendations (even if they are highly recommended) received a project influence of 0. Interviews revealed that in RALIC, some stakeholders were not widely known (e.g., Rick the senior supplier of the project), but their formal roles in the project meant they had influence in the project. Other stakeholders, such as Lam the disability officer, did not make recommendations as they were unavailable during the recommendation period, and as a result, received betweenness of 0. In the search for better solutions, the GA corrected the weights for these stakeholders. For out-degree centrality (E2.7), stakeholders who did not recommend other stakeholders received an influence value of 0. The GA corrected the weights for stakeholders who did not recommend other stakeholders but voiced important requirements in the project. As such, although out-degree performed the worst before the GA was applied (Table 2), the GA was able to improve the correlation to be better than that of degree and closeness centrality (Table 3 E2.7).

Result 3: All or Nothing

The GA found the best results when it was initialised with all stakeholders having zero weights, and the worst results when it was initialised with all stakeholders having maximum weights (RQ3).

Evolving from an initial population of zero weights produced the highest correlation with the ground truth for all three hierarchy levels (Table 3 E3.1). Starting with no stakeholders meant that the GA was free to choose any set of stakeholders, and the GA was more selective. For example, in objectives, many stakeholders continued to have zero weights, and two stakeholders, librarian Reed and gym manager Song, appeared as key stakeholders in 15 out of 20 runs. During the interviews, these stakeholders revealed knowledge about the project that was not possessed by other stakeholders. For example, Reed correctly pointed out that external library users were significant stakeholders in the project.

Similar to the previous experiments, the GA selected a wide range of different stakeholders. In some runs, these stakeholders (Reed and Song) had zero weights, and other stakeholders were combined to get a good solution. This illustrated that a good set of requirements can be constructed from many different subsets of stakeholders, so the concept of who is a "key stakeholder" depends on which other stakeholders have already been identified. This finding is useful for elicitation, because some key stakeholders can be unavailable at times. Similar to previous experiments, the key set of stakeholders was different for each hierarchy level. For example, although Reed appeared in most runs as a key stakeholder for project objectives, in the 20 runs for specific requirements, he was never a key stakeholder (weight = 0 for all runs).

However, there were some stakeholders whose knowledge was very specific, and few other stakeholders had the knowledge. Hence for the requirements to be complete, those stakeholders must be involved. For example, in specific requirements, Lam appeared 20 times, but with an average weight of 0.17. This was because she was the only stakeholder who gave complete requirements regarding access control for disabled staff and students. As such, to get a good set of requirements, Lam must be considered, even though she did not have a formal or significant role in the project.

Initialising the population with all stakeholders having maximum weight (E3.2) produced the worst results for all three hierarchy levels (Table 3). This illustrated that it was not helpful to assume the requirements of all stakeholders were equally important. With all stakeholders considered as highly important, the GA found it difficult to get good solutions. For example, in specific requirements, only 3 stakeholders' weights were not reduced, but these stakeholders were marginally involved, and one reported having no interest in the project.

Result 4: Rating Effect

Data cleaning improved the correlation for project objectives and requirements, but decreased the correlation for specific requirements. Evolution was able to improve the correlation for all three hierarchy levels for the raw ratings dataset (RQ4).

The GA was able to produce higher correlations in specific requirements when using raw ratings (Table 3 E4.1 to E4.3) as compared to when using propagated ratings (Table 3 E1 to E3). Nevertheless, for objectives and requirements, propagated data was able to produce higher correlation (Table 3 E1 to E3). The assumptions behind upward propagation were more valid than downward propagation. Upward propagation assumes that if a stakeholder cares about a specific requirement, they would care equally about the parent requirement (Lim & Finkelstein, 2011). Indeed, when a stakeholder in RALIC rated a specific requirement but not its parent requirement, the parent requirement tended to be equally important to the stakeholder. Downward propagation assumes that specific requirements when unrated by the stakeholder have the same rating as their parent requirement (Lim & Finkelstein, 2011). In RALIC, when a stakeholder rated a requirement but not the specific requirement, it was less likely for the specific requirements to be equally important. For example, a stakeholder who wanted to combine all access control features into one card (requirement) did not agree with combining a bankcard with his access card (specific requirement).

Propagated ratings data enabled existing social networking measures to work more effectively, producing more accurate requirements prioritisation as compared to raw ratings data (Table 2). But the GA was able to provide more meaningful results with raw data – the key stakeholders identified by the GA were more sensible and consistent compared to those who were involved in the project.

FUTURE WORK

This study is based on data from a single project. As such there must be some caution generalising the results to other projects. Future work should conduct similar analysis on different software projects in different organisations. A tool has been developed to collect stakeholders' ratings (Lim et al., 2011), and as more projects use the tool, more data will be collected and can be analysed using the method proposed in this work. Future work should also investigate the robustness of the proposed method when many stakeholders do not provide ratings and when stakeholders provide dishonest or politically motivated ratings.

This work focuses on key stakeholders as stakeholders who contribute significant and useful knowledge to the requirements elicitation process. In a software project, there are various types of key stakeholders, such as those who influence the funding of the project and those who make decisions in the project. Future work should extend the analysis to other types of key stakeholders as well as provide a conceptual understanding of the characteristics of a key stakeholder. In addition, different methods to elicit requirements priorities from stakeholders can also affect the rating data (e.g., 100-point allocation or ranking), and should be investigated in future work.

Although the GA was always able to improve on the weights produced by the existing method and social network measures, in some runs, the improvement was minor and the GA converged prematurely (e.g.,

within 10,000 fitness evaluations). This was caused by the extensive epistasis in the representation (where two or more genes are linked, and the effect of each one partly depends on the values of the others) (Goldberg, 1989). Changing one gene may have different effects on the solution depending on the values of the other genes. Future work should address this issue by creating a more evolvable genotype representation, so that the GA can find better solutions quicker. In addition, interactive optimisation approaches could be investigated so that multiple factors that influence a stakeholder's involvement in the project can be considered and the analyst can select among the solutions with equivalent fitness. Finally, future work should also address the dependencies among the requirements. These dependencies influence the ground truth, and in turn, influence the identification of key stakeholders.

CONCLUSION

Large software projects have many stakeholders. The identification and engagement of key stakeholders is important in order to align the software system and architecture with stakeholder needs. This paper investigates the validity of the assumption behind using social network measures to identify key stakeholders. It proposes the use of search-based techniques to identify the key stakeholders and their actual influence in the requirements elicitation process, given the stakeholders' requirements and the actual set of requirements that is implemented in the project.

Results show that a good set of requirements can be constructed from many different subsets of stakeholders, so the concept of who is a "key stakeholder" depends on which other stakeholders have already been identified. Results also show the assumptions embodied in existing social network measures do not always hold, as such, they sometimes miss out key stakeholders. The work demonstrates a clear need for the development of more appropriate measures tailored to software stakeholders.

Consequently, this work demonstrates the true complexity of requirements elicitation – it is vital that all stakeholders are consulted, but it may be equally important that not all of them are treated as key stakeholders, even if they appear to be significant considering their role in the domain.

REFERENCES

Alexander, I., & Robertson, S. (2004). Understanding project sociology by modeling stakeholders. *IEEE Software, 21*(1), 23-27.

Bentley, P. J., & Wakefield, J. P. (1997). Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy & R. K. Pant (Eds.), *Soft Computing in Engineering Design and Manufacturing* (pp. 231-240): Springer Verlag London Limited.

Brandes, U. (2001). A faster algorithm for betweenness centrality. *Journal of Mathematical Sociology, 25*(2), 163-177.

Brandes, U. (2008). On variants of shortest-path betweenness centrality and their generic computation. *Social Networks, 30*(2), 136-145.

Cheng, B. H. C., & Atlee, J. M. (2007). Research directions in requirements engineering. *Proceedings of the Conference on the Future of Software Engineering*, pp. 285-303.

Cleland-Huang, J., & Mobasher, B. (2008). Using data mining and recommender systems to scale up the requirements process. *Proceedings of the 2nd International Workshop on Ultra-Large-Scale Software-Intensive Systems*, pp. 3-6.

Damian, D., Marczak, S., & Kwan, I. (2007). Collaboration patterns and the impact of distance on awareness in requirements-centred social networks. *Proceedings of the 15th IEEE International Conference on Requirements Engineering (RE)*, pp. 59-68.

Finkelstein, A., Harman, M., Mansouri, S. A., Ren, J., & Zhang, Y. (2009). A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requirements Engineering Journal, 14*(4), 231-245.

Gause, D. C., & Weinberg, G. M. (1989). *Exploring requirements: quality before design*: Dorset House Publishing Company, Inc.

Glinz, M., & Wieringa, R. J. (2007). Guest editors' introduction: stakeholders in requirements engineering. *IEEE Software, 24*(2), 18-20.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*: Addison-wesley.

Greer, D., & Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology, 46*(4), 243-253.

Harman, M. (2007). *The current state and future of search based software engineering*. Paper presented at the Proc. of the Future of Soft. Eng. (FoSE), Minneapolis, USA.

Lim, S. L. (2010). *Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation.* PhD Thesis, University of New South Wales, Australia.

Lim, S. L., & Bentley, P. J. (2011). Evolving relationships between social networks and stakeholder involvement in software projects. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 1899-1906.

Lim, S. L., Damian, D., & Finkelstein, A. (2011). StakeSource2.0: using social networks of stakeholders to identify and prioritise requirements. *Proceedings of the 33rd International Conference on Software Engineering (ICSE)*, pp. 1022-1024.

Lim, S. L., & Finkelstein, A. (2011). StakeRare: using social networks and collaborative filtering for large-scale requirements elicitation. *IEEE Transactions on Software Engineering (TSE)*, in press.

Lim, S. L., Quercia, D., & Finkelstein, A. (2010). StakeNet: using social networks to analyse the stakeholders of large-scale software projects. *Proceedings of the 32nd International Conference on Software Engineering (ICSE) -Vol. 1*, pp. 295-304.

Lopez-Fernandez, L., Robles, G., & Gonzalez-Barahona, J. M. (2004). Applying social network analysis to the information in CVS repositories. *Proceedings of the International Workshop on Mining Software Repositories (MSR)*, pp. 101-105.

Meneely, A., Williams, L., Snipes, W., & Osborne, J. (2008). Predicting failures with developer networks and social network analysis. *Proceedings of the 16th International Symposium on the Foundations of Software Engineering (FSE)*, pp. 13-23.

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). The Pagerank Citation Ranking: Bringing Order to the Web: Stanford InfoLab.

Scott, J. (2000). *Social network analysis: a handbook*: Sage.

Sharp, H., Galal, G. H., & Finkelstein, A. (1999). Stakeholder identification in the requirements engineering process. *Proceedings of the Database & Expert System Applications Workshop (DEXA)*, pp. 387–391.

Tonella, P., Susi, A., & Palma, F. (2010). Using interactive GA for requirements prioritization. *Proceedings of the 2nd Symposium on Search Based Software Engineering (SSBSE)*, pp. 57-66.

van Lamsweerde, A. (2009). *Requirements engineering: from system goals to UML models to software specifications*: John Wiley & Sons, Inc.

Wolf, T., Schroter, A., Damian, D., & Nguyen, T. (2009). Predicting build failures using social network analysis on developer communication. *Proceedings of the 31st International Conference on Software Engineering (ICSE)*, pp. 1-11.

Zhang, Y. (2010). *Multi-Objective Search-based Requirements Selection and Optimisation* PhD Thesis, King's College London.

Zimmermann, T., & Nagappan, N. (2008). Predicting defects using network analysis on dependency graphs. *Proceedings of the 30th International Conference on Software Engineering (ICSE)*, pp. 531-540.

KEYWORDS

Stakeholders

Stakeholders are individuals or groups that can influence, or be influenced by a software project.

Stakeholder analysis
Stakeholder analysis is the process of identifying stakeholders and prioritising them based on their influence in the project.

Search-based software engineering
Search-based software engineering is a field that applies search-based optimisation techniques to address software engineering problems.

Requirements elicitation
Requirements elicitation is the software engineering activity in which stakeholder needs are understood.

Social network analysis
Social network analysis is the application of methods to understand the relationships among actors, and on the patterns and implications of the relationships.

Genetic algorithms
Genetic algorithms are algorithms that mimic biological evolution as a problem-solving strategy.