# Crowd-Informed Goal Models

Georgi M. Kanchev
Lancaster University
g.kanchev@lancaster.ac.uk

Pradeep K. Murukannaiah
Rochester Institute of Technology
pkmvse@rit.edu

Amit K. Chopra
Lancaster University
amit.chopra@lancaster.ac.uk

***Abstract*—A topic of recent interest is how to apply crowdsourced information toward producing better software requirements. A research question that has received little attention so far is how to leverage crowdsourced information toward creating better-informed models of requirements. In this paper, we contribute a method following which information in online discussions may be leveraged toward constructing goal models. A salient feature of our method is that it applies high-level queries to draw out potentially relevant information from discussions. We also give a subjective logic-based method for deriving an ordering of the goals based on the amount of supporting and rebutting evidence in the discussions. Such an ordering can potentially be applied toward prioritizing goals for implementation.**

## I. Introduction

This paper bridges two themes of requirements engineering: one at the core of the discipline and one of more recent interest. The core theme is that of requirements modeling. Broadly, the problem this theme addresses is how to organize, represent, and refine stakeholder requirements for a system to be developed. This theme has seen influential contributions such as goal models [5], [34], [35], problem frames [13], and UML use cases [33]. In many cases, the basic models have been extended with features that enable more sophisticated reasoning, for instance, about which requirements should be considered higher priority [21], [6].

A recent theme of growing interest concerns how crowdsourced information may be brought to bear upon software requirements. As means to achieve that, users and user feedback are starting to become important elements in requirements discovery and prioritization. Groen et al. [11] call for systematic, automatable, and salable methods for analyzing large amounts of user feedback from sources such as social media. Pagano and Bruegge [27] demonstrate that user feedback contains important information for developers, helps to improve software quality and to identify missing features, post as well as prior to deployment. Kurtanović et al. [19] categorize user sentiment from natural language feedback with high accuracy.

Clearly, crowdsourced information, including user feedback in app stores and discussions on online forums, can help inform requirements engineering. However, existing approaches have not adequately considered how such information may be brought to bear upon requirements models. In a sense, what we would like to do is to give a mapping from such information to requirement models. In other words, we want to create a requirements-oriented view over the information [16].

Supporting the creation of such views is precisely the aim of the current paper. We consider discussions between people ("users") on online forums about software applications (our particular examples are chosen from Reddit). Users often start a discussion by proposing the addition, removal, or modification of some software functionality. Others users may vote and comment on that. Some of these interactions (as including both votes and comments) indicate support for the proposal; others may indicate flaws and therefore a rebuttal of the proposal. In general, any comment may itself be supported or rebutted. In previous work, Kanchev et al. [17] have shown how such discussions may be annotated by a crowd and saved to a database so that queries in a high-level requirements-oriented language called Canary may be run against them. The idea is that a requirement engineer can potentially use Canary to get a user discussion-informed picture of system requirements; however, Kanchev et al. did not explore this direction. In this paper, we apply Canary toward the creation of goal models. We choose goal models for concreteness; analogous techniques could be created for other kinds of requirements models.

Our contributions in the paper are the following.

- We give a methodology for incrementally modifying and enriching a goal model for an application by taking into account information generated by running selected Canary queries on user discussions about that domain.
- We create a new kind of goal model where each goal is annotated with the number of supporting and rebutting interactions about the requirement expressed in the goal in the user discussions. We use subjective logic [14] on this model to obtain a prioritization of the goals.

Additionally, this paper also serves as a demonstration of how a high-level requirements language such as Canary may be systematically applied toward requirement engineering.

## II. Related Work

*1) Evidence-Based Goal Modeling:* Esfahani et al. [7] use goal models to evaluate the suitability of fragments from development methods such as Scrum or XP. They develop a goal model of a development project itself, identifying goals, and linking them to practical approaches that they employ to satisfy these goals. They use of evidence to make a qualitative evaluation of the goal model. In contrast, our method helps systematically extract evidence and incorporate that in goal models. Further, our method is generic in that it can be applied to any requirements domain.

Cailliau and Lamsweerde [4] propose a probabilistic framework for goal specification and obstacle assessment. Probabilities are calculated using a precise semantics grounded on system-specific phenomena. For quantitative calculations, we use subjective logic, which also accounts for uncertainty inherent to the evidence.

Asnar et al. [3] propose using a goal-oriented approach for analyzing risks. They use evidence to determine whether a goal will be satisfied or denied. They do not discuss the source or elicitation strategies for such evidence. Sabetzadeh et al. [31] propose a framework that combines goal models, expert elictiation and probabilistic simulation. This approach quantifies goal satisfaction using expert opinions on the leaf goals of a goal model, and propagates the information to higher level goals. Horkoff and Yu [12] propose a framework for iterative, interactive, agent-goal model analysis for early requirements engineering. In contrast to these works that reason about goal satisfaction, our method employs crowdsourced evidence to reason about goal prioritization.

Murukannaiah et al. [26] describe Arg-ACH, which combines arguments and analysis of competing hypotheses (ACH), an analytical method, to attach evidence to a goal model. Our method and Arg-ACH are complementary in that Arg-ACH employs critical questions in argumentation schemes whereas we employ high-level Canary queries to extract evidence. Further, Arg-ACH helps find conflicts but we prioritize goals.

*2) Crowdsourcing and User Feedback:* User feedback and crowdsourcing have been gaining prominence as invaluable avenues for RE [11], [27]. Tools have been developed to enable crowdsourcing requirements in enterprise settings.

Seyff et al. [32] show how general purpose social networking sites can support requirements elicitation, prioritization, and negotiation. StakeRare [22] identifies stakeholders and requirements using collaborative filtering based on social networks. These approaches rely on straight-forward aggregation of votes for the prioritization of requirements. We build on that by using subjective logic for popularity calculations.

Natural Language Processing (NLP) is a promising direction of research for crowdRE [8], [10]. Our methodology can greatly benefit from the application of NLP by automating parts of it that currently rely on human intuition.

Murukannaiah et al. [24], [25] describe how crowdsourcing and automated techniques can be combined to elicit creative requirements from the crowd.

*3) Requirements Prioritization:* Adequate requirements prioritization is crucial to any software project. A frequently studied [2] research approach for requirements prioritization is the Analytic Hierarchy Process (AHP) [30]. AHP is a multi-criteria decision making technique based on a pair-wise comparison approach. The user gives a numeric preference to each pair of requirements to represent their personal favor of one or the other. AHP has an obvious issue in its lack of scalability and exponential increase in complexity as the number of candidate requirements increases [28].

Ramirez et al. [23] propose a tool supported methodology that combines end user feedback with domain expertise. Similar to our approach, they enrich requirements with quantitative information. We explore additional automation with the use of subjective logic.

Davis [6] employs requirements triage to determine project requirements. He suggests conducting a vote to determine requirements importance. In the Hundred-Dollar Test [20], an application of the cumulative voting principle, stakeholders are given a budget of 100 votes and they "bid" on requirements they consider most pertinent. An interesting approach from classical literature are issue-based information systems introduced by Kunz and Rittel [18]. In contrast to these approaches, our method alleviates the need to gather all stakeholders in one location while still leveraging the rich variety of latent information about users' preferences and priorities available in users' comments and votes on online discussions, which is also more scalable.

## III. Methodology Overview

Figure 1 shows an overview of our methodology consisting of two major parts. The first part, consisting of four steps, employs Canary to systematically construct a goal model and attach evidence to it. We provide details on Canary in Section IV. The second part, consisting of another four steps, employs subjective logic to compute opinions about goals based on the evidence gathered in the first part. The opinions are then composed and ordered to prioritize goals. Section V provides details on this part with the necessary background on subjective logic.

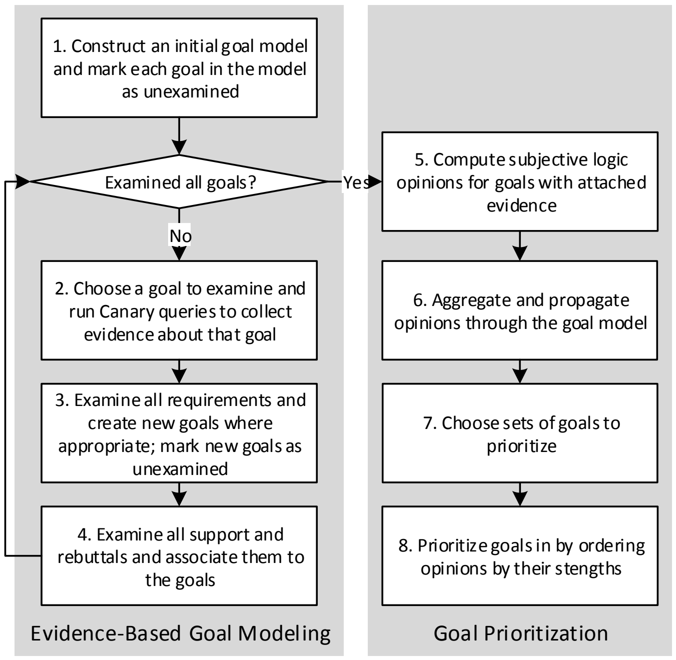

1. Construct an initial goal model and mark each goal in the model as unexamined
Examined all goals?
Yes
No
2. Choose a goal to examine and run Canary queries to collect evidence about that goal
3. Examine all requirements and create new goals where appropriate; mark new goals as unexamined
4. Examine all support and rebuttals and associate them to the goals
Evidence-Based Goal Modeling
5. Compute subjective logic opinions for goals with attached evidence
6. Aggregate and propagate opinions through the goal model
7. Choose sets of goals to prioritize
8. Prioritize goals in by ordering opinions by their stengths
Goal Prioritization

Fig. 1. Methodology overview

In a nutshell, the payoff of this methodology is a prioritized set of goals. The prioritization is based on crowdsourced evidence. As evidence we refer to comments from discussions annotated with the Canary entities (requirements, solutions, rebuttals, and supports). We propose using crowd requirements as evidence to create new goals, and supports and rebuttals to calculate goal prioritization metrics.

## IV. Evidence-Driven Goal Models

### A. Canary Query Language

Canary [17] is an approach for extracting and querying requirements-related information from online discussions. The crux of Canary is a high-level query language that combines aspects of both requirements and discussion in online forums.

User discussions capture information related to social interaction between application *users*. These interactions include users' *comments* (and their replies) and *votes* for those comments, usually measured in a metric called *score*. Canary captures requirements-related information via crowdsourced *annotations* on comments in the user discussion. Currently, Canary supports two kinds of requirements annotations: *requirement* and *solution* for a requirement (a solution always refers to a requirement). A requirement may have multiple solutions. Canary also captures *argumentation* information via annotations on comments made in response to a requirement or solution. The two kinds of argument annotations currently supported in Canary are *support* and *rebuttal*. A requirement or a solution may have multiple support and rebuttal comments. An argument comment may itself be argued about; thus, argumentation is unbounded in depth. The implementation of Canary queries uses *propagation* to infer relationships among annotated objects that arise from nesting in the discussion. Canary also propagates sentiment, which is captured as a metric over the number of supports and rebuttals, and votes.

Canary allows developers to extract pertinent data from the annotated data. Extracting such information manually would be cumbersome due to the potential volume of raw data. The query language captures all of the entities discussed above. Developers can write powerful queries that leverage entire discussions. The methodology proposed in this paper exploits Canary for information extraction.

### B. Goal-oriented Requirements Language (GRL)

We employ Goal-oriented Requirements Language (GRL) [1] since it includes all artifacts we investigate. However, our approach can be adapted for other goal modeling languages.

An important aspect of goal modeling is to look for subgoals of the original goal with AND or OR decompositions [9]. Further, real world systems can have many complexities and goals may often relate to each other in ways that do not fit into the standard AND or OR relationship. Another way of thinking about goals would be to allow for relationships where goals can contribute to each other positively or negatively. We can label such relationships with "+" and "-". The GRL notation we require for this paper is shown in Figure 2.

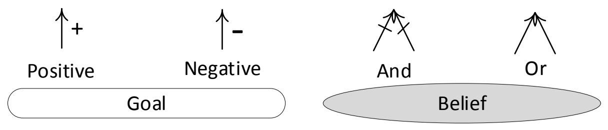


Fig. 2. Notations used for relations in GRL

During the construction of the goal model we start by stating the overall, highest level goals. Then, we proceed to decompose them using the decomposition listed above. The level of detail of the initial goal model used in this methodology may vary from project to project.

*A Running Example:* We illustrate our methodology with a *maps* application; specifically, modeling its *navigation* feature.

### C. Modeling Steps

**1. Construct an initial goal model and mark each goal in the model as unexamined.** Figure 3 shows an initial model a developer may start from for the given scenario. Here, the developer starts with two alternatives for suggesting a route, and seeks to expand on this model.

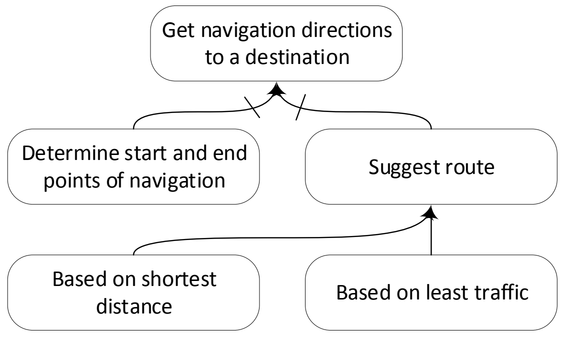

Get navigation directions to a destination
Determine start and end points of navigation
Suggest route
Based on shortest distance
Based on least traffic

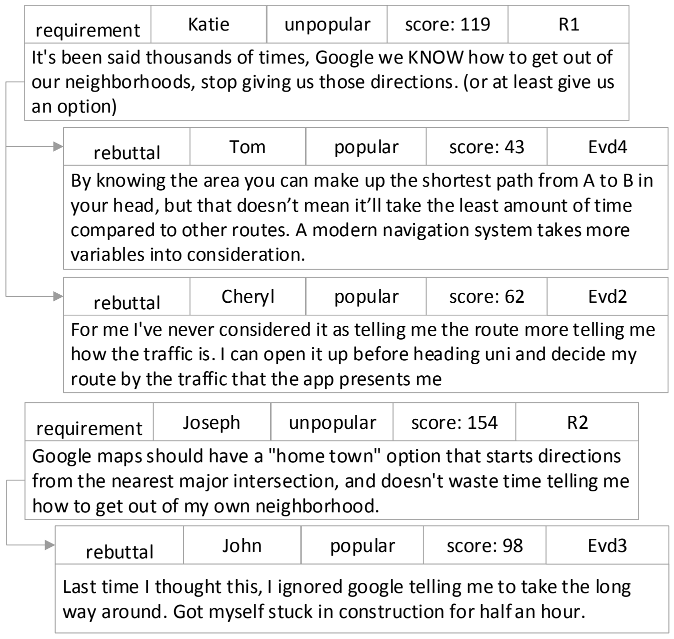
Fig. 3. An initial goal model for the maps navigation feature

Our method relies examining each goal separately until all goals have been examined. The order of examination is not of crucial. For the model in Figure 3, the developer may start by examining the top-level goal to "Get navigation directions to a destination." The developer's intention is to find rationale about it and break it down according to crowd opinions.

**2. Choose a goal to examine and run Canary queries to collect requirements about that goal.** A developer can use Canary to extract requirements-related information from the crowd and associate it to specific goals. Below, we specify a set of queries that can be run for each goal.

Figures 4 and 5 show sample pieces of evidence (annotated discussions) in the Canary database that are closely related to the model in Figure 3. A developer seeking such evidence can use the Canary language to compose various queries and extract such information from online forums.

In traditional RE, goals are used to elicit requirements. In this step we reverse the process somewhat. The developer will query for requirements based on intuition and textual similarity to the goal specification. For our example, it would be reasonable to ask for requirements that contain expressions about terminology similar to the keywords of the goals such as *directions* and *navigation*.



requirement | Katie | unpopular | score: 119 | R1

It's been said thousands of times, Google we KNOW how to get out of our neighborhoods, stop giving us those directions. (or at least give us an option)

rebuttal | Tom | popular | score: 43 | Evd4

By knowing the area you can make up the shortest path from A to B in your head, but that doesn't mean it'll take the least amount of time compared to other routes. A modern navigation system takes more variables into consideration.

rebuttal | Cheryl | popular | score: 62 | Evd2

For me I've never considered it as telling me the route more telling me how the traffic is. I can open it up before heading uni and decide my route by the traffic that the app presents me

requirement | Joseph | unpopular | score: 154 | R2

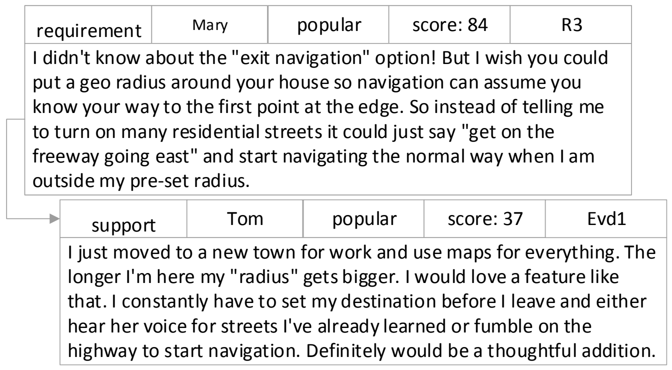
Google maps should have a "home town" option that starts directions from the nearest major intersection, and doesn't waste time telling me how to get out of my own neighborhood.

rebuttal | John | popular | score: 98 | Evd3

Last time I thought this, I ignored google telling me to take the long way around. Got myself stuck in construction for half an hour.

Fig. 4. Example of evidence related to keyword "directions"

An example of a query that would return a set containing such evidence is in Figure 6, where the developers asks for requirements containing *navigation*. Similar queries can be written for *directions* and *destination*. The developer can be creativity in composing elaborate queries to target more specific requirements. However, for the purposes of this example, we stick to basic query types.



requirement | Mary | popular | score: 84 | R3

I didn't know about the "exit navigation" option! But I wish you could put a geo radius around your house so navigation can assume you know your way to the first point at the edge. So instead of telling me to turn on many residential streets it could just say "get on the freeway going east" and start navigating the normal way when I am outside my pre-set radius.

support | Tom | popular | score: 37 | Evd1

I just moved to a new town for work and use maps for everything. The longer I'm here my "radius" gets bigger. I would love a feature like that. I constantly have to set my destination before I leave and either hear her voice for streets I've already learned or fumble on the highway to start navigation. Definitely would be a thoughtful addition.

Fig. 5. Example of evidence related to the keyword "navigation"

```
requirement where text regexp 'directions'
```

| text summary | user | score | id |
|---|---|---|---|
| Don't give directions in home neighbourhood | Katie | 119 | R1 |
| Don't give directions in home town | Joseph | 154 | R2 |

Fig. 6. An example query to find requirements related to the keyword "directions" and the corresponding output

**3. Examine all requirements and create new goals where appropriate; mark new goals as unexamined.** Each requirement we find in Step 2 should ideally map to a goal in the model. Thus, the developer must first compare each extracted requirement to each of the goals in the current goal model to determine if an existing goal maps to the requirement. This determination is subjective. However, it is possible (and desirable in the initial iterations) that some requirements cannot be mapped to any of existing goals in the model. In those cases, the developer must introduce one or more new goals. The developer may introduce the new goals anywhere in the existing model. However, we caution that adding new goals to the model must be done carefully. Such addition may require refactoring the existing goals, e.g., to decompose existing goals or to relate the new goal to the existing goals in the model.

For example, given the two requirements extracted in Figure 6, the developer may find that neither of these requirements can be directly mapped to any of the goals in existing model shown in Figure 3. However, these requirements do bear some similarity with the existing goal to "Determine start and end points of navigation." Accordingly, the developer may decide to introduce new goals to the model by decomposing an existing goal as shown in Figure 7.

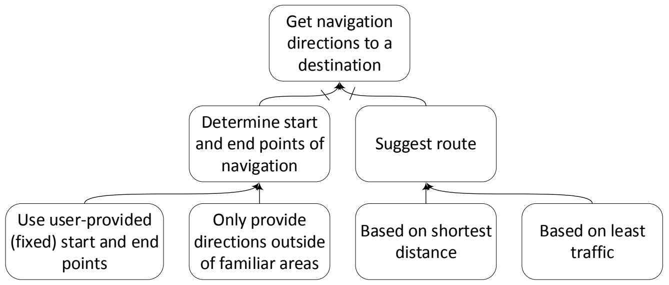

Get navigation directions to a destination
Determine start and end points of navigation
Suggest route
Use user-provided (fixed) start and end points
Only provide directions outside of familiar areas
Based on shortest distance
Based on least traffic

Fig. 7. Refactoring the goal model and adding new goals

**4. Query for support and rebuttals to; Examine all and associate them to goals where appropriate. Query for supports and rebuttals.** After extracting a set of requirements, we can go deeper into the discussion by querying the nested interactions. This may give us an insight about important applications of the goal. Associating such nested evidence is a crucial part of determining users' preferences and priorities.

The argumentative queries Canary provides are valuable in understanding the nested evidence. For example, Figure 8 shows a rebuttal query to identify conflicting evidence for the goal under consideration. Similarly, as shown in Figure 5, the discussions can also have supporting arguments. Figure 9 shows a support query that can extract such evidence.

```
rebuttal {
        requirement where text regexp
            'directions'
}
```

| text summary | user | score | id |
|---|---|---|---|
| Shortest is not always fastest | Tom | 43 | Evd4 |
| I decide based on traffic | Cheryl | 62 | Evd2 |
| Got stuck in construction | John | 98 | Evd3 |

Fig. 8. An example query to find rebuttals of requirements related to keyword "directions" and the output

Further, this step attempts to associate nested interactions (specifically, support and rebuttals) to any relevant goals in

```
support {
        requirement where text regexp
            'navigation'
}
```

| text summary | user | score | id |
|---|---|---|---|
| Useful when moved to a new town | Tom | 37 | Evd1 |

Fig. 9. An example query to find rebuttals of requirements related to keyword "navigation" and the output

the model. Note that not all evidence found in this step needs to be associated. Only those relevant to existing goals must be incorporated to the model. Our method adds evidence to the goal model as *beliefs*. The association is achieved by creating a contribution relation between the new belief and the goal it is to be associated with. The polarity of the relationship (+ or -) is determined by the developer. The results of adding such evidence to the model can be seen in Figure 10.

For example, we can focus on each piece of evidence in the table in Figure 8, namely "Got stuck in construction" and "I decide based on traffic". These are interesting examples of evidence since they relate to more than one goal. This type of evidence is valuable because it can be used to reason about two goals, but also it creates a tangible, evidence-based, rational traceability link between the goals. This particular evidence of "Got stuck in construction," for instance, reasons against the newly created goal "Only provide directions outside of familiar areas" in favor of "Based on least traffic", implying that traffic information is valuable and provides a better experience.

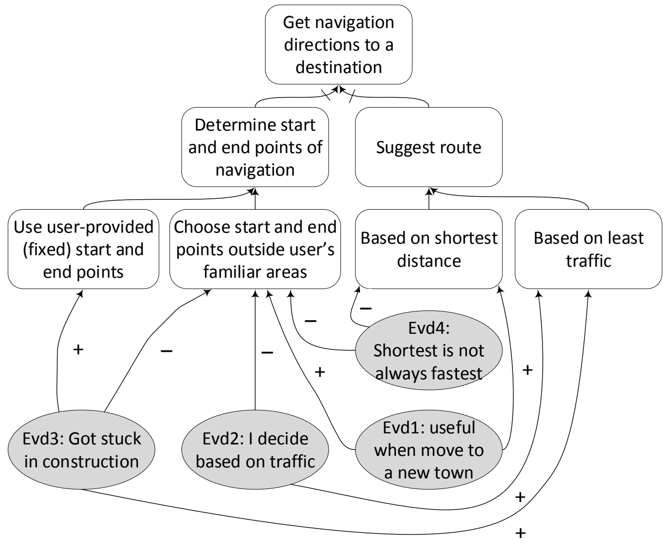

Get navigation directions to a destination
Determine start and end points of navigation
Suggest route
Use user-provided (fixed) start and end points
Choose start and end points outside user's familiar areas
Based on shortest distance
Based on least traffic
Evd4: Shortest is not always fastest
Evd3: Got stuck in construction
Evd2: I decide based on traffic
Evd1: useful when move to a new town
+
−
−
−
+
−
+
+
+

Fig. 10. Adding argumentative evidence that relates to more than one goal

## V. Goal Prioritization

Once we attach crowdsourced evidence to a goal model, we exploit that evidence to prioritize goals. Our objective is to prioritize goals such that a goal desired by more number of users gets higher priority over another goal desired by fewer users. In essence, this strategy helps a development implement requirements catering to large sets of users before implementing requirements catering to smaller sets of users.

We exploit *subjective logic* [14], a well-known belief theory, for evidence-driven goal prioritization. The subjective logic is an approximate reasoning framework that extends ideas from both classical logic and probability theory. Our choice of subjective logic for reasoning with crowdsourced evidence is motivated by three reasons.

- First, the basic premises of subjective logic are that (1) no proposition is absolutely true or absolutely false (unless it is dogmatic), and (2) the evidence on which the truthfulness of a proposition is ascertained is subjective and uncertain (in contrast, in probability theory the truth of a proposition is uncertain, but the pieces of evidence are treated as facts). These premises reflect our scenario very well in that we are seeking to reason about priorities from several subjective comments (pieces of evidence) from the members of crowd; further, since each comment may receive both support and rebuttals, that adds inherent uncertainty to the evidence.
- Second, the subjective logic framework provides negation, conjunction, and disjunction operators for combining and propagating evidence, which is necessary to compute priorities in a goal model consisting of AND and OR decompositions, and positive and negative contribution links.
- Third, the subjective logic framework also provides a fine-grained heuristics for ordering opinions, which is essential for prioritizing goals in the final step.

### A. Subjective Logic

To be self contained, we provide a brief summary of subjective logic primitives pertinent to this paper. Additional details can be found in the original works [14], [15].

*1) Basic Constructs:* The subjective logic describes constructs for reasoning about the truthfulness of the proposition in two spaces: *opinion* and *evidence* space. The opinion space represents a subject $A$'s opinion about the proposition $x$, $\omega_x^A$, as a four tuple $\langle b,\, d,\, u,\, a\rangle$, where:

- $b$ is the extent of $A$'s *belief* that $x$ is true;
- $d$ is the extent of $A$'s belief that $x$ is not true (*disbelief*);
- $u$ is $A$'s *uncertainty* about the truthfulness of $x$;
- $a$ is an *apriori base rate* parameter that determines the truthfulness $x$ when no specific evidence is available;
- $b,\, d,\, u,\, a \in [0,1]$; and $b+d+u=1$.

The evidence space, given $A$'s positive observations, $r$, and negative observations, $s$, about $x$ (i.e., $r$ and $s$ are the numbers of positive and negative observations, respectively), defines a mapping between the evidence and opinions as:

$$b=\frac{r}{r+s+2};\quad d=\frac{s}{r+s+2};\quad u=\frac{2}{r+s+2} \qquad (1)$$

*2) Mapping to Probability:* An opinion $\omega$ can be mapped to a probability expectation value, $E(\omega)$ (see Figure 8 in [15]). We require this mapping in a later stage for ordering opinions.

$$E(\omega)=b+au. \qquad (2)$$

In the evidence space, this mapping can be interpreted as a probability distribution function expressed as a *beta* distribution (for binary event spaces) with parameters $\alpha=r+2a$ and $\beta=s+2(1-a)$ (see Figure 10 in [15]).

*3) Logical Operators:* The subjective logic framework describes the following operators (among others) to facilitate evidence aggregation and propagation.

**Negation**:

$$b_{\neg x} = d_x; \quad d_{\neg x} = b_x; \quad u_{\neg x} = u_x; \quad a_{\neg x} = 1 - a_x. \qquad (3)$$

**Conjunction**:

$$\begin{aligned} b_{x \wedge y} &= b_x b_y; \quad d_{x \wedge y} = d_x + d_y - d_x d_y; \\ u_{x \wedge y} &= b_x u_y + u_x b_y + u_x u_y; \\ a_{x \wedge y} &= \frac{b_x u_y a_y + u_x a_x b_y + u_x a_x u_y a_y}{b_x u_y + u_x b_y + u_x u_y}. \end{aligned} \qquad (4)$$

**Disjunction**:

$$\begin{aligned} b_{x \vee y} &= b_x + b_y - b_x b_y; \quad d_{x \vee y} = d_x d_y; \\ u_{x \vee y} &= d_x u_y + u_x d_y + u_x u_y; \\ a_{x \vee y} &= \frac{u_x a_x + u_y a_y - b_x u_y a_y - u_x a_x b_y - u_x a_x u_y a_y}{u_x + u_y - b_x u_y - u_x b_y - u_x u_y}. \end{aligned} \qquad (5)$$

*4) Ordering Opinions:* The subjective logic orders two opinions, $\omega_x$ and $\omega_y$, the based on the following rules.

(1) The opinion with the greater probability expectation, $E(\omega)$, is the stronger than the other opinion.
(2) The opinion with the lesser uncertainty, $u$, is stronger.
(3) The opinion with the lesser base rate, $a$, is the stronger.

Here, the second or third rule is applied only if the previous rule results in a tie. Further, if there is a tie after applying all three rules, we assume that the ordering is arbitrary.

## *B. Prioritization Method*

Let us resume from Figure 10, where we had a crowd-informed goal model with associated evidence. In order to reason about priorities in this model based on subjective logic, we proceed with the next four steps of our method.

**5. Compute opinions for goals with attached evidence.** First, for each goal with at least one piece of associated evidence, we compute an opinion ($\omega$) for the goal. To do so, first we compute the amount of supporting (positive) evidence, $r$, and rebutting (negative) evidence, $s$. Further, to compute $r$ and $s$, we employ the scores associated with each piece of evidence. For example, consider the scores in Figure 11. Note that these scores are derived from scores in Figures 8 and 9, but are scaled down by a factor of ten so that the uncertainty values in later computations do not become too low (a developer must perform such tuning based on domain knowledge).

Given the scores in Figure 11, for the goal "Choose start and end points outside user's familiar areas," $r = 4$ and $s = 20$. Given $r$ and $s$, we can employ the evidence mapping functions in Equation 1 to compute the $b$, $d$, and $u$ parameters of the opinion (we assume the apriori base rate $a = 0.5$). Figure 11 shows the computed opinions.

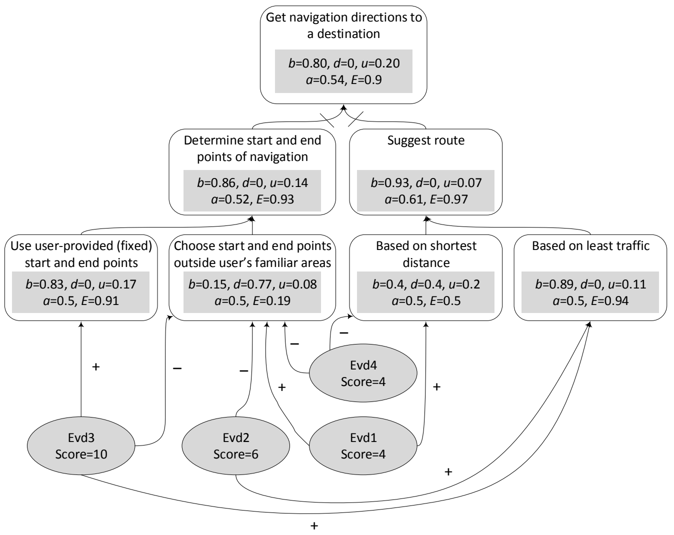


Fig. 11. Example of a goal model with opinions

**6. Aggregate and propagate opinions through the goal model.** In the previous step, we computed opinions for goals with directly associated evidence. However, there can be goals in the model with no direct evidence attached but they are connected to other nodes with direct evidence. In such cases, we employ the aggregation operators described in Equations 3–5 to aggregate evidence. Figure 11 shows examples of opinion aggregation for two OR and one AND decompositions.

**7. Identify sets of goals to prioritize.** Not all prioritizations are meaningful. For example, prioritizing between two goals in an AND relation may not be sensible since all those goals must be accomplished to achieve the parent goal. Similarly, prioritizing between a parent and its child goal may not be meaningful either since the parent goal relies on the child.

In contrast, prioritizing between goals in an OR decomposition can be valuable. Similarly, prioritizing between two top-level goals can be valuable. Such prioritization may help developers in planning which requirements to implement first given the resource constraints. In Figure 11, we see two opportunities for prioritization—one for each OR decomposition.

**8. Prioritize goals by ordering opinions.** Once we compute opinions for goals in a model, goals can be prioritized based on the rules described Section V-A4. In Figure 11, for the OR decomposition on the right, we can prioritize "Based on least traffic" over "Based on shortest distance." Similarly, for the OR decomposition on the left, we can prioritize "Use user-provided (fixed) start and end points" over "Choose start and end points outside user's familiar areas." The latter example is quite interesting in that although the feature corresponding to not navigating in familiar areas sounds innovative, there does not seem to be much evidence to support that users find such a feature useful. Accordingly, the requirement gets less priority.

## VI. Discussion

We described crowd-informed goal models that yield prioritization. The model construction is guided by an incremental yet novel methodology. We take into account information generated by running selected Canary queries on user discussions.

In the resulting goal model, each goal is associated with the number of supporting and rebutting interactions, which are augmented with quantitative metrics such as votes. We leverage such information further using subjective logic to compute the crowds "opinion" of each goal and use those values for the prioritization of the goals.

We demonstrate the value of the methodology via a running example, but lack a formal evaluation. We plan to conduct a full-scale user study to evaluate the practical value of the approach. We intend to evaluate whether our method (1) yields richer goal models and (2) simplifies the modeling process compared to other goal modeling approaches.

Automation is also a future direction. Approaches similar to Robeer et al. [29] can be used to generate an initial goal model from existing requirements artifacts. Further, we currently extract evidence via Canary queries and intuitive textual similarity. A better approach would be to study the applicability of NLP techniques for text matching based on semantic similarity.

Relying on the developer to write queries for the extraction of evidence has other risks associated to it, such as confirmation and other types of cognitive biases. Our approach currently offers no indication to the developer about how much of the available evidence they have used to create the model. Incorporating such measures in the methodology could point out to the developer, for example, that they might have decided to ignore a significant proportion of the negative interaction.

Finally, having a goal with no evidence associated to it can pose a challenge to our methodology. Choosing to assign zero values to goals with no evidence would greatly penalize the values of higher-level goals as well because of the propagation methods of subjective logic. Warning mechanisms can help alleviate this drawback, along with instructions on how to proceed. Incomplete evidence can cause problems too, such as model creation. Measuring the "completeness" of a goal model is non-trivial, but we can provide warnings when an alarming amount of requirements from the database have not been considered. The developer may then seek more evidence from the Canary database as well as from external sources.

## References

[1] Goal-Oriented Requirements Engineering. https://www.cs.toronto.edu/km/GRL/. Accessed: 2018-06-13.

[2] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd NazâĂŹri Mahrin. A Systematic Literature Review of Software Requirements Prioritization Research. *Info. and soft. tech.*,568–585, 2014.

[3] Yudistira Asnar, Paolo Giorgini, and John Mylopoulos. Goal-Driven Risk Assessment in Requirements Engineering. *RE*, 101–116, 2011.

[4] Antoine Cailliau and Axel Van Lamsweerde. A Probabilistic Framework for Goal-Oriented Risk Analysis. *In Proc. of RE*, 201–210, 2012.

[5] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-Directed Requirements Acquisition. *Sci. of Comp. Prog.*, 3–50, 1993.

[6] Alan M Davis. The Art of Requirements Triage. *Comp.*, 42–49, 2003.

[7] Hesam Chiniforooshan Esfahani, Eric Yu, and Jordi Cabot. Situational Evaluation of Method Fragments: An Evidence-Based Goal-Oriented Approach. In *Conf. on Advanced Inf. Syst. Eng.*, 424–438, 2010.

[8] Davide Fucci, Christoph Stanik, Lloyd Montgomery, Zijad Kurtanovic, Timo Johann, and Walid Maalej. Research on NLP for RE at the University of Hamburg: A Report. 2018.

[9] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with Goal Models. In *Internat. Conf. on Concep. Model.*, 167–181, 2002.

[10] Eduard C Groen, Sylwia Kopczyńska, Marc P Hauer, Tobias D Krafft, and Joerg Doerr. UsersâĂŤThe Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews. *In Proc. of RE*, 80–89, 2017.

[11] Eduard C Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, et al. The Crowd in Requirements Engineering: The Landscape and Challenges. *IEEE Software*, 44–52, 2017.

[12] Jennifer Horkoff and Eric Yu. Interactive Goal Model Analysis for Early Requirements Engineering. *In Proc. of RE*,29–61, 2016.

[13] Michael Jackson. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman, 2000.

[14] Audun Jøsang. A Logic for Uncertain Probabilities. *Internat. Journal of Uncertainity, Fuzziness, and Knowledge-Based Systems*, 279–311, 2001.

[15] Audun Jøsang, Ross Hayward, and Simon Pope. Trust Network Analysis with Subjective Logic. In *Proc. of Aust. Comp. Sci. Conf.*,85–94, 2006.

[16] Georgi Kanchev and Amit Chopra. Social Media Through the Requirements Lens: A Case Study of Google Maps. *Workshop on Crowd-Based RE* ,7-12 , 2015.

[17] Georgi M Kanchev, Pradeep K Murukannaiah, Amit K Chopra, and Pete Sawyer. Canary: Extracting Requirements-Related Information from Online Discussions. *In Proc. of RE*, 31–40, 2017.

[18] Werner Kunz and Horst WJ Rittel. *Issues as Elements of Information Systems*, volume 131. Citeseer, 1970.

[19] Zijad Kurtanović and Walid Maalej. Mining User Rationale from Software Reviews. *In Proc. of RE* , 61–70 ,2017.

[20] Dean Leffingwell. *Managing Software Requirements: a Use Case Approach*. Pearson Education India, 2003.

[21] Sotirios Liaskos, Sheila A McIlraith, Shirin Sohrabi, and John Mylopoulos. Integrating Preferences Into Goal Models for Requirements Engineering. *In Proc. of RE*, 135–144, 2010.

[22] Soo Ling Lim and Anthony Finkelstein. Stakerare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. *Trans. on Soft. Eng.*, 707–735, 2012.

[23] Itzel Morales-Ramirez, Denisse Munante, Fitsum Kifetew, Anna Perini, Angelo Susi, and Alberto Siena. Exploiting User Feedback in Tool-Supported Multi-Criteria Requirements Prioritization. *In Proc. of RE, 424–429, 2017.*

*[24] Pradeep K. Murukannaiah, Nirav Ajmeri, and Munindar P. Singh. Acquiring Creative Requirements from the Crowd: Understanding the Influences of Personality and Creative Potential in Crowd RE.* In Proc. of RE*, 176–185, 2016.*

*[25] Pradeep K. Murukannaiah, Nirav Ajmeri, and Munindar P. Singh. Toward Automating Crowd RE.* In Proc. RE*, 512–515, 2017.*

*[26] Pradeep K. Murukannaiah, Anup K. Kalia, Pankaj Telang, and Munindar P. Singh. Resolving Goal Conflicts via Argumentation-Based Analysis of Competing Hypotheses.* In Proc. of RE*, 156–165, 2015.*

*[27] Dennis Pagano and Bernd Brugge. User Involvement in Software Evolution Practice: A Case Study. In* ICSE*, 953–962, 2013.*

*[28] Anna Perini, Angelo Susi, Filippo Ricca, and Cinzia Bazzanella. An Empirical Study to Compare the Accuracy of AHP and CBRanking Techniques for Requirements Prioritization. In Proc.* Workshop on Comparative Evaluation in RE*, pages 23–35, 2007.*

*[29] Marcel Robeer, Garm Lucassen, Jan Martijn EM van der Werf, Fabiano Dalpiaz, and Sjaak Brinkkemper. Automated Extraction of Conceptual Models from User Stories via NLP.* In Proc. of RE*, 196–205, 2016.*

*[30] Thomas L Saaty. Fundamentals of the Analytic Network Process. In* Proc. Internat. Sympos. on the Analytic Hierarchy Process*, 12–14, 1999.*

*[31] Mehrdad Sabetzadeh, Davide Falessi, Lionel Briand, Stefano Di Alesio, Dag McGeorge, Vidar Åhjem, and Jonas Borg. Combining Goal Models, Expert Elicitation, and Probabilistic Simulation for Qualification of New Technology. In* Sympos. on High-Assurance Systems Eng.*, 63–72, 2011.*

*[32] Norbert Seyff, Irina Todoran, Kevin Caluser, Leif Singer, and Martin Glinz. Using Popular Social Network Sites to Support Requirements Elicitation, Prioritization and Negotiation.* Journal of Internet Services and App.*, 1–16, 2015.*

*[33] UML OMG. Unified Modelling Language version 2.5, 2017.*

*[34] Axel van Lamsweerde. Goal-Oriented Requirements Engineering: A Guided Tour. In* Proc. of Symp. on RE*, 249–262, 2001.*

*[35] Eric S. K. Yu. Towards Modelling and Reasoning Support for Early-Phase Requirements Engineering. In* Proc. Symp. on RE*, 226–235, 1997.*