# Canary: Methodology for Using Social Media to Inform Requirements Modeling

**Georgi Kanchev**

**B.Sc., Lancaster University, United Kingdom**

School of Computing and Communications

Lancaster University, UK

Thesis submitted for the degree of Doctor of Philosophy

08 18

# Canary: Methodology for Using Social Media to Inform Requirements Modeling

**Georgi Kanchev**

Submitted for the degree of Doctor of Philosophy

August 2018

# Abstract

Online discussions about software applications generate a large amount of requirements-related information. Social media serves as an extensive repository of user interaction related to software applications. Users discuss application features and express their sentiments about them in both qualitative (usually in natural language) and quantitative ways (for example, via votes). This information can potentially be usefully applied in requirements engineering; however currently, there are few systematic approaches for extracting such information. To address this gap, I applied a three-fold research approach in exploring interesting aspects of social media that can be useful to RE, pioneering a methodology for query based extraction of RE-related information from social media, and the systematic methodology for enriching established goal models with information extracted using Canary queries.

First, a study of interaction among users about Google Maps on the forum Reddit. I highlight important artifacts relevant to requirements in these interactions. I discuss goal modeling as an archetypal requirements modeling approach and use that as a basis for enhancing requirements modeling with notions that capture user interaction. To back up my observations I systematically collect, annotate, and present empirical data on the structure and value of online discussion about software applications.

Second, Canary, an approach for extracting and querying requirements-related information in online discussions. The highlight of my approach is a high-level query language that combines aspects of both requirements and discussion in online forums. I give the semantics of the query language in terms of relational databases and SQL. I demonstrate the usefulness of the language using examples on real data extracted from online discussions. My approach relies on human annotations of online discussions. I highlight the subtleties involved in interpreting the content in online discussions and the assumptions and choices I made to effectively address them. I demonstrate the feasibility of generating high-accuracy annotations by obtaining them from lay Amazon Mechanical Turk users.

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.

Third, I contribute a method following which information in online discussions may be leveraged toward constructing goal models. A salient feature of my method is that it

applies high-level queries to draw out potentially relevant information from discussions. I 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.

# Contents

# List of Figures

# List of Tables

# 1

# Introduction

This thesis 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 I address is how to organize, represent, and refine stakeholder requirements for a system that is to be developed. This theme has seen influential contributions such as goal models [DVLF93, vL01, Yu97], problem frames [Jac00], and UML use cases [Gro17]. 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 [LMSM10, Dav03].

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. [GSA+17] call for systematic, automatable, and salable methods for analyzing large amounts of user feedback from sources such as social media. Pagano and Bruegge [PB13] 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.

Put into simple words, the overarching goal of this thesis is to make social media discussions about software products available and easy to consume by requirements engineering practitioners. Raw social media data can be difficult to consume in a software development process. This creates a gap between a valuable feedback source and the people and processes it can assist. I intend to bridge this gap by using requirements models as a communication artifact. My goal is to enrich some of the most prominent and promising existing RE models with discussions generated by a crowd of users in an interactive online environment.

The importance of contact with end users and taking into account their feedback is well established as a success factor in a software development project [MM11, NS11]. Conventional methods for requirements elicitation involve direct communication between stakeholders and requirements engineers via interviews, questionnaires, focus groups, workshops, and consultations with field experts [NE00]. Combinations of such methods are often successfully used, but eliciting the more elusive, tacit requirements remains

challenging [SS13]. Informing requirements from user feedback on applications [SGM10, WSF+10, MNJR16] and *crowdsourcing* requirements [HPTA14, MAS16a] are new avenues for obtaining a fuller picture of user requirements.

Research has identified the missing link between software development teams and end users [CKI88, WRB11]. User feedback about software applications in social media is currently being either ignored or taken into account by developers in ad-hoc ways. The reasons for this state of affairs are varied. Interaction in social media is informal, voluminous, often meandering. Online interactions also come in many different types and forms. Current approaches do not systematically support *making sense* of this interaction in terms of their impact on requirements. Such approaches include techniques and tools to systematically gather, organize, visualize, and reason about such information from a requirements perspective [BL11].

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. The key idea of this thesis is to give a mapping from such information to requirement models. In other words, I aim to create a requirements-oriented view over the information. Such requirements oriented view would be directly beneficial to software engineering practitioners who are specifically interested in market-driven software development [RB05]. Market-driven software development is concerned with software projects that are not tied to any specific group of users, such as bespoke software for a company. Such projects target a crowd of users who do not necessarily have an association between each other. Eliciting requirements for such projects is particularly challenging because it lacks any rigid idea of who the users will be [KDR+07, GSA+17]. Social media can be used to address this problem, by eliciting requirements directly on a crowd of users, but information in social media comes with innate challenges.

In this thesis, I perform an analysis on the different aspects of social media and how they can be leveraged to benefit requirements engineering. I am especially interested in systematic approaches of organizing requirements-related information found in social media in a structured way. The *requirements lens* is a metaphor for such approaches. The term requirements lens is meant to encapsulate in itself my aim to skew, twist, mine the augment raw information found in social media. It is supposed to communicate with the reader that the tools I build will give practitioners that wish to use them a powerful new way of focusing on specific objects of interest that they are after from a potentially large and confusing data source. I found the necessity to coin the term requirements lens on account of the emerging other metaphorical uses of the word *lens* with a similar connotation. This uptake in the use of the word in such contexts will help me to communicate the purpose of my research and how it can be useful to people that have encountered the metaphor in other, abstractly similar situations. Traditional elicitation techniques fail to take advantage of rich user feedback on social media. In this research work I perform an analysis on the different aspects of social media and how they can be leveraged to benefit requirements engineering. I set out to identify, study and explore ways of leveraging the underlying structure that emerges in online discussions.

Content in social media, and especially forums, is usually in the form of unstructured natural language enhanced with various quantitative attributes gained from user interaction in an online collaborative setting, such as votes and ratings [ACD+08]. Producing a

requirements view of social media would require exploration into the domain of discussion about software in social median. Observations on the innate structures of such discussions must be gathered, studied and recorded. Organized, visual end-user feedback will create an interesting, high-level, abstract view that will potentially aid various processes in requirements engineering.

Supporting the creation of a requirements view is precisely the aim of this thesis. My research explores whether there is an underlying naturally occurring structure to online discussions about software requirements that can be leveraged for the purposes of building better software specifications during the requirements engineering stages of a software development projects. A requirements view can be achieved by providing developers with tools and systematic methodologies that would enable them to mine and explore for information that can be leveraged to build better and improve existing software. If there was an underlying structure to online discussions about software such software tools can be built to leverage it.

## 1.1 Research Questions and Problems

Following from the previous, throughout this thesis I will explore the following research questions:

- **Problem** Online discussions about software products are unstructured.
  - **RQ1** What abstractions can be used to capture and structure the information that is of relevance to requirements engineering?

- **Problem** Online discussions are voluminous. Annotation of large amounts of data in natural language is challenging and error prone. In order to be able to apply my work to the software engineering, an exploration of how the annotation process can be scaled to large volumes of text is needed.
  - **RQ2** How effective are crowd-sourcing techniques at scaling up the annotation of online discussions about software products?

    In the context of this research question effectiveness is measured in accuracy and efficiency.

- **Problem** No systematic ways of leveraging labeled online discussions have been formally conceptualized and subjected to scientific study.
  - **RQ3** How to design an artifact that extracts pertinent information for RE practitioners from labeled online discussions?

- **Problem** Information from online discussions cannot be applied directly to the existing requirements engineering process due to a lack of integration with existing promising RE artifacts.
  - **RQ4** What is a method for goal-driven prioritization that uses evidence from online discussions about software products?

In the context of this thesis I will use the term **artifact** as per the definition used in the field of design science (detailed in Section 1.2.3) by Wieringa [Wie14]. An artifact is something created by people for some practical purpose. Examples of artifacts designed and studied in information systems and software engineering research are algorithms, methods, notations, techniques, and even conceptual frameworks. They are used when designing, developing, implementing, maintaining, and using information systems and software systems. When an artifact is used, it is used by people, which means that it interacts with a context that, along with other things, contains people.

In order to achieve this requirements view three major research goals must be explored and reached in sequence. Each sequential goal builds on top of the findings, software tools, and results of the previous one. All together the results of the three steps will produce an enriched perspective on the requirements of an existing or newly conceived software product.

- In order to address **RQ1**, I must first take on an exploration into the underlying structure of online discussions about software applications. The nature of the source of information is highly unstructured. In order to be able to build a tool-assisted framework for such a source I must investigate and make observations of any naturally occurring patterns in online discussions about software that can be leveraged. Once I have an intuition of can be candidate sources of abstractions, I can properly address **RQ1** with an experiment of manual annotation of discussions. Several aspects of online discussions make the task of using them to inform application development challenging. The data is often voluminous with a single top-level post often invoking a long discussion. Additionally, most of the discussion is carried out in natural language; so identifying aspects of the discussion (e.g., whether some comment expresses a requirement) is nontrivial. I will address such concerns and address **RQ2** with another experiment that applies a scalable approach of annotation, such as crowdsourcing.

- Second, build a tool-assisted methodology that leverages the structures and organizes the information using the abstract concepts explored in the previous goal in order to allow practitioners to gain a high level view of discussion. Building such a methodology would explore one possible answer to **RQ3**. There is currently no systematic methodology of leveraging requirements-related information from social media that supports features related to social interaction. For example, there is no tool that would allow a requirements engineer at Google to formulate a query such as *give me the five most controversial requirements about Google Maps over the last two months*, where "controversy" captures aspects of social interaction (discussion).

- Third, devise a methodology that feeds any gained insights into the RE practitioner process. In order to create a full requirements view of social media it is necessary to propose systematic ways of feeding the insights gained from exploring the feedback into enriching existing requirements engineering artifacts. Enrichment of existing artifacts, as opposed to creation of completely new ones, will ease the adoption. Goal models emerge as a strong candidate, because they are a popular research topic with high potential to be taken up by practitioners. An exploration into the enrichment of goal models will provide an insight into the answer of **RQ4**.

## 1.2 Methods

There are many considerations to be taken into account when making methodological decisions about the work needed to adequately answer the research questions defined previously. In the beginning of this chapter I will outline and introduce relevant research techniques. I will then proceed to define and defend the methodological choices I've made to answer my research questions.

### 1.2.1 Grounded Theory

Grounded theory is a systematic methodology originating from the social sciences. Its application constructs theories through methodical gathering and analysis of data [SC94, MT86]. The mean research method of grounded theory is inductive. This is a stark contrast to the more traditional for scientific endeavours approach, based on the formulation of a hypothesis that is investigated using deduction. The starting point of a study that applies the grounded theory method is a question. In some cases the study begins with a collection of qualitative data. Through a systematic review of the data collected, repeated ideas, concepts or elements become apparent. Researchers make use of tags and codes to label such repeated observations, which have been observed in the data. The proper application of grounded theory calls for collecting an increasing amount of data and further grouping the codes into concepts and into categories. New theories can be formulated on the basis of these concepts and categories. Thus, grounded theory differs significantly to the traditional model of research, where the flow of the study is reversed. Typically, the researcher chooses an existing theoretical framework, then collects data to show how the theory does or does not apply to the phenomenon under question [All03].

### 1.2.2 Empirical Research

Empirical studies are the collection and analysis of primary data based on direct observation or experiences in the "field". Empirical research is research using empirical evidence. It is a way of gaining knowledge by means of direct and indirect observation or experience. Empiricism values such research more than other kinds. Empirical evidence (the record of one's direct observations or experiences) can be analyzed quantitatively or qualitatively. Quantifying the evidence or making sense of it in qualitative form, a researcher can answer empirical questions, which should be clearly defined and answerable with the evidence collected (usually called data).

### 1.2.3 Design Science

Design science, as defined by Wieringa [Wie14], is the design and investigation of artifacts in context. The artifacts under study are designed to be an interactive solution, placed in a problem context in order to improve the state of the art solutions in that domain. There are two major parts to a design science project. First, its object of investigation. Second, its two major activities. The object of study is an artifact in its context. The two major activities are designing and investigating this artifact in context. Design and

investigation, directly equated to two kinds of research problems, namely, *design problems* an *knowledge questions*.

- **Design problems** call for a tangible change in the real world and require an analysis of actual or hypothetical stakeholder goals. The finished solution of such problems is a design, and there are usually no single best solution. Many possible solutions could be in existence in response to a design problem. It is possible that one will come up with as many solutions as the number of designers one assigns to tackle the design problem. Evaluation for solutions must reflect that and the evaluation must be taking into consideration stakeholder goals and how well they are met. There is no single best solution.

- **Knowledge questions** by contrast, do not call for a change in the world but ask for knowledge about the world itself, as it is. The answer is a proposition, and when we try to answer a knowledge question, we assume that there is one answer only. We do not know the answer, and we may give the wrong answer; we may have degrees of (un)certainty about the answer, and the answer may be true in most but not all cases. But answering a knowledge question would lose all meaning if there would be as many answers as researchers. And answers to knowledge questions do not depend on stakeholder goals. Rational discourse implies the assumption of single truth but must be combined with the assumption of **fallibilism**: we can never be sure that we have actually found the answer to an empirical knowledge question.

To summarize, design science iterates over solving design problems and answering knowledge questions. The social context of a design science project consists of stakeholders who may affect or may be affected by the project. The knowledge context consists of knowledge from natural science, design science, design specifications, useful facts, practical knowledge, and common sense. Generalizations produced by design science research may abstract from some conditions of practice but do not make unrealizable idealizations. They generalize beyond the case level but are not universal.

A word is necessary for the *design cycle*. A design science project iterates over the activities of designing and investigating. The design task itself is decomposed into three tasks, namely, **problem investigation**, **treatment design**, and **treatment validation**. This set of three tasks is called the design cycle, because researchers iterate over these tasks many times in a design science research project. The design cycle is part of a larger cycle, in which the result of the design cycle — a validated treatment — is transferred to the real world, used, and evaluated. This larger cycle is called the engineering cycle.

With the context of this introduction to the research methodologies I've used I will now proceed to investigate each of my research questions, presenting and defending my choice of methodological investigation.

- **RQ1** presents my research question that seeks the presence and examines the nature of a naturally occurring structure to online discussions about software products. The nature of this question lies in the domain of knowledge. In [Wie09] Wieringa provides a classification of questions. According to the classification, this question is about conceptual modeling. It attempts to answer the question of which concepts to use in

order to model a discussion. Grounded theory requires for the methodical gathering of data and forming a theory based on the observations. Due to the lack of previous observations and studies to use as basis for the formation of a theory, grounded theory is an adequate approach to finding an answer to this question. The choice for grounded theory is based on the fact that a study on the naturally occurring structure of online discussions is a first of its kind to the best knowledge of the author and therefore the methodology needs to reflect the exploratory nature of this experiment. Grounded theory is a fitting candidate.

A subqestion to **RQ1** asks whether combination of abstractions borrowed from argumentation and requirements engineering would adequately capture the structure of online discussions about software applications. In my exploration of the data using grounded theory I will produce a set of observations based on which I can form a theory on what abstractions can be used to model online discussions. The fit of this theory can be a reasonable subject of doubt and questioning due to the qualitative nature of its origin. I can put my theory to a test that by creating an experiment in which I apply an initial set of annotations to the dataset and perform an analysis to provide evidence for the nature of structure of online discussions.

- **RQ2** aims at exploring whether crowdsourcing is an effective, scalable way of applying annotations on large volumes of discussions about software products in natural language. **RQ2** is also a knowledge question that can be studied and answered. In order to address this question I designed an empirical user study to be carried out on Amazon Mechanical Turk. It aims at evaluating how good semi-automated approaches are at leveraging human intelligence by deploying small tasks to a large number of people in exchange for monetary incentive in the case of successfully completed task. In order to answer my research question I collect a number of empirical observations relating to efficiency and accuracy of annotations. I also make meta observations on the effects of repeating the task and provide some observations based on the demographics of the participants for completeness.

- My next research question, stated in **RQ3**, is explored using design science. It seeks out a satisfactory way of designing an artifact that extracts pertinent information for RE practitioners from labeled online discussions. **RQ3** is a design problem. In order to address this problem I designed a solution in the form of the Canary methodology and query language. I present a detailed specification of my solution and defend my design choices against the specifics of the problem. My solution is validated by the use of real life examples manually extracted from social media to use as a demonstration of the value of the Canary query language and methodology. Additionally I reflect on the strengths and weaknesses of the design of my solution.

- In my final work, **RQ4**, is also explored using design science. It aims to find what would be a suitable method for goal-driven prioritization that uses evidence from online discussions about software products. In the nature of design science I design and implement my solution to the problem, driven by the requirements of the envisioned stakeholders, in this case RE practitioners. For validation purposes I use examples extracted from real online discussions. This work builds on top of all my previous work and thus further emphasizes the power of abstractions that I apply to the raw information and the value of Canary by using it to enrich a high profile artifact from

the field of requirements engineering.

### 1.2.4 Contributions

The work described in this thesis provides he following contributions to science:

- The key findings of a grounded theory exploration of natural language discussions in social media, shown in Chapter 3.2.
- The results of an experiment involving expert annotation of discussions with RE and argumentative labels shown in Chapter 3.6.2 and the produced dataset.
- The results of an experiment of using crowdsourcing as a scalable source of annotations shown in 3.7.3.
- A tool-assisted methodology for the systematic querying of requirements-related information form online discussions derived using design science, presented in Chapter 4. The centrepiece of the methodology is the novel Canary Query Language, detailed in Chapter 4.3.
- A methodology, derived using design science, for the enhancement of goal models with data found in social media with the purpose of prioritization, presented in Chapter 5.

## 1.3 Thesis Overview

In Chapter 2, I present background literature and position the problem and contributions of this these. In Chapter 3, I address **RQ1** by carrying out an analysis of online discussions by applying grounded theory to explore the naturally occurring structures found in discussions. Additionally, it contains a section detailing a scientific experiment that explores scalable approaches for the enrichment of raw data to contain meta data that would allow the automation of information mining, thus addressing **RQ2**. Next, **RQ3** is addressed in Chapter 4, where I present a methodology that enables software developers to systematically query large the bodies of natural language produced by discussions using a structured query language named Canary. Chapter 5 holds my work on addressing **RQ4**, namely another methodology that shows the value of Canary by using its queries to enrich goal models, a requirements engineering artifact that has the potential to be widely adopted by practitioners. In Chapter 6, I present my final thoughts, findings and reflections and conclude my work.

# 2

# Background and Related Work

## 2.1 Overview

In this chapter I position my work in relation to the background literature. Additionally I go over a selection of the existing scientific works that are related to my work. The chapter is divided into those two subsections. The subsection on related work is further divided, organizing the works that I comment on into logical groups ordered in the same chronological order that the rest of my thesis is structured in. It begins with description of the background in which my work is positioned, which is crowd-based requirements engineering. I then move on to comment on existing techniques that allow practitioners to query requirements-related information. The next broad category of works I have included is on evidence-based goal modeling, where I relate my work to existing studies on that topic. There are hundreds of studies on enriching goal models. Next I discuss some of them and reference a comprehensive meta study. The section is concluded with a discussion on requirements prioritization, which is also the chronological payoff to my thesis.

## 2.2 Background

In order to give a better understanding of who the end user is in different software projects, Lubras et al. [LPR93] introduce the terms market-driven and customer-specific development. In customer-specific development the project has a *specific*, contractual customer (who will be the end user), who can be addressed directly. In the case of market-driven projects there is more than one *potential* end user.

Maalej et al. [MHR09] classify the types of user feedback. One, *pull* if the feedback is pulled from the user; two, *push* if the feedback is pushed by the user; three, *explicit* if the user has intention to provide the feedback; four, *implicit* if the user unintentionally provides the feedback, for example background usage statistics. To this classification I add *internal* (if the feedback is collected within the system, that is internal bug report signal), *external* (if the feedback is gathered from a third party system, for example social

media), *qualitative* (if the feedback gathered is of qualitative nature—natural language, pictures, video, etc.) and *quantitative* (if the feedback is of quantitative nature, such as usage statistics, numerical rating system). In this thesis I consider pushed, explicit, external, qualitative (enhanced with various quantitative attributes) feedback in social media.

Modern elicitation techniques are good at extracting an abundant set of requirements. This introduces problems, such as filtering and prioritization [KDR+07, WRB11]. Developers gather requirements into potentially huge repositories in natural language. The lack of context associated with those requirements creates difficulties for developers in interpreting and prioritizing them. Studies call for more structured (context-aware) input of requirements-related data [RB05].

*Argumentation* has long been advocated as a way of recording the rationale of requirements [RD92]. In recent work, Yu et al. [YFT+15] apply Toulmin's argumentation schema [Tou58a] to automated reasoning about security requirements. The reasoning is akin to running queries on argumentation bases. Versions of Canary targeted toward expert users could support richer argumentation and reasoning.

### 2.2.1 Crowd-based requirements engineering

Nowadays every product has a crowd. Groen et al. [GK18] define such a crowd as a very large, heterogeneous, and physically divided pool of stakeholders who interact with each other online. They also claim that that the ability to channel and manage such a crowd gives a competitive advantage. In another publication Groen et al. [GSA+17] also recognize that engaging a large number of users in requirements engineering is challenging with traditional RE methods. They note that this holds stronger validity when RE should involve a large number of software product users (a crowd) who are beyond an organization's reach. Traditional requirements engineering (RE) techniques face scalability issues and require the co-presence of stakeholders and engineers, which cannot be realized in a crowd setting [GDA15].

A strict definition of what crowd-based RE consists of is still in the process of emerging in literature. The most concise and on point definition comes from various publications by Groen et al. and it positions crowd-based requirements engineering (CrowdRE) as an umbrella term for automated or semi-automated approaches to gather and analyze information from a crowd to derive validated user requirements. Normally, the crowd is an undefined group of people [SF14]. But for CrowdRE, the crowd is in most cases a large group of current or potential users of a software product who interact among themselves or with representatives of a software company (for example, the product owner or development team).

The overall goal of CrowdRE is to allow participation to in the RE process to a large number of end users, usually through various forms of online collaboration, including discussions, collaborative filtering, etc. Although there are no strict goals and incentives in CrowdRe users are usually thought to be driven by the common goal of evolving the existing software to better suit their common needs. Such ad-hoc interactions of end users are referred to as "user feedback". User feedback can also come from other stakeholders. Attempts at categorizing feedback have been made by Maleej et al. [MHR09]. They

define pull feedback to be the process when the software company explicitly asks the crowd for feedback. Push feedback is when the crowd initiates feedback. CrowdRE can also be stretched to involve other methods of gathering metrics, such as application context and usage.

Groen et al. [GSA+17] identify four distinct aspects of the CrowdRE approach, *motivating crowd members*, *eliciting feedback*, *analyzing feedback*, and *monitoring context and usage data*. In my work, I've addressed issues concerning elicitation and analysis of crowd-based data that can improve the RE process.

### 2.2.2 Argumentation

Argumentation has a long history in research, but a monumental early step is undoubtedly Toulmin's [Tou58b] influential analysis of argument (Toulmin, 1958). At the time it was conventional to break down arguments into a premise and conclusions. Toulmin studied the use of arguments with the aim to identify different roles that parts of a statements can play in arguments, in other words, how do different segments of a statement contribute to the persuasive force of the overall argument. Toulmin proposed a scheme with six functional roles (see Figure 2.1.a): Conclusions are derived based on some basis ("data") and a possibly implicit but feasible generalization ("warrant"). The conclusion can be "qualified" using a modal operator. The inference of the argument can be undermined using a "rebuttal". Support for an argument is presented using "backing".

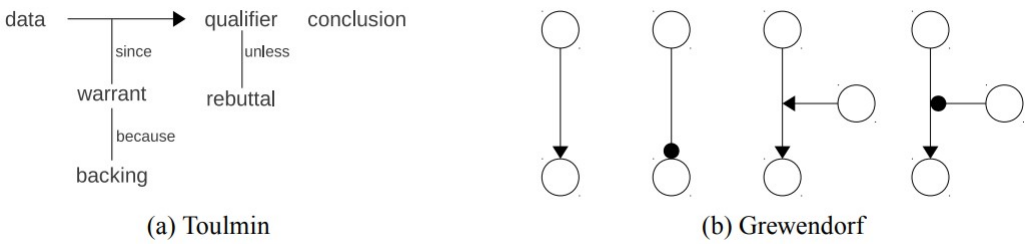



Figure 2.1: Diagramming techniques in theories of argumentation of Toulmin and Grewendorf

From then on, Grewendorf [Gre80] thought of a dialog-oriented diagram method. It allows for more interactive discussions to be modeled by distinguishing between counterarguments that are brought up by the opponent as attack from those that the proponent himself presents in order to refute them. Another major difference is that Grewendorf replaces the tree structure with a graph, so that nodes can participate in multiple support or attack relations. In Figure 2.1.b different arrows arrowheads denote support, those with a circle an attack. Finally, Grewendorf makes the important move to allow support and attack not only for statements (nodes) but also (recursively) for support and attack relations.

In more recent work, Peldszus and Stede [PS13] provide a definition of argumentation and identify a set of relations between entities (premises) in an argument. An argument consists of a non-empty set of premises supporting some conclusion. Argumentation is the structure that emerges when multiple arguments are related to each other and form larger complexes. A high level set of relations between entities in an argument is *support*, *attack*, and *counter-attack*.

- *Support.* There are multiple ways to provide further support to the conclusion. The four identified patterns by Peldszus and Stede are linked (two separate premises that are dependent on each other to provide support to the claim), multiple (two independent premises that are able to support the claim on their own), serial (premises supporting each other), and example (when the premise consists of an example) support. Attack There are two actions that can be done to attack a premise - rebut and undercut.
- *Rebuttal* is a claim to prove that something (another premise) is false or invalid. Undercutting is an attack on the validity of the relation between the premises, rather than on the validity of either of the premises. Attack relations can be classified as rebut a conclusion, rebut a premise, undercut an argument, and support a rebutter.

## 2.3 Related Work

### 2.3.1 Crowdsourcing and User Feedback

User feedback and crowdsourcing have been gaining prominence as invaluable avenues for RE [GSA+17, PB13] and has long been argued as essential for the RE process [GDA15]. Tools have been developed to enable crowdsourcing requirements in enterprise settings.

A study by Seyff et.al. [STC+15] shows how popular social networking sites can support requirements elicitation, prioritization, and negotiation. The study was carried out on groups of students that were asked to use Facebook as a medium of communication. The study was carried out in Facebook which, although popular and widespread, is lacking key elements that would enhance the results. Facebook, and many other popular social media outlets, doesn't support a hierarchical, chronological comment/reply view of discussions, which makes them difficult to follow and therefore demotivates participants to have a continuous, in depth negotiation. In my empirical evaluation of online discussions about software applications I found evidence that a significant portion of the valuable information from a discussion can be found in the depths of a discussion, rather than the first few, shallow, responses. The voting system used on Facebook also has drawbacks, as it doesn't support negative votes — participants are unable to express their dissatisfaction with a post. Seyff et. al. don't discuss any possible directions of research regarding gaining a systematic understanding of the resulting data.

Various social media techniques have been applied in research for the purposes of requirements engineering. Lim et al. proposed StakeNet [LQF10] as a crowdsourcing solution to stakeholder identification. They create a social network of stakeholders involved in the software project by having each of them to recommend stakeholders and stakeholder roles, build a social network whose nodes are stakeholders and links are recommendations, and prioritise stakeholders using a variety of social network measures. Later that idea was expanded into StakeRare [LF12], where stakeholders in the network were also allowed to propose requirements and prioritize them using collaborative filtering. StakeRare identifies stakeholders and requirements using collaborative filtering based on social networks. StakeRare inherits all stakeholder identification features from StakeNet. On top of that it then asks the stakeholders to rate an initial list of requirements, recommends other relevant requirements to them using collaborative filtering, and prioritises their requirements using their ratings weighted by their project influence. These approaches

rely on straight-forward aggregation of votes for the prioritization of requirements. I build on that by using subjective logic for popularity calculations. These techniques are beneficial to customer-specific projects, but their application is limited in market-driven development. StakeNet and StakeRare take advantage of the ability to directly address a commissioned customer. Greenwood et al. introduce UDesignIt [GRW12], where they draw text from social media and apply natural language processing to generate a prioritized feature model.

Natural Language Processing (NLP) is a promising direction of research for CrowdRE [FSM+18, GKH+17]. My methodology can greatly benefit from the application of NLP by automating parts of it that currently rely on human intuition.

Murukannaiah et al. [MAS16b, MAS17] describe how crowdsourcing and automated techniques can be combined to elicit creative requirements from the crowd. They investigate how human personality and creative potential influence a requirement acquisition task. They propose a sequential Crowd RE process, where workers in one stage review requirements from the previous stage and produce additional requirements. They find that exposing a worker to ideas from previous workers cognitively stimulates the worker to produce creative ideas. These findings are of high relevance to my work, since social media inherently benefits from the same boosts in creativity studied in the paper. All new participants are exposed to the requirements proposed by earlier participants and can go in an iterative process of refinement through a comment-reply interaction.

Johann and Maalej [JM15] discuss giving users varying degrees of influence in RE by using different e-democracy strategies. They systematically delegate the responsibility for developing the requirements of a software project and deciding about future releases to the crowd of users. As a vision, paper they discuss the pros and cons of their vision, its main challenges, and sketch promising solution concepts. Democratic approaches can be seen as a higher level of abstract structure above mere argumentation. The work outlined in this paper can be used as an extension to my work in the future.

Numerous services now support user feedback on applications. Bajic et. al [BL11] analyze how software companies are finding ways to use social media techniques to gather feedback from users collectively. As an example from their findings, UserVoice attracts participation from a large community of users and has elicited thousands of bugs, problems, and suggestions for improvements.

Some work builds requirements-related annotations into a user platform [BS14b, MPP15]. The alternative would be to obtain the annotations offline, either manually or with natural language processing support [MVM+15]. I note ongoing efforts to develop conceptual models of online user discussions [MPG15].

Various other benefits of involving the crowd in the RE process have been studied in literature, such as creativity [MAS16a, HML15, ST15]. Stimulating the creativity of the crowd by, e.g., interacting with each others' ideas, has been found to measurably increase the quality of elicited requirements. Canary is built upon data generated by interaction in the form of argumentation between users in online forums. Murukannaiah et al. [MKTS15a] report that such argumentation promotes the elicitation of better requirements. Murukannaiah et al. describe Arg-ACH, which combines arguments and analysis of competing hypotheses (ACH), an analytical method, to attach evidence to a goal model. My method and Arg-ACH are complementary in that Arg-ACH employs

critical questions in argumentation schemes whereas I employ high-level Canary queries to extract evidence. Further, Arg-ACH helps find conflicts but I prioritize goals.

### 2.3.2 Natural language processing (NLP) in RE

has traditionally been seen as a promising tool for requirements analysis and there is resurgence of interest, recently [SRC05, AG06, GZ14, MSJ+16, PSG+16]. Likewise, user feedback on Google Play and Apple Appstore has been used toward gaining a better understanding of requirements for the next release [PM13a]. In their paper they report on an exploratory study, which analyzes over one million reviews from the Apple AppStore. They investigated how and when users provide feedback, inspected the feedback content, and analyzed its impact on the user community. Among other things they report that quality and constructiveness vary widely, from helpful advice and innovative ideas to insulting offenses. Their work proves the value of user feedback and paves the way for other structured approaches at leveraging it, such as the ones I am investigating in this thesis.

Maalej and Nabil [MN15] provide NLP techniques for classification of user feedback with a high degree of precision and recall. This paper introduces several probabilistic techniques to classify app reviews into four types: bug reports, feature requests, user experiences, and ratings. Canary's annotations bear some similarity to Maleej and Nabil's categories; however, the online discussions I analyze are more complex in structure and richer in content.

Maalej et al. [MNJR16] in fact discuss the lack of systematic approaches to organize, summarize, and aggregate data from user communities. They report on trends that suggest a shift toward data-driven user-centered identification, prioritization, and management of software requirements. My approach using human intelligence is complementary to machine intelligence techniques.

Ciurumelea et al. [CSPG17] advocate that developers must closely monitor and analyze the user feedback they receive in form of reviews. However they claim that the amount of data that needs to be processed can be overwhelming. They suggest applying NLP techniques on reviews and code of mobile apps with the goal of better release planning. Then they built the User Request Referencer (URR) prototype, using Machine Learning and Information Retrieval techniques, to automatically classify reviews according to their taxonomy and recommend for a particular review what are the source code files that need to be modified to handle the issue described in the user review.

NLP can also be successfully applied in mining Twitter feeds with the aims to infer users' needs, detect bugs in their code, and plan for future releases of their systems, as Williams and Mahmoud have shown [WM17]. They report on a three-fold study that is aimed at leveraging Twitter as a main source of software user requirements. Their results reveal that around half of collected tweets contain useful technical information. Their results also show that text classifiers can be very effective in capturing and categorizing technically informative tweets.

SAFE [JSM+17] is an approach of feature extraction from app description and app reviews. The approach relies on 18 part-of-speech patterns and 5 sentence patterns that

are frequently used in text referring to app features. The patterns were manually built by the researchers.

Kurtanović et al. [KM17] categorize user sentiment from natural language feedback with high accuracy. Through a grounded theory approach and peer content analysis, they investigated how users argue and justify their decisions, for example about upgrading, installing, or choosing a competitor software applications. They then used the truth set of manually labeled review sentences to explore how accurately they can mine rationale concepts from the reviews.

### 2.3.3 Querying requirements.

IBM DOORS [IBM] enables capturing users discussions on requirements; however, querying is limited to text searches. IBM DOORS is a requirements management application for optimizing requirements communication, collaboration and verification throughout your organization and supply chain. It allows you to create relationships, trace dependencies, empower multiple teams to collaborate in near real-time and handle versioning and change management. IBM DOORs, does little to address the needs of those who wish to mine raw natural language data to elicit requirements. Canary queries are significantly more sophisticated and would allow for the elicitation of more pertinent information. Tools such as DOORS would benefit from Canary. TiQi by Huang et al. [PLA+14] allows the transformation of spoken natural language into structured SQL. The achieve this by the use of general database query mechanism and a domain-specific model populated with trace query concepts, project-specific terminology, token disambiguators, and query transformation rules. It's designed to make traceability information easily accessible to its users, similar to what I am trying to achieve with Canary and information generated in online interaction. ReqIF [EJ12] makes the exchange of formalized requirements between autonomous business partners possible. It also allows for the creation of custom relations between objects in the database and other queries based on attributes and pattern matching on strings.

### 2.3.4 Evidence-Based Goal Modeling

Esfahani et al. [EYC10] propose using goal models to evaluate and adapt the suitability of fragments from well-known software development methods such as Scrum or XP. They leverage the increasing availability of empirical evidence on the success or failure of various software development methods under different situational conditions by creating a repository of such evidence [EY10]. They develop a goal model of a development project itself, identifying goals such as *improved awareness of teammate activities*, and linking them to practical approaches that they employ to satisfy these goals, such as *conduct daily scrum meetings* and use the repository of evidence to make a qualitative evaluation of the goal model. In contrast, my method of incorporating evidence to goal models is generic and it can be applied to any requirements domain.

Cailliau and Lamsweerde [CVL12] propose a probabilistic framework for goal specification and obstacle assessment. Probabilities are calculated using a precise semantics grounded on system-specific phenomena. The probability of a root obstacle to a goal is thereby computed by up-propagation of probabilities of finer-grained obstacles through the

obstacle refinement tree. For quantitative calculations, I use subjective logic, which also accounts for uncertainty inherent to the evidence.

Asnar et al. [AGM11] propose using a goal-oriented approach for analyzing risks during the requirements analysis phase. They analyze risks along with stakeholder interests, and then identify and introduce countermeasures. The methodology consists of three types of constructs: goals, tasks, and events. Each construct has two attributes: SAT and DEN. Such attributes represent, respectively, available evidence that the construct N will be satisfied or denied. They do not discuss the source or elicitation strategies for such evidence. Sabetzadeh et al. propose a framework that combines goal models, expert elicitation and probabilistic simulation for quality assurance of high-risk projects. This approach quantifies goal satisfaction using expert opinions on the probability of satisfaction of leaf goals (the most actionable, low level goals) of a goal model, and propagates the satisfaction probabilities to higher level goals. However, this approach does not specify the source of evidence or the elicitation process, which are the key aspects of my method.

Letier and Lamsweerde [LVL04] explore ways to quantify the impact of alternative system designs on the degree of satisfaction on non-functional goals. Partial degrees of satisfaction are characterized in terms of application-specific phenomena, such as *quality variables* (response time) and *objective functions* (probability of response time being less than 8 seconds), and are propagated upwards or downwards in goal refinement graphs according to application-specific equations. Horkoff and Yu [HY16] propose a framework for iterative, interactive, agent-goal model analysis for early requirements engineering. The framework uses a set of qualitative evaluation labels made up by the modeler and assigned to intentions to express their degree of satisfaction or denial. The process starts by assigning labels to intentions related to the analysis question, and propagates the labels through the model links, either forward or backward, using defined rules. In contrast to these works reasoning about goal satisfaction, my method employs crowdsourced evidence to reason about goal prioritization.

Murukannaiah et al. [MKTS15b] describe Arg-ACH, which combines arguments and analysis of competing hypotheses (ACH), an analytical method, to attach evidence to a goal model. My method and Arg-ACH are complementary in that Arg-ACH employs critical questions in argumentation schemes whereas I employ high-level Canary queries to extract evidence. Further, Arg-ACH helps find conflicts but I prioritize goals. Elrakaiby et al. [EFM18] propose a refinement calculus for requirements engineering (CaRE) for solving this problem, which takes into account the typically dialectic nature of requirements activities. The calculus casts the requirement problem as an iterative argument between stakeholders and requirements engineers, where posited requirements are attacked for being ambiguous, incomplete, etc. and refined into new requirements that address the defect pointed out by the attack. In contrast, my approach gives requirements practitioners a more passive role of merely consuming the ongoing debates in social media as opposed to being active participants in the discussion. My approach takes advantage of already existing and continuously and independently created datasets. The propose approach doesn't address the problem of incentivization of the stakeholders to engage in discourse with the practitioners or approaches at handling the potentially overwhelming volume of data generated from successful attempts at crowdsourcing.

### 2.3.5 Enriching Goal Models

Horkoff et al. [HAC+16, HAC+17] perform a systematic review of existing literature to illustrate the state of the art in the field and paint a roadmap for the promising future directions. They classify papers in the field of requirements engineering into 11 different broad topic types. The topics of greatest relevance to my work are *extension* and *proposals*. The describe extensions as containing publications which focus on some concept(s) which is not a named language or method being added to goal model (e.g., capabilities, commitments). Proposals are described as any publication that proposes something new, e.g., a language, extension, integration, algorithm. They find that 91% of RE papers contain a proposal and 42% contain an extension. This indicates that goal modeling is an active area of experimentation and ideas similar to the one I am proposing as an extension are commonplace and widely accepted by the community. It also validates my intuition that goal modeling would be suitable venue for making an impact to practitioners since they would be used to the idea of existing specifications of the modeling approach.

Ali et al. [ADG10] propose a framework for adding context to goal modeling and introduce contextual goal models. As context they take qualitative specifications of the surroundings of a stakeholder and use it to extend the Tropos goal model. In a much similar way I am trying to enrich requirements modeling, but with quantitative metrics gathered through collective user interaction by a crowd of users.

### 2.3.6 Requirements Prioritization

Adequate requirements prioritization is crucial to any software project. A frequently studied [ASIM14] research approach for requirements prioritization is the Analytic Hierarchy Process (AHP) [Saa99]. AHP is a multi-criteria decision making technique based on a pair-wise comparison approach. The process calls for creating a matrix where each row and each column is labeled by the requirements so that each element of the matrix can be identified by a pair of requirements. The user then proceeds to give a numeric preference to each pair of requirements in their personal favor of one or the other. AHP has an obvious issue of is the lack of scalability and increase in complexity as the number of candidate requirements increases [PSRB07].

Ramirez et al. [MRMK+17] propose a tool supported methodology that combines end user feedback with domain expertise. Similar to my approach, they enrich requirements with quantitative information. I explore additional automation with the use of subjective logic.

Davis [Dav03] proposed the requirements triage as the process of determining which requirements a product should satisfy given the time and resources available. He suggests gathering all stakeholders together in one location and conducting a vote to determine requirements importance based on quantitative information. Other quantitative voting approaches exist, such as the Hundred-Dollar Test [Lef03], which is an application of the cumulative voting principle, where stakeholders are given a budget of 100 votes and they "bid" on requirements they consider most pertinent. In contrast to these approaches, my 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.

Gamification is another approach for deeper inclusion of end-users in RE. For example, Fernandes et al. [FDR+12] present iThink, a game-based collaborative tool called iThink that aims at improving the participation in a requirement elicitation process. iThink takes advantage of the association between "gamification" concepts and the six hats of thinking method for collecting both new requirements and feedback about existing ones and for presenting the requirement elicitation process in a form of a collaborative game. Snijders et al. [SDB+15] present REfine, a gamified online platform for requirements elicitation and refinement by involving a crowd of stakeholders: users, developers, analysts, etc. REfine allows users to suggest needs, comment on needs and other comments, branch needs and vote for needs and comments. There lacks full understanding of the incentive for participation in online discussions of crowds of users in social media. Gamification might be an excellent route towards understanding and improving such incentives. A recent study by Lombriser et al. [LDLB16] evaluated application of gamification to RE and found that gamification improves the quantity and quality of elicited requirements and has a positive effect on motivation and participation. They developed the gamified requirements engineering model (GREM) that relates gamification, stakeholder engagement, and RE performance. To gather findings they evaluate GREM by building an online gamified platform for requirements elicitation, and they report on a rigorous controlled experiment where two independent teams elicited requirements for the same system with and without gamification. Canary complements these works by leveraging richer content creation by the crowd via annotations and enabling sophisticated queries of the content.

A more recent approach Kifetew [KMP+17b] et al. study the application of gamification to requirements prioritization. They combine gamification and automated reasoning techniques to support collaborative requirements prioritization in software evolution. Garuso [KG17] is a collaborative platform in which gamification is used to provide additional motivation to users to participate in the requirements engineering process. The authors claim to have found statistically significant differences between different algorithms controlling single game elements on the contributions of stakeholders to the prioritization of requirements. Tools are also emerging to support the gamification of requirements prioritization, such as DMGame [KMP+17a]. DMGame exploits game elements to engage distributed stakeholders to contribute to the overall decision-making process. AHP and Genetic Algorithms are used as key component of the game engine, which enables an iterative prioritization process Canary methodologies can be run in parallel with such gamification strategies. Gamification seeks to increase collaboration and improve the accuracy and amount of quantitative interaction produced in online platforms. Such improvements would also be highly beneficial to the methods proposed in this thesis. In the future, a unification should be sought between novel attempts of improving incentives (gamification) of online collaboration and novel approaches to systematically leveraging the content produced by online collaboration (Canary).

# 3

# Analysis of Online Discussions

Online discussions between current and past users of a software product can be used to ameliorate the field of requirements engineering. In particular I would like to examine the discussions that take place in a threaded forum, such as Reddit.com. Previously in **RQ1** I formulated a research question that aimed to seek out, examine and verify any underlying naturally occurring structures of such online discussions with the purpose of using them as an annotation schema that will add structure to unstructured data.

At the time of conducting this research no record existed of adequate examination for an answer to this research question. With no prior knowledge and expectations I decided to apply a grounded theory method of study of the target source of information. For the purposes of reaching an answer to **RQ1** I explored a vast number of online discussions and manually examined their structure, recording my observations and intuitions as I went. The beginning of this chapter presents my findings. The results of this undertaking are a set of abstractions that can be used to annotate online discussion and capture valuable information. The annotations I came up with are inspired by abstractions borrowed from requirements engineering and argumentation. Such applications of grounded theory can be subject to many cognitive biases because of the nature of how they are conducted. In order to address this threat to validity in the second half of this chapter presents the process I undertook a quantitative study in aims to verify my intuition in more rigid evidence and further assert my answer to **RQ1**. I undertook a quantitative analysis on a concrete number of hand-picked online discussions. Using the annotations previously elicited using grounded theory two software engineers, myself and a more experienced academic colleague, systematically annotated 5 online discussions. In the later parts of the chapter I present, graph, examine and discuss the results of my efforts.

Finally in this chapter, I address **RQ2** by conducting a study on the scalability of acquiring annotations by employing Amazon Mechanical Turk (MTurk) users. In the study I commission lay users (a crowd) to annotate online discussions with the annotation schema I introduced earlier.

## 3.1 Case Study

Google maps was chosen as the target application for this case study because it falls into the category of a product that has been developed using requirements driven by the market. Google Maps is an application that is intended to be used by a crowd of users. It is popular and has a large user base, which enabled me to find a large number of discussions to study. The choice to only use one application for this case study is justified in my intuition that it might be valuable to examine how different discussions on the same topic can be combined together to form a picture that is larger than any one discussion can give. The choice of one specific application for this study doesn't take away from the ability to generalize the findings.

### 3.1.1 Target Application: Google Maps

Google Maps is a Web application developed, provided, and maintained by Google Inc. It provides information about geographical regions and sites worldwide with a focus on road and traffic systems. In order to achieve this, Google Maps combines aerial satellite imagery with conventional road maps. I list some of the services offered by Google Maps below:

**Route Planner** enables users to plan a route and receive visual and voice directions on how to make their trip from one location to another.

**Driving Directions** combines *Route Planner* and information about the current location of the host device to provide real time instructions to the user. It uses a "next action" basis, for example "Turn left in 50 meters and you will have reached your destination." as synthesized voice message combined with on-screen instructions.

**Voice Command** allows users to execute commands on Google Maps verbally, with minimal physical interaction with the host device. The user is able to navigate through most of Google Maps features by voice and receive synthetic voice as response, thus enabling the driver to keep their eyes on the road at all times.

### 3.1.2 Social Media Outlet: Reddit.com

Reddit is a Web forum. It uses crowd-sourcing techniques to distribute the work of content creation, moderation, and filtering to its community. People may submit their own content and vote on other user's submissions. Content with more positive votes is ranked higher, and therefore shown to a larger portion of the community. Registered users on Reddit are called *redditors*. Reddit has a complex hierarchical structure of entities, shown in Figure 3.1. The community of redditors is divided into subreddits, where each subreddit represents a general topic of discussion, such as programming, music, or football. Redditors can submit content to a subreddit in the form of submission. A submission holds information in the form of natural language, image or hyperlink. Redditors can post comments on a submission and other redditors can post comments on existing comments. All comments are stored and displayed in a hierarchical, waterfall-type structure, as seen in Figures 3.2 and 3.3, so the progress of any discussion can be easily observed. Other relevant features of Reddit are:

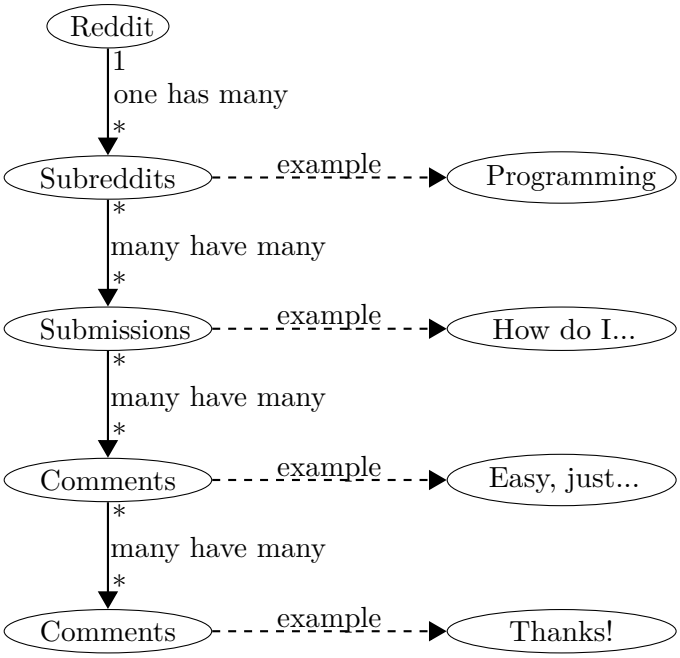



Figure 3.1: A hierarchical view on how Reddit artifacts relate to one another

**Vote system** Users can show their approval or disapproval of a submission or a comment by giving it an up or down vote, respectively. Votes are combined with other data to create the *karma* score (measured in points) of each submission and comment. Karma score is used to order posts within a subreddit and comments within a post, so that the most relevant to the topic and popular information is showed to the largest amount of people.

**User reputation**. Each redditor has karma score to serve as reputation. User karma is affected by the amount of community approval for comments and submissions the user has created.

## 3.2 Results

Here I explore the results of my observations of the data. I present two forum discussions, found on Reddit, shown in Figures 3.2 and 3.3. I added annotations to the figures (A, B, C, and so on) in order to put emphasis on key elements, such as comments that shape the discussion on the forum. Below I analyze the discussions from a requirements perspective.

### 3.2.1 Overview

*Figure 3.2* shows that people are having difficulty with finding ways to make the navigation in Google Maps repeat its last instruction to the user. Deeper in the discussion that requirement is broken down into two smaller, easier to achieve requirements: tap the microphone-shaped button and then input a relative voice command. The participants in this discussion don't comment on having to tap the button, but they make an *observation* regarding relative commands—relative commands are unintuitive. This observation has

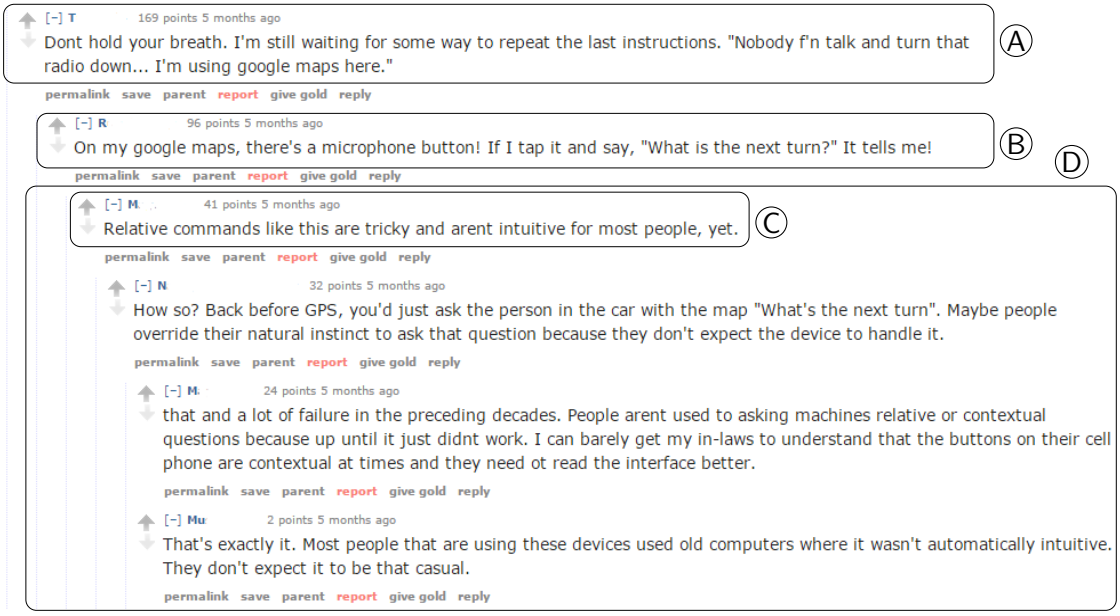



Figure 3.2: A discussion about Google Maps on Reddit. Participants identify and debate on the unintitiveness of relative voice commands in the application.

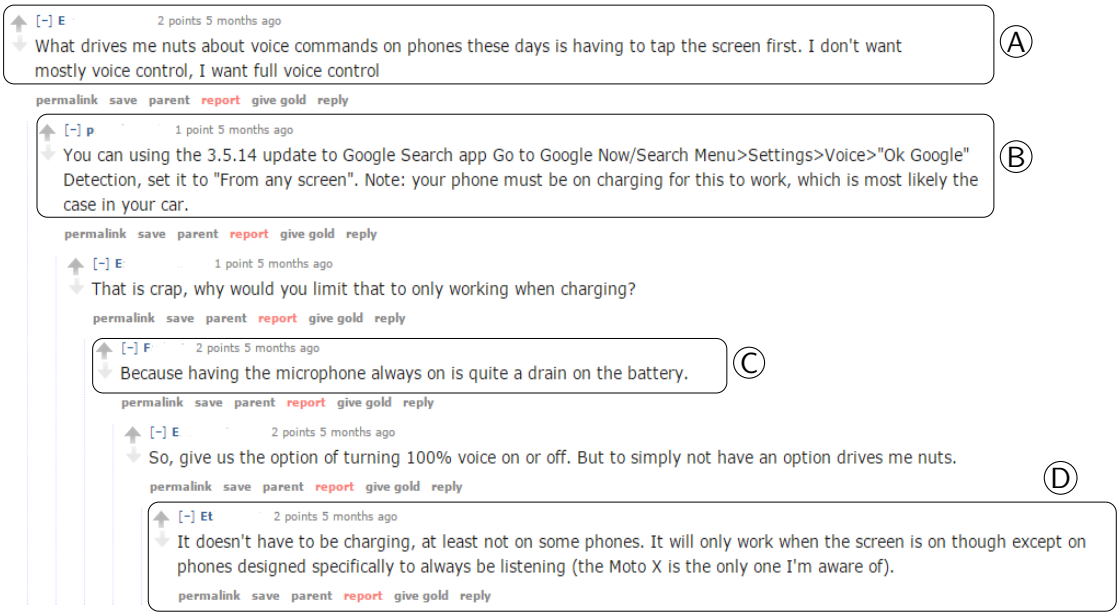



Figure 3.3: A discussion about Google Maps from Reddit. Participants are discussing the plausibility of having voice control enabled at all times

a highly negative influence on the system and has strong support in the community, as illustrated by the comments expressing sentiment towards it.

*Figure 3.3* shows that the community would prefer having an alternative to tapping the microphone for activation of *listening* mode (state in which the device awaits a voice command). A redditor proposes voice activation of *listening* mode by saying a key phrase as a possible solution. Another redditor mentions how another application developed by Google Inc. solved the problem (OK Google), and also states an observation on the solution, that the device must always be charging when the solution is used. Further in the discussion another observation is made about voice activation of *listening*

mode—screen must be on.

### 3.2.2 Key Findings

During the analysis I found several recurring artifacts and made note of some aspects of social media that can be of use to requirements engineering.

**Requirements.** In Figure 3.2.A, I found a usability problem. The problem is then broken down as the discussion progresses and the reason for it is identified through end-user experience and observation of the system. Requirements on social media are not expressed in a clear-cut sense, unlike in traditional requirements specification documents found in software projects. Often requirements need to be derived from expressions of ideas, needs or desires.

**Observations.** The preliminary analysis showed that end-users express tacit knowledge in online forums in the form of observations: system behavior, requirements interdependencies, and others. They explicitly state information that may not be apparent to developers. Such information is referred to as unknowns [Ger13], [SS13]. Discovery of unknowns is challenging, as it is highly exploratory. End-user feedback is an excellent candidate source of unknowns because end users have know-how and experience with interacting with the system. Examples of observations can be seen in Figures 3.2.C, 3.3.B, and 3.3.D.

**Lack of awareness of existing solutions.** The comments in Figure 3.2.A and Figure 3.3.A both show examples of a requirement statement from redditors. They both briefly describe and request functionality. In the next comments, found in Figure 3.2.B and Figure 3.3.B, other redditors inform them that the required functionality is already implemented in the application. The unawareness of existing functionality can used as a surrogate for identification of user interface problems. It shows that end-users are unable to find out how to use specific functionality that is readily available to them.

**Expressions of sentiment in comments.** End-users express sentiment towards solutions or requirements in online forums. Figure 3.2.D shows a string of comments expressing sentiment and support towards the non-intuitiveness of relative commands. Negative sentiment towards a solution can be treated as a sign that the solution is not good enough and positive sentiment towards a requirement can be seen as indication that the requirement must be considered with high priority. Sentiment is a direct representation of end-user satisfaction (towards solutions) and desire (towards requirements).

**Community support.** Social media uses collaborative filtering to filter content. Voting on Reddit can be used to measure the support of the community towards a requirements artifact using quantitative metrics (upvotes and karma score). For example the comment in Figure 3.2.A, asking for a way to repeat the last instruction from navigation, has 169 points (karma score). In contrast, the comment in Figure 3.3.A, asking for voice activation of *listening* mode, has 2 points. Repeating the last instruction receives larger support from the community, and can potentially be treated with priority.

## 3.3 User Feedback and Goal Modeling

During the analysis I labeled sections of data that contain requirements-related artifacts. In this section, I highlight some gaps in current requirements modeling techniques in relation to information that is available in social media. For the sake of discussion, I use goal modeling [YM98] as a concrete example.

Goal modeling is intended to introduce a high-level abstraction over system requirements that is easy to communicate to stakeholders, without committing to a technological solution. Goal models are especially suitable for exploration in the context of my work since they can be used to represent a high-level view of arguments as well. In Figure 3.4, I present a simple goal model of the domain knowledge found by studying Figures 3.2 and 3.3. In Figure 3.4, ellipses and clouds represent goals and soft goals respectively. Arrow labeled "-" and "- -" represents negative and highly negative contribution, respectively. I present my key findings below:

### 3.3.1 Quantitative Information Loss

The design of current requirements modeling techniques, including goal modeling, doesn't allow metadata to be freely added to the representation. This leads to loss of information available in the source. If represented the information can be useful to requirements elicitation, prioritization and others.

**Community support and sentiment.** There is no visual representation of community support for an artifact in the model. Also, community support doesn't *propagate* through the relations of the target artifact. In Figure 3.2.A we see a usability issue, because users are unable to find out how to repeat the last navigational instruction from Google Maps. In Figure 3.2.C the fault for the issue is placed on an the observation that relative commands are not intuitive. In Figure 3.2.D, there are comments showing support (positive and negative) for the observation, but the goal model cannot represent that information. The link between the usability issue of repeating the last instruction and the unintuitiveness of relative commands is not made evident in the model.

**User reputation.** Earlier I discussed the personal karma score of each user. It can be used as a quantitative metric for further filtering and elicitation of entities. User reputation is also based on collaborative filtering. Users with high reputation are users who have provided valuable to the community input in previous discussions. It is a new means of adding context to individuals in an otherwise homogeneous crowd of end-users.

**Controversial entities.** Entities causing high controversy, such as large discussions or fluctuating approval and sentiment, can be detected and brought forward in the model. An interpretation of such information can detect usability issues for certain demographics. The comments in Figure 3.2.D indicate agreement that the age of users is a factor when considering the intuitiveness of relative commands. The amount of social media content (comments, votes) generated in regards to a specific artifact can be used as a metric for controversy. A formula can be conceived, which takes into account the ratio of positive (votes, comments, supports) interaction and negative

(votes, comments, rebuttals) interaction. In the case that such a ratio is gravitating towards balance, it would be reasonable to make assumptions about the controversy of an entity. In such a circumstance the entity would have attracted a similar amount of both polarities of interaction, thus making it controversial. The specifics of a proposed equation that would capture such relationships mathematically are defined in further research in Chapter 4.3.2.

### 3.3.2 The Bigger Picture

Combining discussions on the same topic, such as Google Maps, into the same model gives a much richer understanding of the problem domain. The model in Figure 3.4 is the result of combining domain knowledge gained from Figures 3.2 and 3.3. The combined model gives a better picture of the system. It represents a larger portion of the domain and shows how requirement entities are intertwined on a larger scale. Relationships between the entities are well represented using goal modeling, but I noticed that interaction on the forum is structured as argumentation. Further research may prove argumentation-related notions to be more informative for modeling of interdependencies of entities from multiple sources.

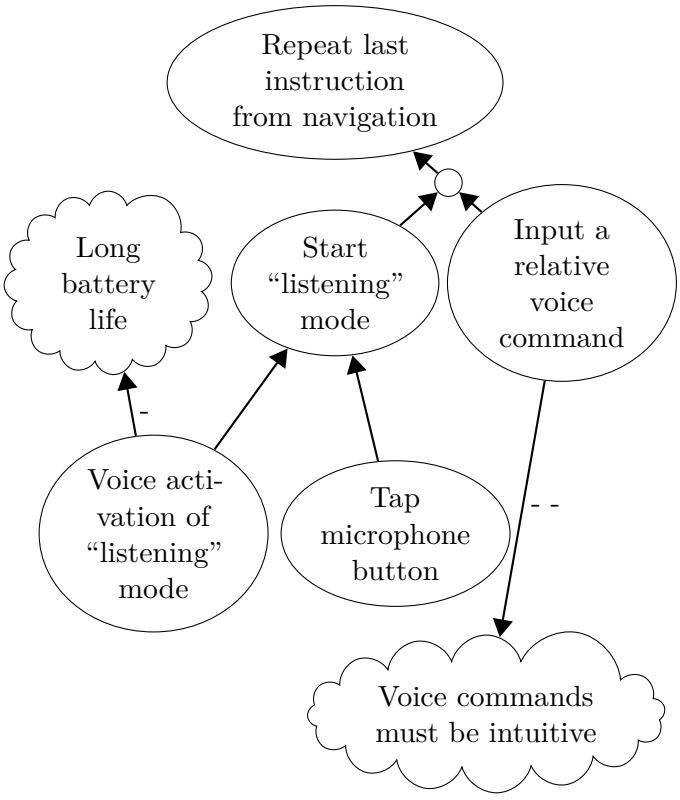



Figure 3.4: A model representation using goal modeling techniques of the domain knowledge gained regarding Google Maps from Figures 3.2 and 3.3

## 3.4 Literature

From then Lubras et al [LPR93] proceed to identify the different challenges that affect each type of development. For the purposes of this thesis I will focus on market-driven

development. The participants in the survey conducted by Lubras et al. reported issues with forming a mental image of who the customer is, producing a product that is not responsive enough to customer needs and prioritizing requirements. The absence of a commissioned customer renders standard requirements elicitation and validation methods inapplicable. Companies are reported to have little to none customer or user involvement and requirements are self-invented [KHS98, JGW10]. There is need for balance between developer and user elicited requirements [SSK99]. With time, market change, and company evolution the project might lose up-to-date domain knowledge that regular contact with end-users provides.

Modern elicitation techniques are better at extracting an abundant set of requirements. This introduces new problems, such as filtering and prioritization [KDR+07, WRB11]. Developers gather requirements into potentially huge repositories in natural language. The lack of context associated with those requirements creates difficulties for developers in interpreting and prioritizing them. Studies call for more structured (context-aware) input of requirements-related data [RB05].

User feedback is a candidate for context-rich source of requirements [PM13b].

Ali et al. [ADG10] propose a framework for adding context to goal modeling and introduce contextual goal models. As context they take qualitative specifications of the surroundings of a stakeholder and use it to extend the Tropos goal model. In a much similar way I am trying to enrich requirements modeling, but with quantitative metrics gathered through collective user interaction by a crowd of users.

A formalization of argumentation is proposed by Chopra and Singh in Colaba [CS11], where they propose tool-based assistance for collaborative design of cross-organizational processes. Argumentation is presented as a process of six steps that can be repeated recursively until agreement is reached. The steps described can be observed in a forum discussion. A similar notation can be used for specification of argumentation in social media.

## 3.5 Discussion

This thesis presents the results of a preliminary analysis of feedback in social media. I carried out a case study about Google Maps on Reddit.com, which is a Web forum. I set out to highlight and bring forth important artifacts relevant to requirements engineering in the interactions between users on forums. I also critically evaluate the effectiveness of goal modeling techniques to capture the information contained within feedback in social media with the purpose of enhancing requirements modeling with notions that capture user interactions.

In order to achieve that I manually explored Reddit.com in search for discussions between users regarding Google Maps. In those discussions I discovered several key findings regarding the information of value to requirements engineering available in social media. My key findings include *requirements*, *observations made by end-users* (explicit statements of tacit knowledge), *lack of awareness of existing solutions*, *expressions of sentiment towards previously discovered artifacts* (users tend to express their agreement that a requirement is necessary or that a solution is not good enough), and *community support*

(well defined, pertinent artifacts are recognized by the community using the collaborative voting system).

I extracted two discussions from the forum and built a goal model of Google Maps using the domain knowledge gained solely from those discussions. Comparing the information captured in the resulting model to the amount of information available in the source, I concluded that requirements modeling can benefit from enhancements that capture user interactions by taking advantage of several key elements of social media that involve quantitative metrics. Examples of that are *community support and sentiment*, *user reputation*, and *controversy of entities*. In addition, I also discovered that the interaction between users on the forum is structured as argumentation and that combining multiple sources of domain knowledge in the same model enriches my understanding of the domain.

My findings can be used to motivate research in designing a requirements model. The notions supported by the model can capture user interaction as context to traditional requirements artifacts, such as goals or requirements. Social media is a highly interactive environment, where users are encouraged to evaluate, rate, and vote on each others contributions to the community. Such a model would require abstractions that can be informed by my findings.

Argumentation emerges as a strong candidate source of abstractions that can capture the qualitative requirements-related information in social media, and can be expanded to accommodate the quantitative metrics mentioned above. Further research is needed on the applicability of argumentation towards understanding user feedback in social media.

The interactivity of social media makes the content within it dynamic. As a result, using it as a source of information can have some interesting implications on the value of the information I extract. New content emerges in social media at a fast paste, so how long after data is created it is still relevant? When is it most relevant? How do the dynamics of social media affect the design of a requirements model intended to capture interactions between the users of that media?

Analyzing the vast quantity of information in social media manually proved to be challenging and error-prone. An interesting area for future research is tool support for assistance and automation of the analysis of feedback in social media. Support can be provided for the finding, filtering, and extraction of information, as well as reducing the amount of manual labor needed for its analysis by applying natural language processing techniques.

## 3.6 Evaluation

In order to back up my observations with measurable evidence I conducted a study to determine the frequency and type of entities I can expect to see in online discussions about software applications.

In Section 3.2.2 I presented a list of key findings that were the result of my application of a study executed grounded theory in search of a naturally occurring structure in an unstructured data source. Those findings present my subjective qualitative intuition that that the content of online discussions about software applications gravitates around a number of abstract concepts. Those abstract concepts have been previously studied in

various branches of science, namely argumentation and requirements engineering. In order to test my observations so far I designed and conducted an empirical study of a number of selected online discussions found from social media. For concreteness I picked two abstractions from each field I identified to be involved in the structure of discussions. From requirements engineering I picked *requirement* and *solution*, and from argumentation I picked *support* and *rebuttal*. The list of annotations is simpler than some requirements models that have been conceived before in previous publications. In this study I do not consider conflict, priority, positive and negative contributions, and assumptions [BPG+04, vL09]. Including these concepts in Canary would require considering their meaning in the context of user discussions. The trade-off of richer requirements models is more complex annotation. Enrichment of the annotation schema can be easily achieved once the initial proof of concept of my designs proves valuable.

### 3.6.1 Methodology

As study units, I selected five online discussions from two social forums, each involving user discussions about Google Maps and related software applications. Table 3.1 summarizes these discussions (actual discussions are available in an online appendix [KMCS17]). Initially, I planned to include more discussions from Google Forums. However, I noticed that unlike Reddit discussions which involved a rich variety of user interactions, Google Forum discussions were much simpler in that they involved a few requirements and a huge number of supports (Table 3.2). Thus, I decided against adding more Google Forum discussions since that would not likely influence potential conclusions.

Table 3.1: Summary of the online discussions I employed in my study

| Disc. | Comments | Words | Source | Discussion title |
|---|---|---|---|---|
| 1 | 141 | 6034 | Reddit | Feature Google Now should add: "Nearest x that is still open" |
| 2 | 184 | 13333 | Reddit | There is no way to properly save an address in Google Maps |
| 3 | 79 | 3975 | Reddit | Google Maps should use your average walking speed from Google Fit to calculate walking times |
| 4 | 261 | 8654 | Reddit | Google maps should have an "I need gas" feature... |
| 5 | 605 | 11310 | Google Forum | Make 'avoid tolls' option sticky otherwise directions are wrong |

To determine the validity of my intuition, two software engineers, acting as expert annotators, annotated each discussion in three rounds. It is worth noting that one of the software engineers was myself and the second was another academic colleague that was also involved in the design and implementation of the research paper that

this experiment was published in. In the first round, the two experts annotated the discussions independently without seeing each others' annotations. In the second round, they saw each other's annotations and updated their annotations, independently. In the third round, the experts discussed their annotations, resolved differences, and settled on one set of annotations as the *ground truth*.

### 3.6.2 Results

Table 3.2 summarises the experts' annotations. Note that the two experts were in complete agreement after the third round.

Table 3.2: A summary of expert annotations for each discussion

| Disc. | Requirement | Solution | Support | Rebuttal |
|---|---|---|---|---|
| 1 | 26 | 20 | 17 | 13 |
| 2 | 53 | 50 | 27 | 29 |
| 3 | 12 | 3 | 10 | 17 |
| 4 | 51 | 65 | 16 | 12 |
| 5 | 21 | 7 | 127 | 3 |

### 3.6.3 The Structure of Discussions

In this section I will attempt to illustrate the naturally occurring underlying structure of discussions using data plots.

#### The Shape of Discussions

Discussions come in many different shapes. Below I attempt to illustrate this by creating a graph of all the discussions.

I discovered a strong distinction between two major shapes of discussions. In my research I considered two types of online forums.

First, I consider flat forums. An example of such is Google Forum. It doesn't allow for direct responses to existing comments and all comments are displayed on the same level, i.e. flat. They all respond to the original thread. I provide a graph illustration of my example, Discussion 5 from Table 3.1, discussion in Figure 3.5. The green nodes are comments, Red nodes are users, The purple node is attached to the root comment to signify it is root.

A strong pattern emerges from this graph. As is evident, it shows a central root node, and all other nodes emerge as responses to it. Flat discussions don't allow for complexities in interaction.

Second, I consider complex discussions. Such discussions allow for complex interactions between the participants in the form of a comment/reply structure of the discussion. The discussion forms an interaction tree of arbitrary depth in which complex interactions can observe observed. Such functionality in discussions is currently possible in Reddit. In my

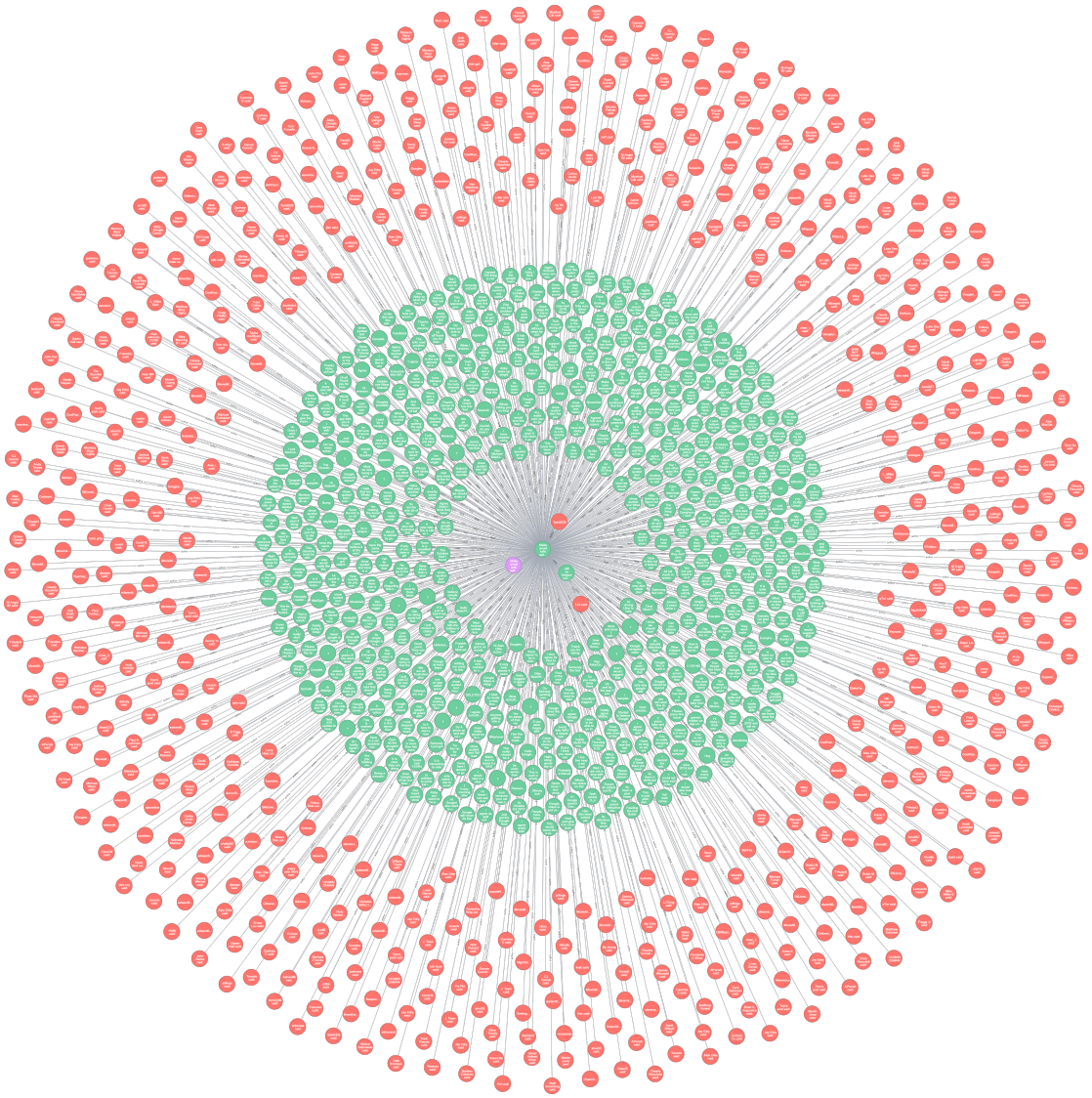


Figure 3.5: A graph that captures a flat discussion on Google Forum. Discussion 5 from 3.1. The green nodes are comments, Red nodes are users, the purple node is attached to the root comment to signify it is root.

example such discussions are, Discussions 1, 2, 3, and 4 from Table 3.1, and their graphs can be seen in Figures 3.6, 3.7, 3.8, 3.9.

From these graphs it becomes immediately visible that more sophisticated platforms such as Reddit allow for more complex discussions to emerge. In the next section I will examine the content of discussions in order to gain an understanding of whether the sophistication of the platform and the complexity of discussions were observed to be related to the variety and quality of content that emerges from discussions.

### The Content of Discussions

In this section I'll discuss the content of discussions. First I'll begin by creating a parallel between the quality content found on flat discussions and complex ones. As a metric of quality of content I'll consider the amount and variety of objects of interest I identified in the annotation task. As objects of interest I consider requirements, solutions, supports,

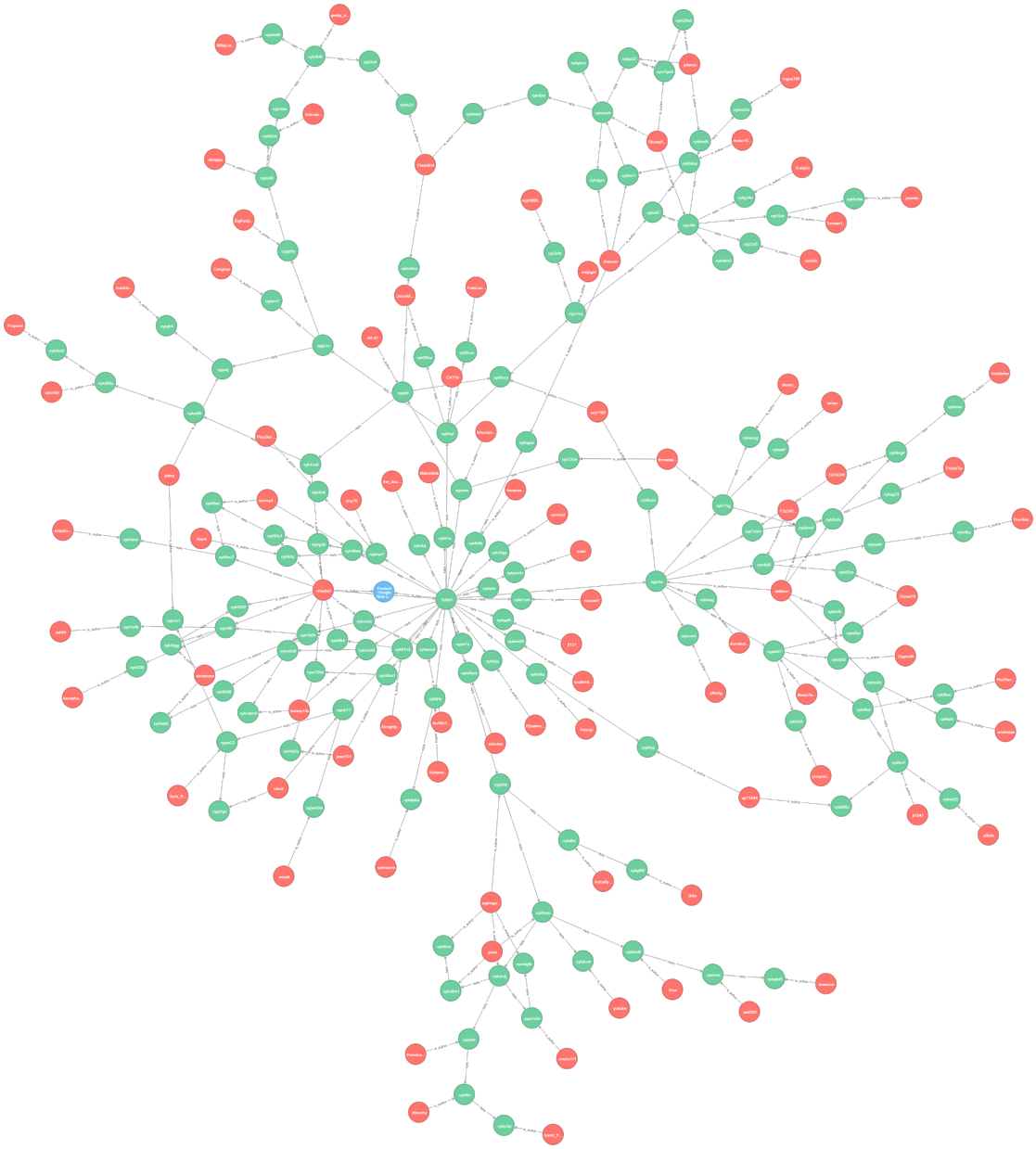


Figure 3.6: A graph that captures a complex discussion on Reddit. Discussion 1 from 3.1. The green nodes are comments, Red nodes are users, the blue node is attached to the root comment to signify it is root.

and rebuttals.

In Figure 3.11 I show a bar chart of all the occurrences of each object of interest found in Reddit discussions in that I have considered in this research. In Figure 3.10 I show the same bar chart with a discussion found in Google Forums.

The comparison of these two plots clearly shows that the flat discussion (Google Forums), seen in Figure 3.10, attracted mainly supporting statements, a small number of requirements and an even smaller number of solutions and rebuttals. Since there is a single level of interaction all the supporting arguments are for the original requirement that started the discussion. There is very little depth to this discussion and it it reasonable to argue that there is very little valuable information stored in such discussions.

It is also worth noting that for all discussions I examined the majority of comments were

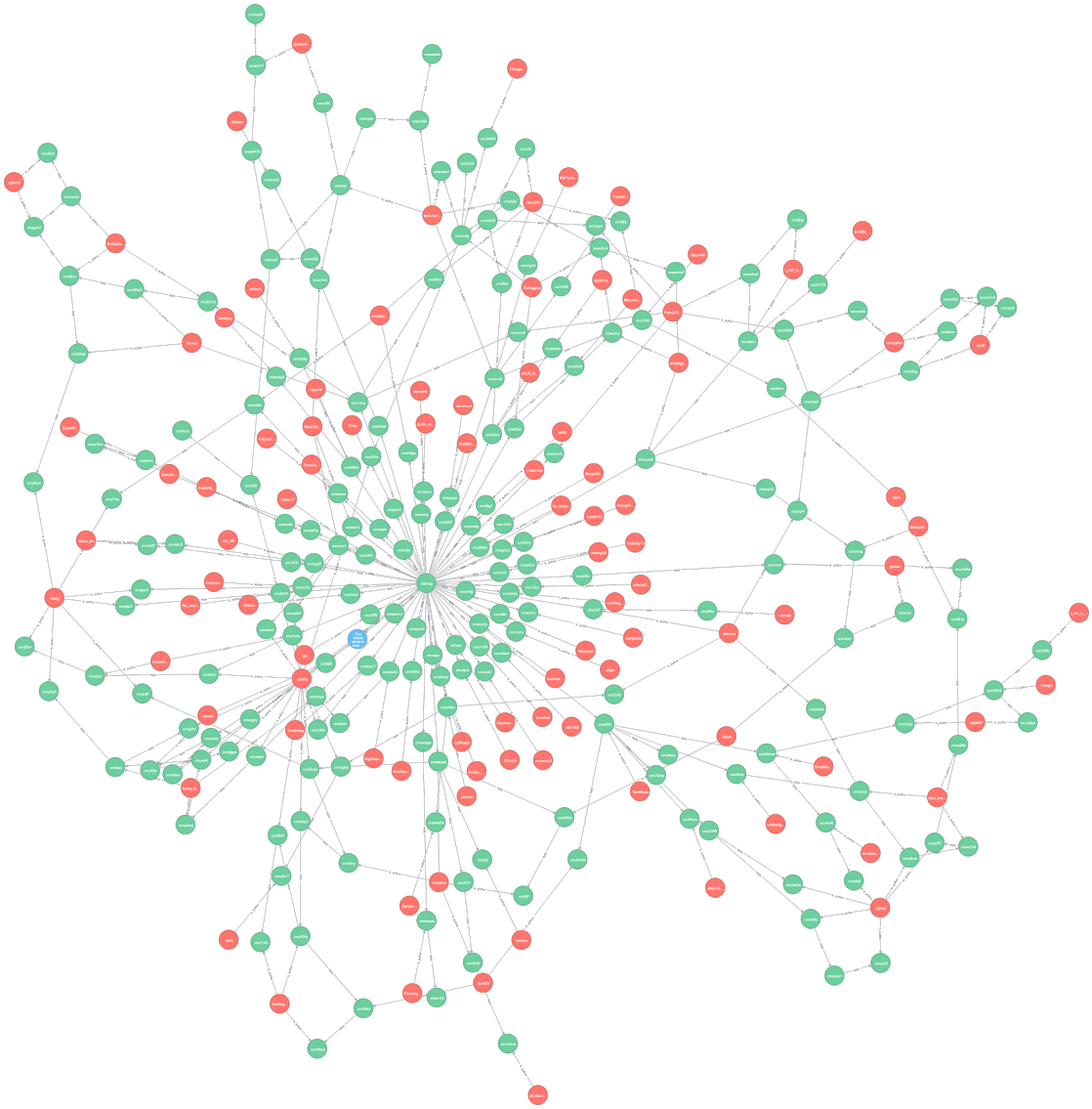


Figure 3.7: A graph that captures a complex discussion on Reddit. Discussion 2 from 3.1. The green nodes are comments, Red nodes are users, the blue node is attached to the root comment to signify it is root.

still labeled as "noise". This is not surprising. Online discussions are driven by lay people participating in an ad-hoc way for no obvious incentive. The valuable content that is labeled as interesting is unimpaired by the noise. These results, reporting on the ratio split between useful and not useful shouldn't discourage any future efforts for leveraging social media. Noise is an immutable part of the information found in such crowd-based sources.

Since complex discussions are interesting I'd like to present a few more plots of their quantitative measurements. In Figure 3.12 I show how the number of comments is distributed in the various depths of the discussion. The number of comments appears to steadily decrease as the depth increases.

What would be even more interesting is to see how the depth affects the distribution of objects of interest in Reddit discussions. In Figure 3.13 I show a distribution of requirements throughout the various depth levels of the discussion in Reddit. The same

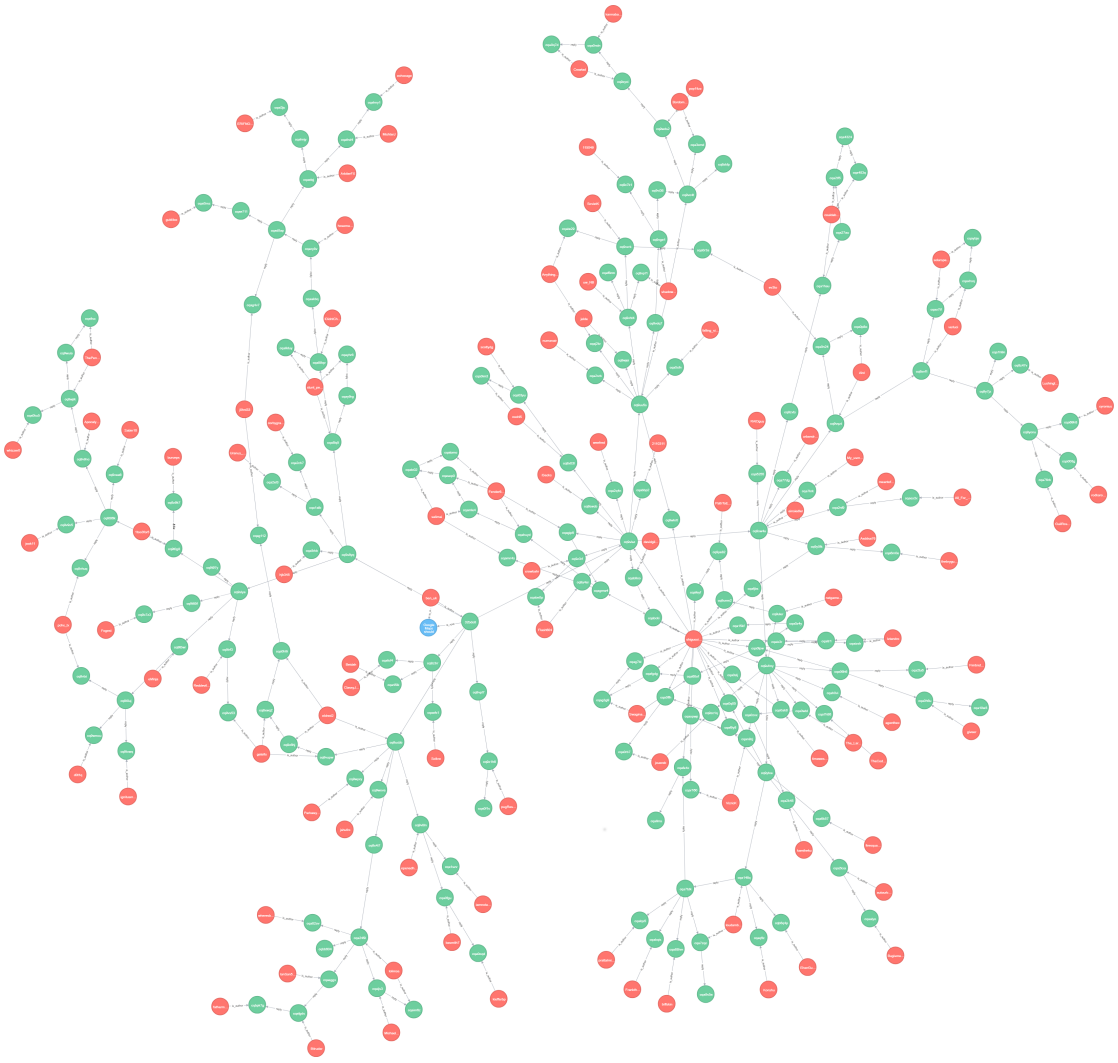


Figure 3.8: A graph that captures a complex discussion on Reddit. Discussion 3 from 3.1. The green nodes are comments, Red nodes are users, the blue node is attached to the root comment to signify it is root.

can be found for solutions, supports, and rebuttals in Figures 3.14, 3.15, and 3.16, respectively.

From my exploration of the quantitative side of online discussions I can conclude two major points.

- complex discussions promote more varied interaction and thus better quality of content
- in order to extract the maximum value from the information found in online discussions tools are needed that can leverage the recursive nested structure of complex discussions

## 3.7 Acquiring Annotations

In order to address **RQ2**, I conducted a study on the scalability of acquiring annotations by employing Amazon Mechanical Turk (MTurk) users. The study is my answer to the question of how effective crowd-sourcing techniques are at scaling up the annotation of online discussions about software products.

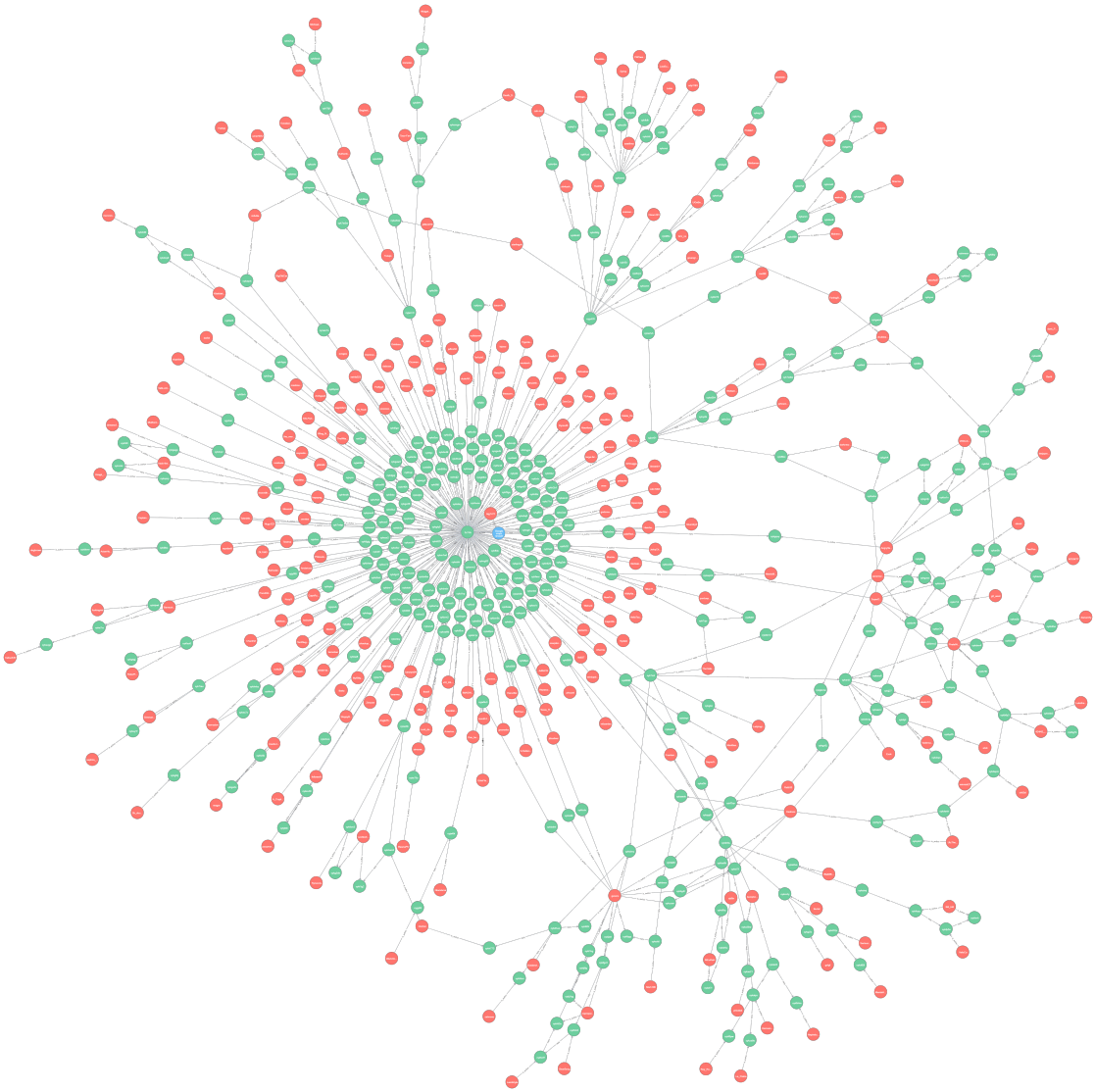


Figure 3.9: A graph that captures a complex discussion on Reddit. Discussion 4 from 3.1. The green nodes are comments, Red nodes are users, the blue node is attached to the root comment to signify it is root.

As study units, I selected five online discussions from two social forums, each involving user discussions about Google Maps and related software applications. Table 3.1 summarizes these discussions (actual discussions are available online. Initially, I planned to include more discussions from Google Forums. However, I noticed that unlike Reddit discussions which involved a rich variety of user interactions, Google Forum discussions were much simpler in that they involved a few requirements and a huge number of supports (Table 3.2). Thus, I decided against adding more Google Forum discussions since that would not likely influence potential conclusions.

We selected four discussions from Reddit and one from Goole Forums for two reasons. First, the Google Forums discussion was much longer than the other Reddit discussions. Second, we found Reddit discussions to have richer interactions between users (including a variety of requirements, solutions, supports, and rebuttals) than the Google Forum discussion (including a variety of requirements, solutions, and supports, but lacking rebuttals).

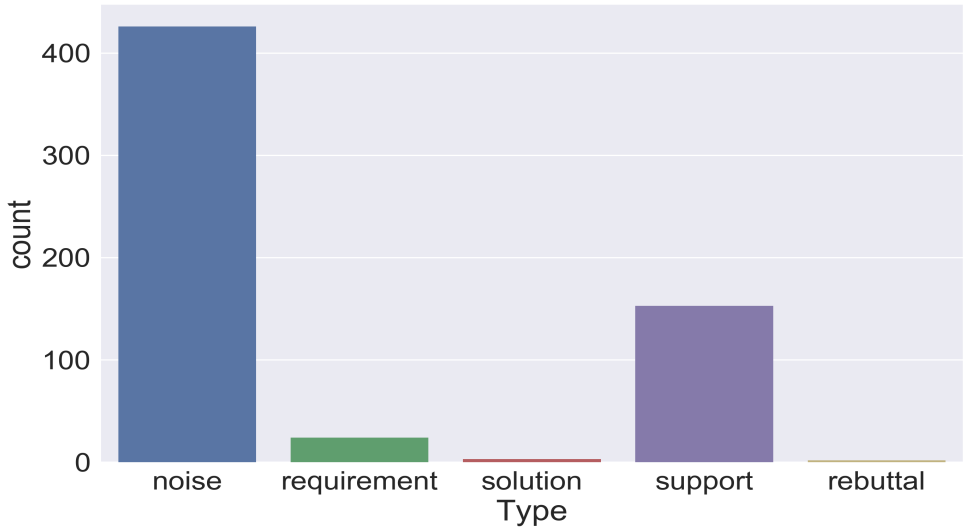



Figure 3.10: A sum bar chart of all total of all the occurrences of each object of interest found in Google Forums.

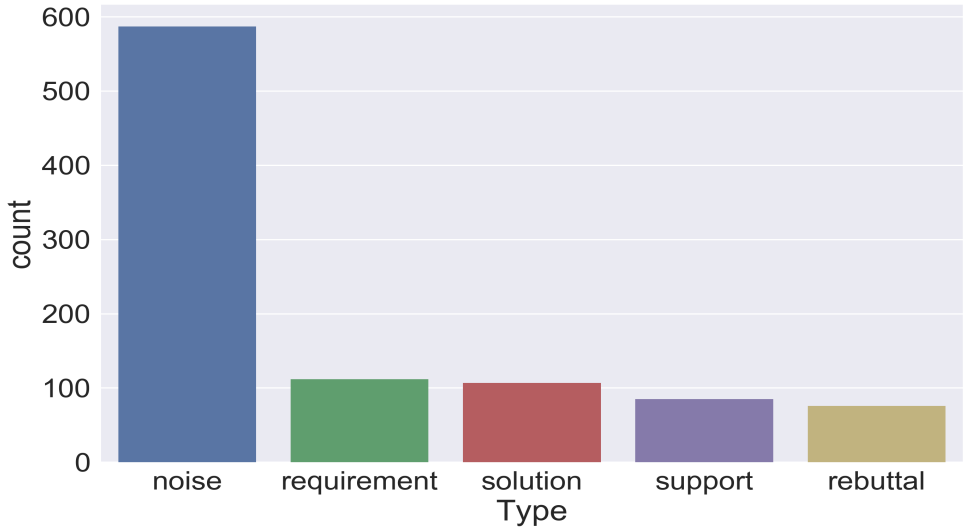



Figure 3.11: A sum bar chart of all total of all the occurrences of each object of interest found in Reddit.

A typical online discussion starts with a topic (e.g., a question) and several top-level threads fork from it. These discussions, in general, tend to be long. A lay user may require several hours to annotate one full discussion. For tasks crowdsourced to lay users, Cheng et al. [CTIB15] find that requiring users to perform smaller parts (*microtasks*) of a large task (*macrotask*) yields higher output quality, completion rate, and experience than requiring workers to perform the macrotask (my tasks are similar to the ones Cheng et al. study in that both require analytical and language understanding abilities). Accordingly, I decided to split the discussions in my study into smaller chunks of approximately equal length ($\mu = 1882$ characters and $\sigma = 229$ words). In this case, I use $\mu$ as denoting as the constant at which the total character length for each task is aiming. Similarly, $\sigma$ is

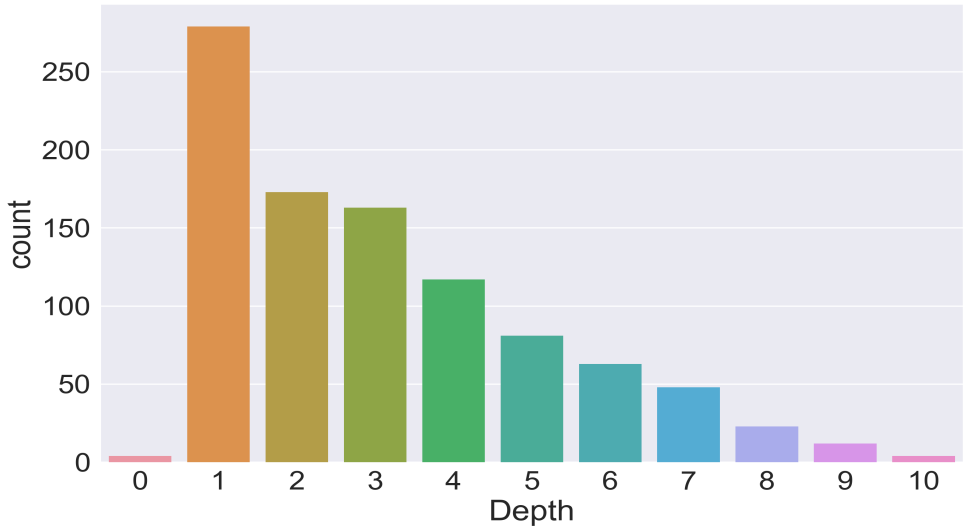



Figure 3.12: A sum of the total number of comments in depth throughout the discussions in Reddit.

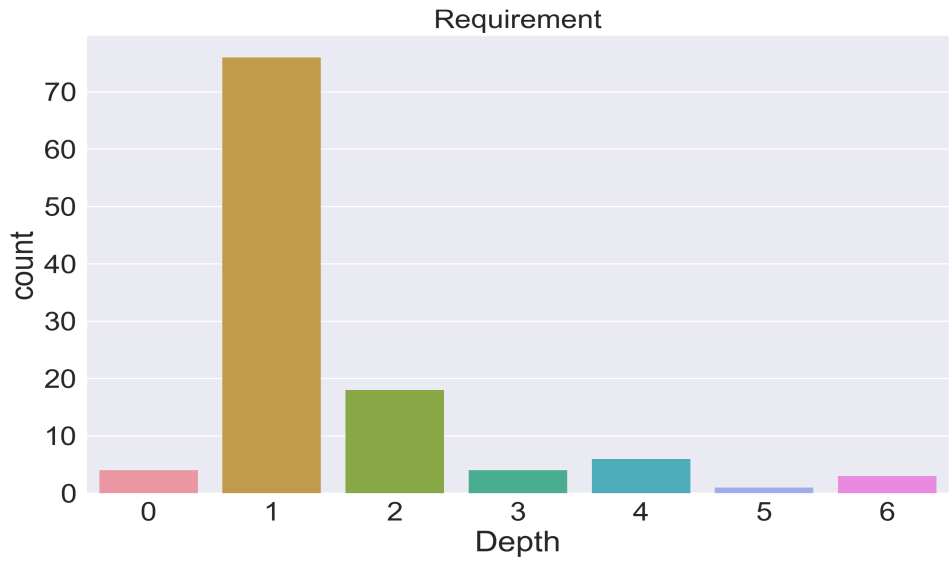



Figure 3.13: A bar plot of requirements found in each depth level throughout the discussions in Reddit.

the constant which gravitates the word count of all tasks. Each microtask in my study included the topic of a discussion followed by one or more threads about it. I decided not to split the top-level threads so as to preserve the context of interactions (I note, however, that some interactions may span top-level threads, but from my experience such instances are rare).

Splitting the discussions yielded 38 microtasks. I sought to acquire annotations from two users for each microtask so as to get a reliable estimate. Accordingly, I launched 76 HITs (human intensive tasks) on Amazon MTurk and collected annotations from 44 unique MTurk users. I restricted participants to be from majority native English speaking countries; UK, USA, Canada, Australia, and New Zealand. Overall, I rejected

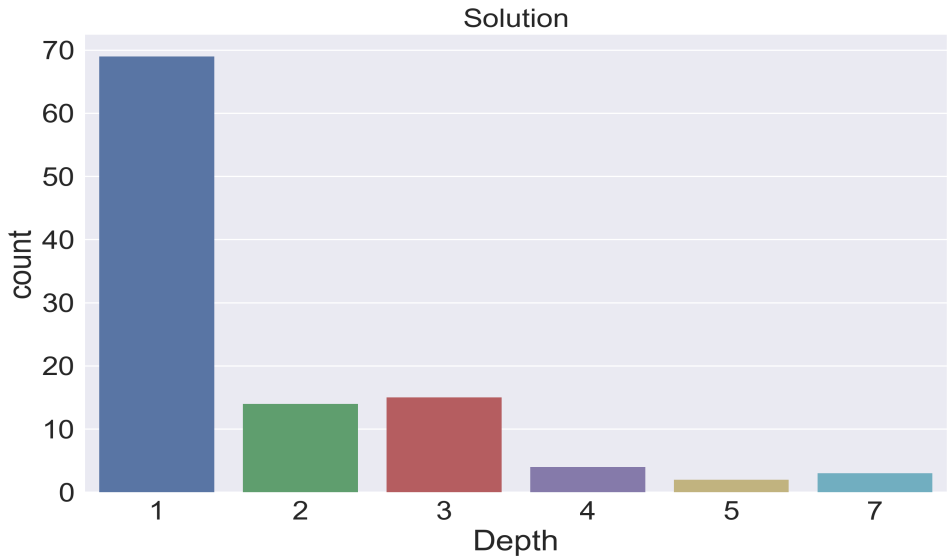



Figure 3.14: A bar plot of solutions found in each depth level throughout the discussions in Reddit.

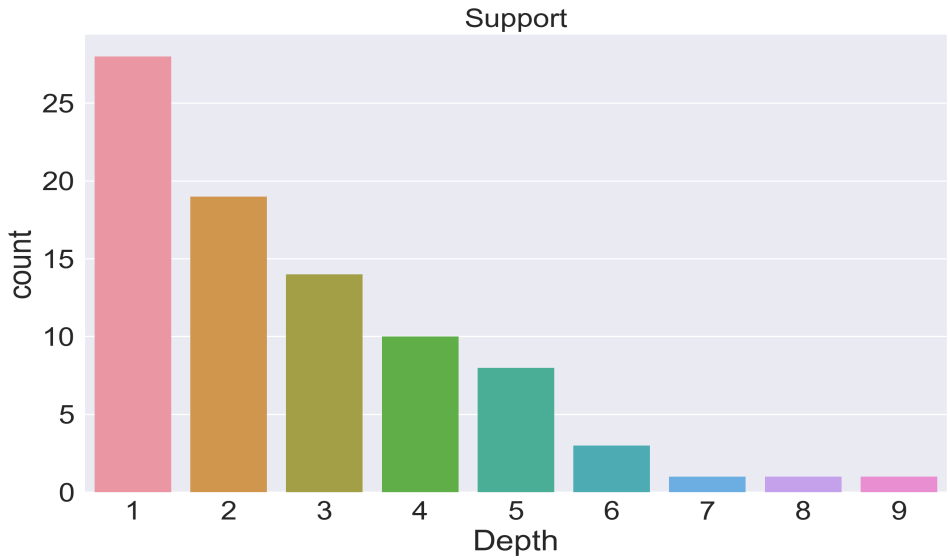



Figure 3.15: A bar plot of supports found in each depth level throughout the discussions in Reddit.

three HITs as incomplete. I paid USD 4 for each successful HIT. My study was approved by the Ethics Board at Lancaster University and I received an informed consent from each participant. Complete questionnaires and ethics documents are available the online appendix [KMCS17].

As part of the study, the MTurk users were asked to (1) answer a pre-survey about demographics, (2) complete a main task, and (3) answer a post-survey about time, difficulty, and the understanding of instructions and the concepts.

In the main task, first, I asked users to read about the core concepts of requirements, solutions, supports and rebuttals from a document I provided. The document included multiple examples for each concept from online discussions. Next, I asked users to download a PDF file consisting of a discussion chunk and annotate the file via Adobe

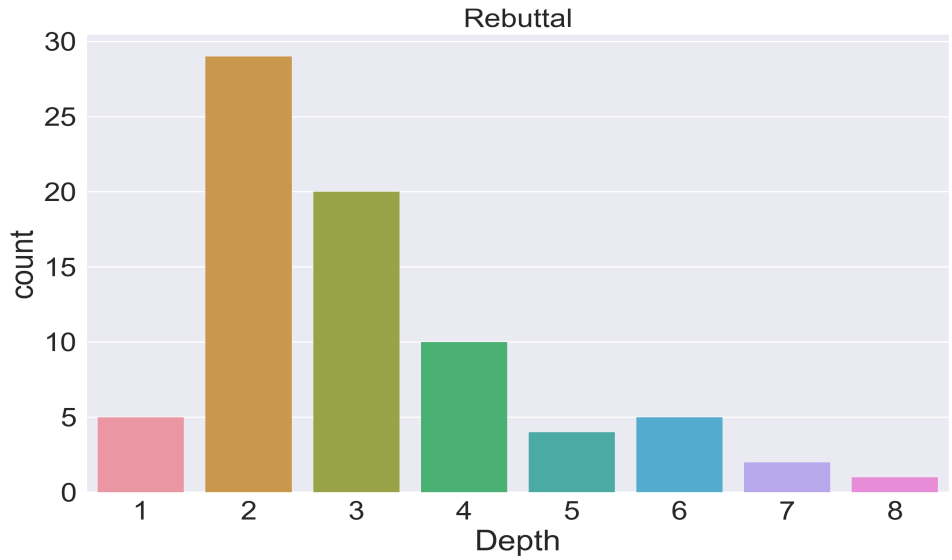



Figure 3.16: A bar plot of rebuttals found in each depth level throughout the discussions in Reddit.

Acrobat Reader. I provided instructions to install the software, a tutorial on performing annotations, and an example PDF file with relevant annotations. Finally, I asked users to upload the annotated PDF file via an URL I provided.

I instructed users to select any comment or a part of a comment in the provided discussion and annotate it as one of the four entities: *requirement*, *solution*, *support*, or *rebuttal*. For simplicity, I asked users to annotation a piece of text with at most one entity. Further, for each annotation, I asked users to indicate their confidence in the annotation on a Likert scale of *very low*, *low*, *medium*, *high*, and *very high*.

In the post-survey, I asked participants to report the time they spent for the main task (i.e., time for reading and annotating the PDF file with the discussion, excluding the time for pre- and post-surveys). I also asked participants, the difficulty of the main task, and how well they understood the concepts of requirements, solutions, supports, and rebuttals, each on a scale of 1 (very low) to 5 (very high). Finally, I asked users to provide additional comments, if any.

### 3.7.1 Ground Truth

A key challenge in evaluating the accuracy of annotations is establishing a standard for comparison. For this experiment I use the dataset generated in Chapter 3.6. In summary, to establish the ground truth, two software engineering researchers, who were also involved in the design and implementation of the experiment, acting as expert annotators, annotated each discussion in three rounds. In the first round, the two experts annotated the discussions independently without seeing each others' annotations. In the second round, they saw each others' annotations and updated their annotations, independently. In the third round, the experts discussed their annotations, resolved differences, and settled on one set of annotations as the *ground truth*.

Table 3.2 summarizes the experts' annotations. Note that the two experts were in complete agreement after the third round.

### 3.7.2 Measures

After expert and MTurk users' annotations, I prepared a dataset for measuring the accuracy of annotations as follows. First, for each discussion, I collected all pieces of text that had an annotation (from experts or users). For each such piece of text, I assigned an *expert-label* and a *user-label* as follows. The expert-agreed annotation (requirement, solution, support, or rebuttal) of a piece of text is its expert-label. If a piece of text in the list was not annotated by experts, I assigned *none* (noise) as its expert-label.

Each piece of text that needed to be annotated was sent to two random members of the MTurk crowd. The workers were asked to annotate the text, but also provide a confidence score from very low to very high. For each piece of text, if the two users' annotations for the text match (irrespective of confidence), I assigned the corresponding annotation as the *user-label* for the piece of text. Further, for a piece of text such that the two annotations do not match, but one annotation has a confidence of high or very high, and is higher than the confidence of the other annotation, I assigned the annotation with the highest confidence as the user-label for the text. I assigned none (noise) as the user-label for all remaining pieces of the text.

For each discussion, given the list of annotated text, and their expert- and user-labels, I measured the accuracy of user annotations via the following metrics.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}; \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}};$$
$$F_1\text{-score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}};$$

where TP, FP, TN, and FN refer to true and false positives and negatives, respectively.

### 3.7.3 Results

#### Accuracy

Table 3.3 shows a table comparing the expert- and user-labels, aggregating counts across all five discussions. Table 3.4 shows the mean and variance of the per-discussion precision, recall, and $F_1$ scores, indicating that crowdsourcing is a viable approach for acquiring high-accuracy requirements-related annotations.

Table 3.3: A table comparing experts' and MTurk users' annotations (counts aggregated for five discussions)

| | | MTurk User Annotation | | | | |
|---|---|---|---|---|---|---|
| | | Req. | Sol. | Sup. | Reb. | None |
| Expert Annotation | Req. | 122 | 3 | 6 | 3 | 29 |
| | Sol. | 1 | 107 | 14 | 4 | 19 |
| | Sup. | 1 | 7 | 131 | 1 | 57 |
| | Reb. | 2 | 0 | 3 | 51 | 18 |
| | None | 3 | 9 | 7 | 8 | – |

Table 3.4: The accuracy of MTurk users' annotations

| | Precision | | Recall | | $F_1$ Score | |
|---|---|---|---|---|---|---|
| | Mean | Var. | Mean | SD | Mean | SD |
| Requirement | 0.94 | 0.05 | 0.71 | 0.12 | 0.81 | 0.10 |
| Solution | 0.72 | 0.33 | 0.70 | 0.22 | 0.70 | 0.28 |
| Support | 0.76 | 0.19 | 0.68 | 0.19 | 0.67 | 0.10 |
| Rebuttal | 0.76 | 0.18 | 0.67 | 0.33 | 0.68 | 0.28 |

First, I observe that a vast majority of MTurk users' annotations are true positives (diagonal elements in the table). Considering the complexity of the annotation task, I believe that the overall accuracy of user annotations is quite promising. Specifically, the precision for requirements annotations is very high. Second, I observe that the counts in the row corresponding to the *none* class-label are quite low in the table. This indicates that users are quite effective in distinguishing text containing requirements-related information from noisy comments (i.e., comments irrelevant for RE) that abundant in online discussions.

### Efficiency

Figure 3.17 (top-most box plot) shows the distribution of the durations reported by the users. The mean amount of time participants spent on the main task is about 35 minutes. However, the variance in time spent is high. A few users, in the comments, indicated that they were using the Adobe Reader software for the first time and I believe that there may be other such users in my sample. I conjecture that it is for such users that the main task duration is high.

I had a few returning users in my dataset ($n = 10$). The bottom two box plots in Figure 3.17 compare the durations reported by these users for the first task and the second tasks. I find that users take significantly less time the second time ($p = 0.02$; measured via Wilcoxon's ranksum test, excluding outliers). The test was executed by dividing the dataset into two — one consisting of the users first attempt and one consisting of their second attempt. This suggests that lay users can annotate online discussions in a time-efficient manner once they are familiar with the concepts and tools.

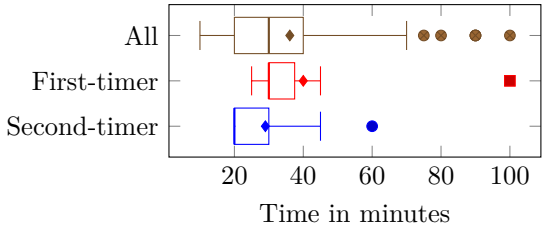



Figure 3.17: Times spent by MTurk users for one annotation task

Figure 3.18 shows the distributions of clarity and difficulty ratings reported by MTurk users. A vast majority of users rated their understanding of instructions and the concepts involved as high or very high. Further, the difficulty ratings suggest that the annotation task is of moderate difficulty.

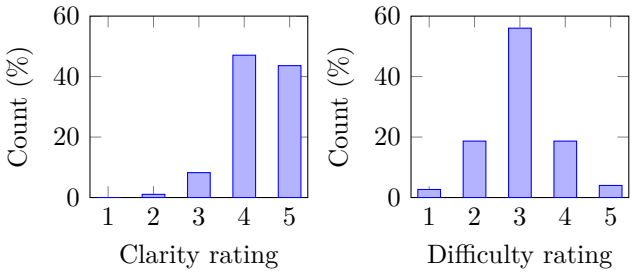



Figure 3.18: Clarity and difficulty ratings provided by MTurk users

Finally, I manually analyzed 24 comments that users provided. I found 11 comments to be conveying a positive, 8 neutral, and 6 negative comments. The negative sentiment comments mainly indicated that some annotations can be ambiguous; one comment indicated that annotations in Adobe Reader are clunky. However, many of the positive comments indicated the task to be fun and interesting.

### 3.7.4 Supplementary Documents

In the appendix .1 I show the supplementary documents used to conduct this study.

The documents presented in the appendix are in the following order.

- First I present the demographic results of the study.
- The consent form used for the study.
- The participant information sheet.
- The participant instructions used in the study.
- Next I present the questionnaire the the participants of the study were asked to fill out as part of the study.
- In order to conduct the study I had to acquire an ethics approval. Lastly I present the approval documents of the ethics committee in Lancaster University Faculty of Science and Technology.

### 3.7.5 Threats to Validity

I identify an internal threat to validity of my results. The two software engineering researchers, who are also involved in the design and implementation of the experiment, performed expert annotations were also part of designing the evaluation. Although the authors performed expert annotations systematically, in multiple rounds, there is a slight risk that the experts have subconsciously tried to second-guess how lay users are likely to annotate. Future studies can mitigate this risk by employing third-party experts.

As for the conclusions, the reliability of some of the self-reported metrics can be brought to question, such as total time take on the task, difficulty etc. Due to the self-reported nature of those measures they are inherently subject to bias. Additionally, a threat of

random heterogeneity exist. All participants are drawn from a random crowd of people by MTurk, so obviously their skill and commitments to the task vary.

In terms of construct validity my experiment is based on my findings from Chapter 3.6. Since those results are extracted as a result of grounded theory, many would consider them unreliable. To this critique of validity Wohlin et al.[WRH+12] emphasize that knowledge is not only statistical significance. My grounded theory findings were validated systematically using an experiment, which may be subject to bias as well, as examined next.

For this study I only targeted Google Maps as a target application for the discussions. This may prove to be a threat to validity in reliability. If another researcher was to take up this type of experiment with a different target application my results can prove to be unrepresentative for their choice of application and therefore unreliable. It would be interesting to see how well my results generalize over many different applications.

On a similar note, I also only used Reddit and Google Forums as a source of discussion, with a focus on Reddit. Variations in the social media sources might affect the results as Canary is closely tied to the raw data input. More variation can provide for more general results.

# 4

# Canary Methodology

## 4.1 Overview

A fundamental part of my contribution is a methodology called Canary that effectively creates a requirements-oriented view of online discussions. The methodology is derived as part of a design science approach to answering **RQ3**. The methodology is my answer to the question of how one can design an artifact that extracts pertinent information for RE practitioners from labeled online discussions.

The pivotal part of the methodology is a conceptual model that combines requirements-relevant and argumentation-relevant aspects of online discussions. This choice is not random, it comes from my work leading up to this, presented in Chapter 3. I first looked for patterns in discussions about software applications using grounded theory, my results shown in Chapter 3.2.2. Then proceeded to evaluate my findings using an empirical experiment, detailed in Chapter 3.6. Finally, I showed that these annotations of discussions can be achieved at scale using crowdsourcing in Chapter 3.7. The research I conduct is driven by a demonstration that it is feasible to transform online discussions into instances of this model by crowdsourcing labeled annotations natural language discussions about software applications found in social media.

In this chapter, I am interested in exploring what interesting things can be done with labeled online discussions about software applications. Canary contains a novel query language that leverages abstractions of requirements engineering and argumentation in interesting ways. Canary queries are to be executed over databases of labeled discussions. Canary queries are mapped to SQL queries by a fully implemented compiler. The queries are then ran on a MySQL database containing the raw information and a trace of the annotations gathered previously. Canary significantly reduces the effort required to run sophisticated queries on unstructured data source. Importantly, it goes significantly beyond technology such as IBM DOORS [IBM], where querying is limited to text searches.

An online discussion about a software application typically has the following structure. First, a user posts a question about how one may accomplish something with a software application, or describes a feature that one wishes the application had. I consider such

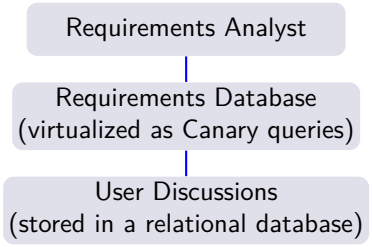



Figure 4.1: Canary realizes a requirements-oriented store over user discussions stored in traditional information stores. The view is realized via a mapping from Canary queries to SQL queries.

natural language descriptions—problems the users encounter and descriptions of what they expect the application to do—as requirements, broadly. Next, other users may respond with expressions of support for the user's comment or rebut it, e.g., by pointing out the infeasibility of what the user is requesting. Yet others may respond by proposing solutions that accomplish what the original user wanted. That is, a solution describes an existing means of addressing (solving) a requirement. In addition, users may employ upvotes and downvotes to indicate their support (or lack of support) for comments. In general, such a discussion would have a nested structure in that users may respond to any user's comment, not just the original poster's.

In a nutshell, online discussions include two kinds of information valuable to RE: requirements-oriented and social interaction or argumentation-oriented. It is reasonable to assume that extracting such information might help understand the challenges stakeholders face, which in turn helps formulate requirements and prioritize development tasks. However, current techniques are inadequate for extracting and utilizing information in online discussions.

## 4.2 Methodological Details

I present a conceptual model of online discussions and describe the subtleties in real discussions that I encountered.

### 4.2.1 Conceptual Model

My conceptual model relates elements of users discussions with elements of requirements and argumentation. Figure 4.2 shows main types of information Canary considers.

User discussions capture information related to social interaction between application *users*, who may be playing different *roles* in the discussion. Interaction is captured via *comments* (and their replies) and users' *votes* for comments, usually measured in a metric called *score*. Requirements information is captured via annotations to 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. Argumentation information is captured via annotations to 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.

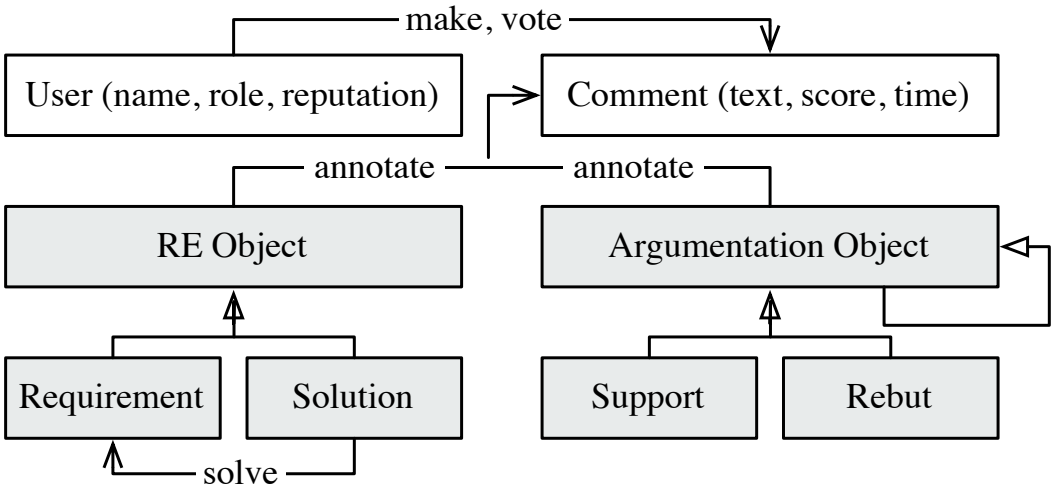



Figure 4.2: Conceptual model of information considered in Canary

### 4.2.2 Canary Methodology

Canary has four main steps, potentially performed by a requirements analyst.

1. *Acquire discussions.* Analyst acquires data from online discussion forums, where possible by using an API (e.g., from Reddit). The extracted data reflects the discussion accurately, including the flow of the discussion, user names, reputation, votes, and so on.

2. *Acquiring annotations.* At this stage, the discussion data contains no information about requirements or argumentation-related information. To obtain such information, the analyst sets up annotation tasks on Amazon Mechanical Turk (MTurk).

3. *Creating a database of discussions.* The analyst creates a relational database that reflects the schema of Figure 4.2 (the full schema is available in [KMCS17]) and loads the annotated discussion data into the database.

4. *Run Canary queries.* Analyst runs Canary queries against the database. Internally, the Canary compiler generates the appropriate SQL queries that can be run on the database.

### 4.2.3 Challenges and Assumptions

Online forums provide a flexible way of interaction between users. This creates several non-trivial challenges for acquiring requirements-related information from them. One primary challenge is how do I infer relationships between annotated comments (objects),

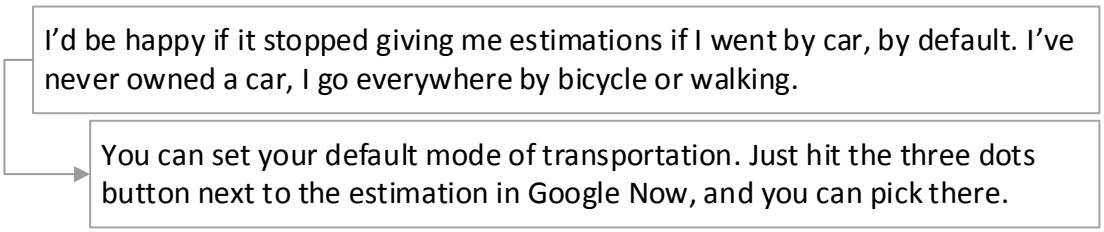



Figure 4.3: Example of a solution to a requirement

e.g., for the purpose of determining whether the sentiment for some requirement should count as sentiment toward another or whether the solution for one should count as solution for another. To be precise, the challenge arises from the fact that (1) there are no restrictions on what a user can say at any point in the discussion and (2) that discussion is of the form of a tree of unbounded depth.

In general, my strategy is to infer relationships between two objects only if one of them appears in a comment that is a reply (however deeply nested) to the comment in which the other one appears. This means that it is possible that I would miss relationships between two objects that do not appear along the same path in the discussion tree. This is the price I chose to pay for simplicity of annotations. Specifically, if I had wanted to capture all relationships regardless of nesting, the annotation task would have become much more difficult. In that case, the annotators would have to give a unique name to each object and indicate explicitly the relationships among them. However, because I decided to forego such relationships, annotators need not use labels or indicate relationships; they simply need to pick the annotation that applies best. I expect that arbitrary relationships would be rare occurrences in any discussion.

The implementation of Canary queries uses *propagation* to infer relationships among annotated objects that arise from nesting in the discussion. I propagate both semantics (e.g., if a solution is deeply nested in a requirement, I assume that the solution addresses the requirement) and sentiment, which is captured as a metric over the number of supports and rebuttals and up and down votes for an object. Propagating sentiment would mean that, e.g., if a requirement acquires some sentiment then its parent requirement (if any) will also acquire that sentiment. In general, a parent would acquire the sentiments of all its children.

In order to support my assumptions I show examples from real discussions of instances where some subtleties occur. When a solution is nested in the responses to a requirement, I assume that the solution is proposed to address the requirement (Figure 4.3). A requirement can occur in reply to another requirement. In this case, I assume that the former is *derived* from the latter (Figure 4.4 shows three derived ones). The notion of a derived solution is analogous (Figure 4.5).

Argumentation objects can be nested as well. Positive and negative argumentation about a requirement is shown in Figure 4.6 and Figure 4.7, respectively, and mixed argumentation is shown in Figure 4.8. A support for an object of interest expresses positive sentiment toward it. Supporting the supporting argument adds positive sentiment to the original object. A rebuttal to an object of interest expresses negative sentiment toward it. A rebuttal to this rebuttal adds positive sentiment toward the original object of interest. In unbounded nesting of rebuttals the sentiment may switch between positive

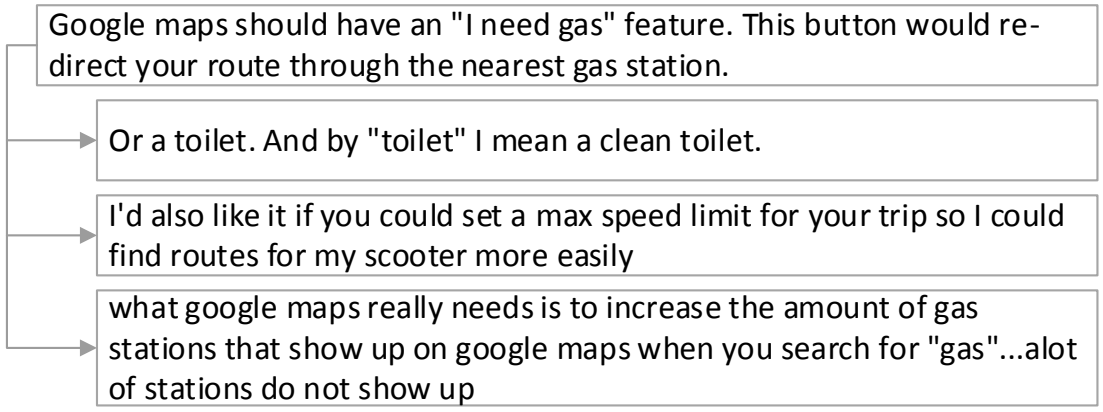



Figure 4.4: Example of derived requirements

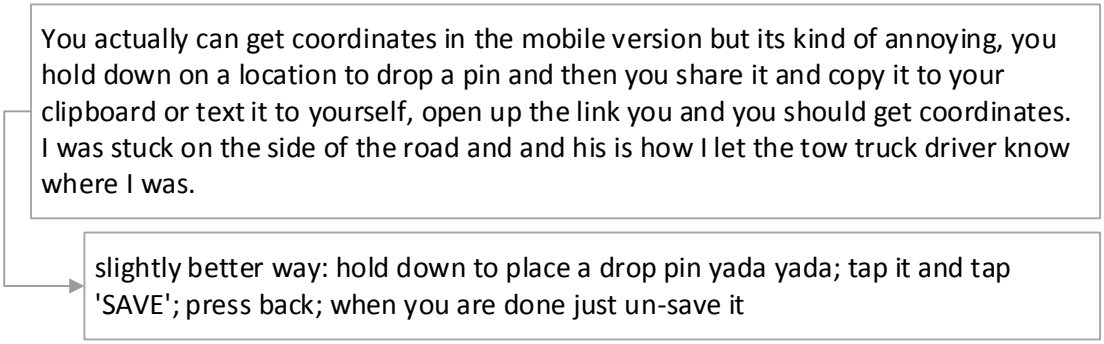



Figure 4.5: Example of a derived solution

and negative toward the root object. Finally, imagine a support to an object of interest is observed. A rebuttal of this support adds negative sentiment to the original object of interest. If another rebuttal is added then the sentiment becomes positive toward the original object. Analogously, supporting of rebuttals adds to the negative sentiment.

Gaps in nesting can occur. Imagine a comment with no object of interest, and then in a reply to this comment something interesting is observed. Such gaps provide a challenge in the way Canary infers relationships between objects of interest. Canary queries propagate to the end of the discussion tree to make sure all relations between objects are detected.

The flexibility of natural language allows for more than one object of interest in the same comment. I make the assumption that each comment contains one object of interest. During annotation I favor RE objects over argumentation.

Online discussion forums can differ. In this thesis, I include observations from two forums: Reddit.com (which allows for unbounded nesting of comments) and GoogleForums.com

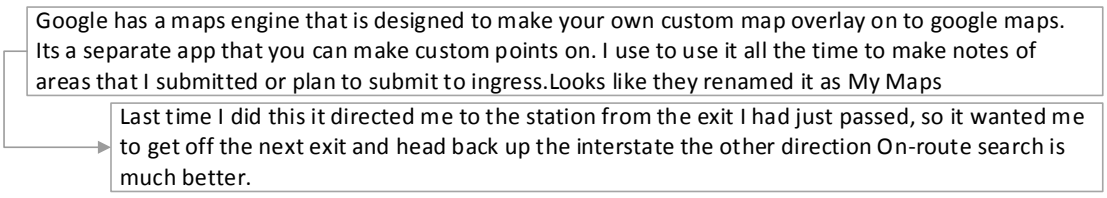



Figure 4.6: Example of a rebuttal to a requirement

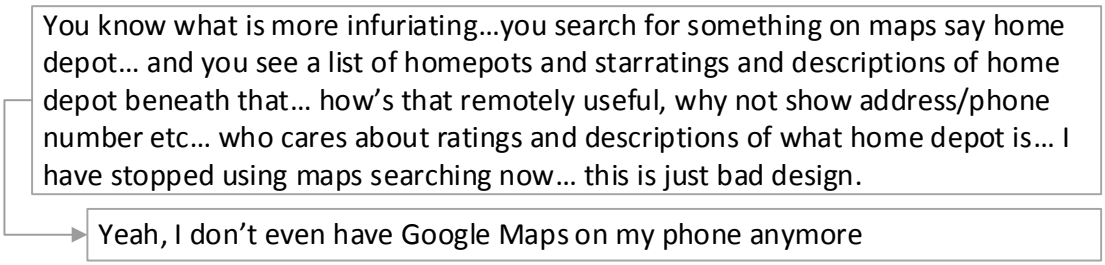



Figure 4.7: Example of a support to a requirement

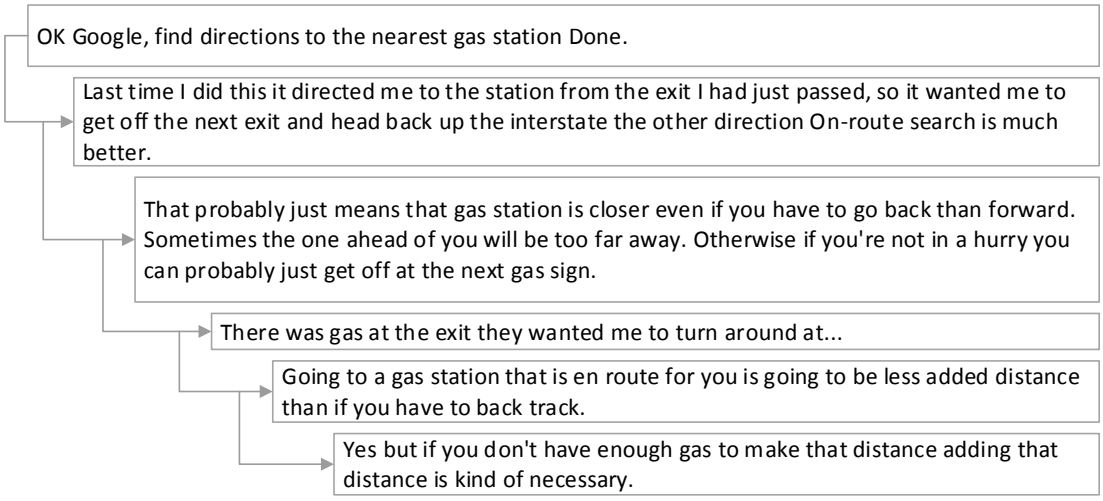



Figure 4.8: Example of mixed argumentation about a solution

(which only allows one level of nesting). Forums also store different details of the interaction that become available metadata. For example, Reddit stores the sum of votes for each entity in a metric called score. Google forums has a role associated with its users, such as Google community manager or a regular user. Reddit has a numerical value for reputation of user of the forum. Queries for missing metadata, for example a query for user role in Reddit, are handled gracefully by Canary. It runs the rest of the query as normal and ignores the condition, producing a warning.

## 4.3 Queries in Canary

In this section, I provide examples of high-level Canary queries and their results on a database of real online discussions from Reddit. Then, I provide the formal syntax and semantics of queries. For brevity, I only show an overview of the semantics (full details available in the appendix under "MySQL Canary Query Definitions".

### 4.3.1 Example Discussion

Figure 4.9 shows an example discussion that illustrates the framework. In the example, users are discussing features and requirements of Google Maps. The example is extracted from Reddit. John suggests a requirement about being able to save addresses in Google Maps. The requirement gets a score of 805. Mary expresses support for the requirement.

Henry and Patrick both propose solutions for the requirement (and get a score of their own, which is a result of summing upvotes and downvotes). Henry's solution attracts rebuttal comments in addition to a score. Patrick's solution attracts support and a score. In general, each comment may attract up and down votes (resulting in a positive/negative score), as explained above.

### 4.3.2 Queries

Below are examples of queries that leverage the framework. Each query is shown in a listing, and the output of the query is shown in a table immediately after. The queries are run on the example discussion shown in Figure 4.9, so the information returned is taken from there. The natural language text is shortened for space reasons.

Figure 4.10 shows a query to select all objects of interest annotated as requirements in the discussion.

The score of the "save address" requirement is different from the discussion above because of propagation. In the replies of "save address" there is one support with a score of 2 and one derived (nested) requirement with the score of 105. So, the score of these propagates up to the original requirement and is added to its own score, yielding 912.

Requirements are the only objects of interest that are stand-alone. A solution must address a requirement, a support must address something, and a rebuttal must rebut something. I will exemplify using solutions below; writing queries for support and rebuttal is analogous.

Figure 4.11 shows a query to select solutions for requirements that mention 'save address'. This figure shows to interesting elements. First, it shows an example of *conditions*, which can be applied to objects of interest to leverage the associated metadata such as score, natural language text (with support for regular expressions and fuzzy matching), user reputation or role, creation time, and depth in discussion. Second, it shows the value of propagation. In the original discussion 'long press...' has a bigger score than '...overlay engine', but with propagated values the score of '...overlay engine' increases greatly because of its two supporting arguments, while the score of 'long press...' drops from the rebutting arguments.

In Figure 4.12, I query for *popular* requirements; this is an example of what I refer to as *aggregator* queries. The idea of popular is to select those requirements which caused a high amount of positive interaction from the community. As positive interaction I consider score, supporting arguments, or any object of interest in the reply tree that has positive sentiment toward the original object of interest. Canary is able to propagate support through nested positive interaction entities (support of support, support of derived, etc.)

Popularity is measured as the ratio of the sums of quantitative data about positive and negative interactions.

$$\text{Popularity Score} = \frac{\sum_{i=0}^{n} \text{PV(i)} \times \text{n}}{\sum_{i=0}^{m} \text{NV(i)} \times \text{m}}$$

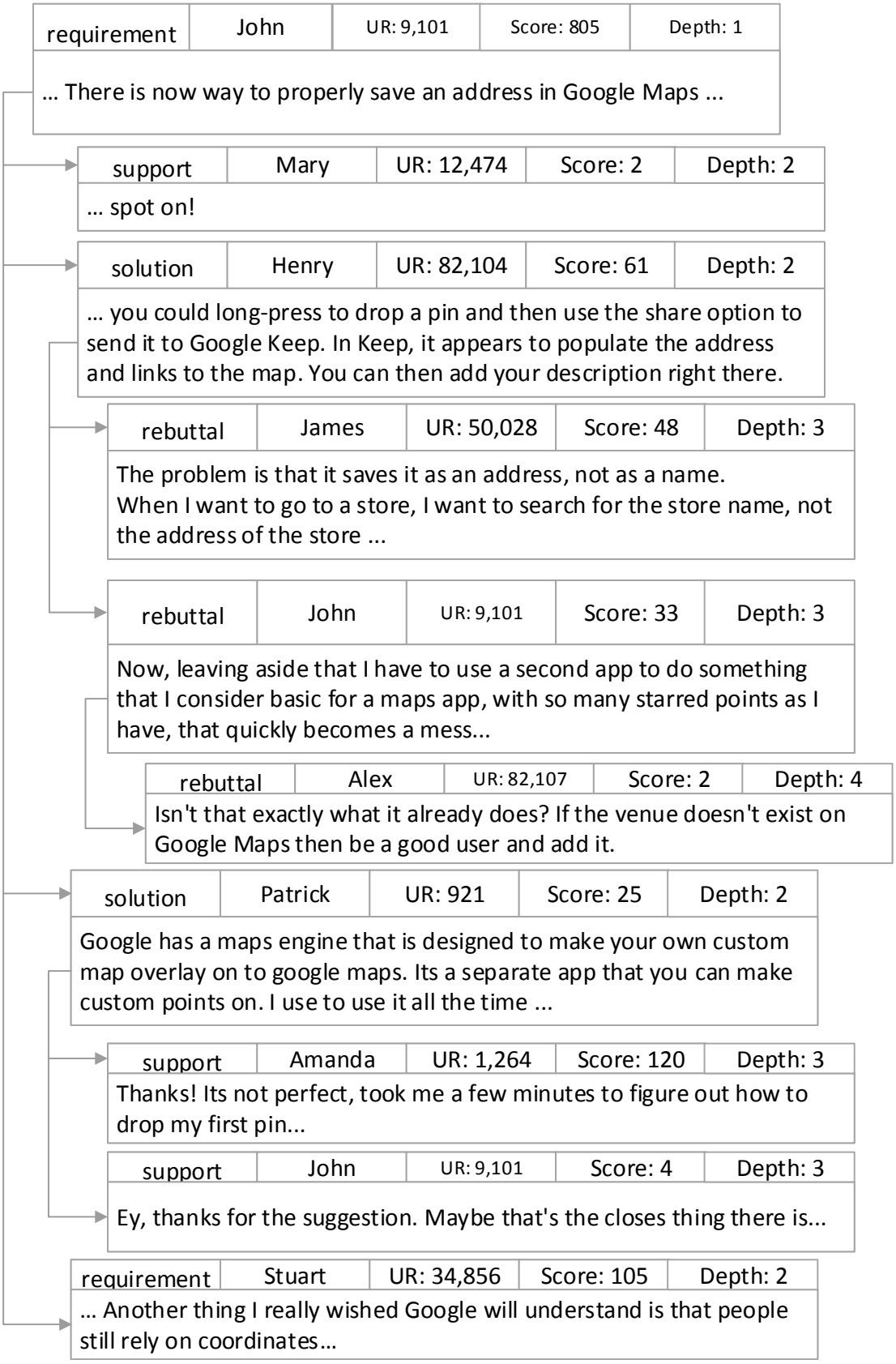



Figure 4.9: Example discussion following information framework

where $n$ is the number of children with positive sentiment and PV(i) the number of votes for a given child $i$, and $m$ is the number of children with negative sentiment and NV(i)

```
requirement
```

| text | annotation | user | UR | score | depth |
|---|---|---|---|---|---|
| save address | requirement | John | 9101 | 912 | 1 |
| coordinates | requirement | Stuart | 34856 | 105 | 2 |

Figure 4.10: Canary requirement query

```
solution (
  requirement where text regexp 'save address'
)
```

| text | annotation | user | UR | score | depth |
|---|---|---|---|---|---|
| long press and send to Keep | solution | Henry | 82,104 | -18 | 2 |
| custom overlay engine | solution | Patrick | 921 | 149 | 2 |

Figure 4.11: Canary solution query and results

```
popular ( solution ( requirement where regexp 'save address' ))
```

| text | annotation | user | UR | score | pop score |
|---|---|---|---|---|---|
| custom overlay engine | solution | Patrick | 921 | 149 | 298 |

Figure 4.12: Canary aggregator query and results

the number of votes for a given child $i$. In Table 4.1, I present the assumptions that can be made based on the ratio. The ratios are explained with the help of a threshold denoted by $\sigma$. I discuss possible data driven approaches to calculating thresholds in Section 4.3.3.

Table 4.1: Aggregator assumptions

| Pop ratio | Aggregator | Assumption |
|---|---|---|
| $<1 - \sigma$ | Unpopular | Negative interaction has prevalence |
| $\geq 1 - \sigma$ and $\leq 1 + \sigma$ | Controversial | Balance between positive and negative |
| $>1 + \sigma$ | Popular | Positive interaction has prevalence |

### 4.3.3 Threshold setting

The choice of threshold values has been a topic of intense academic interest for the recent years. Boucher et al. [BB18] conducted a systematic empirical comparison of different threshold setting methods for fault-proneness in software systems. In their experiment they approach the identification of the fault proneness of software units as a classification problem using various software metrics as features for each unit. They investigate three threshold calculation techniques that can be used for fault-proneness

prediction: ROC curves, VARL (Value of an Acceptable Risk Level) and Alves rankings. For comparison they use machine learning ideas. Threshold values were computed on 12 different open source software projects. They discovered that ROC curves outperform the other investigated methods. Alves rankings were a close second.

The modeling techniques discussed by Boucher et al. can easily be adjusted to model the dataset discussed in this paper. My techniques would greatly benefit from automated threshold setting for the aggregator calculation. Aggregation can also be considered as a classification problem. Instead of software metrics I can use the various social media and popularity metrics that I have discovered in my studies.

#### Alves rankings

While Alves ranking was found to be second in terms of accuracy, it has a strong advantage above ROC curves. Alver ranking doesn't require historical fault data. Mapped to my problem, I would be able to automatically generate popularity threshold based on the popularity metrics in the current discussion without a historical records of previous discussions to serve as examples of classifications. This fits well with my methodology, which is intended to be applied as a stand-alone method. In the original paper Alves et al. [AYV10] set out to deliver a methodology that is:

1. The method should not be driven by expert opinion but by measurement data from a representative set of systems (data-driven);
2. The method should respect the statistical properties of the metric, such as metric scale and distribution and should be resilient against outliers in metric values and system size (robust);
3. The method should be repeatable, transparent and straightforward to carry out (pragmatic).

#### ROC curves

While ROC curves require historical data, they performed best out of the compared methodologies by Boucher et al. [BB18]. As a resolution to that applicability problem the researchers suggested using previous versions of the software. They used records of records of software faults as historical data to train the model and tested how well it works for estimating the likelihood of faults subsequent versions. They found the method provided promising performance for most software repositories that they considered.

A similar approach can be employed to thresholds required in the Canary methodology. While I recommend using Alves rankings for pilot studies, historical data that results from such studies can be used for more accurate estimates of thresholds using ROC curves.

### 4.3.4 Formal Syntax and Semantics

In this section, I describe the language formally. Table 4.2 defines the syntax of Canary.

The semantics of every expression in the language of Table 4.2 is given as an SQL query. Formally, for any such expression $x$ in Table 4.2, the function SQL($x$) gives the SQL query that $x$ maps to. Below, I define SQL inductively from the simplest expressions to the most complex ones. For the purposes of this thesis I give the definitions in pseudocode.

SQL(support(x)). Similarly, in order to find supporting arguments for an RE entity, I first traverse the tree, then reduce the score, and then join it with the RE entity itself (in this case represented as SQL(x)), using a common foreign key.

SQL(rebuttal(x)). Finding rebuttal arguments for an RE entity is analogous: I traverse the tree, then reduce the score, and join it with the RE entity itself (represented as SQL(x)), using a common foreign key.

Canary supports aggregators such as *discussed*, *popular*, *unpopular*, and *controversial* to allow a selection of objects of interest based on aggregate metrics.

SQL(discussed(x)). I define a requirement to be discussed if the sum of the number of replies (support and rebuttals) and upvotes and downvotes is greater than some threshold. I find the number of supports by using the SQL count function and grouping by the foreign key I will use for the join with the entity. I compute the number of rebuttals in an analogous way. Then I join with SQL(x). The resulting relation has both counts of support and rebuttal comments as attributes, and I apply the threshold condition $\alpha$ so I get only those $x$ with values above threshold. Note that the value of $\alpha$ will be configured by the analyst. I discuss possible data driven approaches to calculating thresholds in Section 4.3.3.

Listing 4.4: SQL(discussed(x))

```
SQL (x) as object
where (object.rebs + object.sups) > α
```

The sets of *popular*, *unpopular*, and *controversial* records are all subsets of *discussed*. They are all discussed entities, where either the positive sentiment dominates (popular), or the negative sentiment dominates (unpopular), or there is a balance between the two (controversial). I define positive sentiment as the sum of the number of support and upvotes and negative sentiment as the sum of the number of rebuttals and downvotes. In order to calculate them, I encapsulate SQL(discussed(x)) in a select statement and apply the appropriate selection filter to it.

SQL(popular(x)). Gives all discussed $x$ where the ratio of positive to negative sentiment is greater than $\beta$. Again, $\beta$ is configurable by the analyst. For this thesis, I set it to 1.15, as shown in Listing 4.5. I discuss possible data driven approaches to calculating thresholds in Section 4.3.3.

Listing 4.5: SQL(popular(x))

```
select * from(
  SQL (discussed(x))
) as object
where ((object.supsScore * object.sups) /
  (object.rebsScore * object.rebs) ) > 1.15
```

SQL(unpopular(x)). Gives all discussed $x$ where the ratio of positive to negative sentiment is less than $\theta$. Again, $\theta$ is configurable by the analyst. For this thesis, I set it to 0.85.

Table 4.2: Syntax of Canary

| | |
|---|---|
| <query> | : <expr> \| <arg-expr> |
| <expr> | : <req-expr> \| <sol-expr> |
| <req-expr> | : requirement \| requirement where <condition> \| <aggregator> ( <req-expr> ) |
| <sol-expr> | : solution ( <req-expr> ) \| solution ( <req-expr> ) where <condition> \| <aggregator> ( <sol-expr> ) |
| <arg-expr> | : <arg-entity> ( <expr> ) \| <arg-entity> ( <expr> ) where <condition> |
| <arg-entity> | : support \| rebuttal |
| <aggregator> | : popular \| unpopular \| controversial \| discussed |

SQL(requirement). This gives the SQL query to return all the comments expressing requirements. I then traverse their trees (comment-reply structure of the discussions) to reduce their propagated scores. In the example in Listing 4.1 posScore is the sum of the score of positive interaction (supports or derived objects), posInter is the count of how many positive objects of interest are in the tree. Calculating the negative side of the tree is done the same way.

Listing 4.1: SQL(requirement)

```
select comment.*, sum(posScore), count(posInter),
    sum(negScore), count(negInter) from (
  select comment-reply-tree in (
    select * from comment
    join requirement
    on comment.id = requirement.idcomment  ) )
```

SQL(x) where $\phi$. Gives the SQL query to return all $x$ that satisfy the condition $\phi$. In essence, $\phi$ acts as a selection filter, as shown in Listing 4.2.

Listing 4.2: SQL(x) where <condition>

```
select * from SQL (x) where  φ
```

SQL(solution(x)). Since solutions must be related to a requirement to make sense, I must link them to the requirements they satisfy. I accomplish this by joining the solutions table and SQL(requirement), in this case represented as SQL(x). Listing 4.3 is the SQL pseudocode for the definition.

Listing 4.3: SQL(solution(x))

```
select comment.*, sum(posScore), count(posInter),
    sum(negScore), count(negInter) from (
  select comment-reply-tree in (
    select * from comment
    join solution
    on comment.id = solution.idcomment  )
) as solution
join SQL (x) as requirement
  on solution.idparent = requirement.idcomment
```

SQL(controversial(x)). Gives all discussed $x$ where the ratio of positive to negative

sentiment lies between $\theta$ and $\beta$.

The appendix .2 provides a technical report that details the full implementation in MySQL the Canary query language.

### 4.3.5 Implementation of Canary Compiler

I implemented a compiler for Canary syntax in Java. I use the Eclipse XText (version 2.9) language definition and parsing library. The compiler takes queries written in Canary grammar and generates SQL queries following the definitions in Section 4.3.4 that can then be run on the aforementioned database. To accomplish this, the compiler essentially takes advantage of XText facilities. Given a Canary expression, XText creates a parse tree of a Canary expression based on the grammar and allows a recursive traversal of the tree, plugging in the SQL expressions that each node in the tree maps to.

#### Graph Databases

Relational databases are not the ideal solution to every problem. NoSQL data storage and querying solutions have recently emerged as a very strong candidate to remedy the shortcomings of relational databases in various aspects.

One major aspect that my implementation suffered in was the fact that SQL cannot handle recursive queries gracefully and efficiently. MySQL, my initial choice of storage and query language doesn't support native recursive queries. In order to go around that limitation without restarting the implementation of Canary I had to implement my own recursive algorithm in MySQL. That algorithm can be seen in Appendix [REF]. This limitation of SQL comes from the nature of how relationships are stored in relationship databases. Basically, SQL databases store a Foreign Key (FK) reference in the record of each entity that need to have be part of a relationship. Therefore, answering a recursive query would require multi-level joins between the related entities on the FK. As mentioned earlier, MySQL doesn't support foreign recursive queries in a native expression, but regardless all implementations of recursion in a relational database suffer from this limitation, including mine and any of the commercial versions of SQL. Recursion in a relational database is inefficient and the complexity of necessary joins can grow out of hand very quickly with even just a few nested entities in a large dataset.

This limitation of relational databases affects the implementation of Canary. The value of the Canary query language comes from being able to query entire discussions with powerful high-level queries. In order to achieve that it would need to recursively traverse all of the nested comment-reply nested entities in the discussions. Recursion is necessary since the discussions can be of arbitrary depth. In Chapter 3 I found that in the discussions I studied the depth went to as high as 10 and there was valuable information and interaction all the way down to the deepest parts of the discussions.

NoSQL databases have emerged as a solution to the shortcomings of relational databases. One such example are Graph databases. They are databases that use graph structures to query data directly leveraging nodes and edges. Graph databases are a data storage solution that is designed to capture the intricacies of connected data. It has native recursive queries and can execute them efficiently. It is for that purpose that I changed

my preferred back end of Canary from MySQL to Neo4j. Neo4j is a graph database management system. It often described as an ACID-compliant transactional database with native graph storage and processing.

I will not re-define the semantics of Canary from earlier in this chapter and in the appendix, as they are still valid, but for future implementations or work that builds on mine I recommend using Neo4j (or any other implementation of graph databases) storage and generating Cypher (the query language that Neo4j uses) code rather than MySQL.

## 4.4 Discussion

In this chapter, I have presented Canary; a tool-supported approach for querying requirements-related artifacts from user discussions. The centerpiece of the approach is a high-level query language in which requirement analysts can pose simple but useful queries to take advantage of the social features of online discussions. My query language has a translation into SQL, which means that queries can be executed against discussions stored in relational databases. I implemented a compiler and demonstrated the results of a few Canary queries on a database of real discussions. Analysts and developers may use Canary to inform their reasoning when compiling the list of formal requirements.

To obtain the metadata for storage in database, I obtained requirements and argumentation-related annotations from Mechanical Turk users. I demonstrated the efficacy of my approach for annotations by providing a detailed empirical analysis of the accuracy of annotations. Although the results are promising, I observe a high variance in results in Table 3.4. This suggests that the accuracy may vary for discussions.

Canary annotations are less complex than some requirements models in the literature. Some examples of what I did not consider include conflict, priority, positive and negative contributions, and assumptions [BPG+04, vL09] and argumentation [Tou58a]. Richer requirements models would require a more complex annotation schema. It is conceivable that there would be a proliferation of conceptual models with various levels of technical sophistication and associated query languages and tools.

An important future direction is to augment Canary with automated annotation techniques based on NLP. Argumentation mining [BHDM07] has recently been applied to social media [PC14, BŠ14a] and Canary may be able to exploit argumentation mining toward automating annotations. Another interesting application of NLP would be to use NLP as the underlying query processing engine. Such an engine might, for example, detect discussions about similar requirements in two or more distinct discussions and merge the interaction from both discussion to calculate the output of the query.

The overall methodology is currently labour intensive. I had to extract data from online forums, and reproduce it in a format suitable for annotation by MTurk users, and then load the annotated data into the database. I created several custom tools to help me with the tasks, e.g., for extracting data from Reddit using its published API and scripts to load annotated data into a database. A future direction will be to build an automated tool chain.

# 5

# Crowd-Informed Goal Models

## 5.1 Methodology Overview

In this chapter I present a design science approach to answering **RQ4**. I demonstrate how a high-level requirements language such as Canary may be systematically applied toward requirement engineering. Goal modeling is one of the strong emerging artifact candidates of requirements engineering. Strong emerging candidate in this context is defined by its recognition in literature as likely to be adopted by practitioners. In this chapter I augment goal models using the previously discussed structured query language Canary. The augmentation is my answer to the question of what is a method for goal-driven prioritization that uses evidence from online discussions about software products. The Canary language was built to leverage the underlying argumentative structure of online discussions and numerical metrics generated through interaction between users in online platforms, such as votes. In this part of my research my main motivation comes from an aim to allow my previous contributions to be easily and seamlessly integrated into existing RE practices. A reasonable way of doing that is to enrich existing artifacts, such as goal models.

I apply crowdsourcing to support a rich, dynamic, and user-driven understanding of an application's requirements from information in online discussion forums. Further, I employ argumentation [RD92, HLMN08, MKTS15a] to structure discussions for obtaining high-quality requirements.

The payoff of this chapter consists in an exploration into a systematic approach that can be used to feed valuable information found in social media into the development process.

- I 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.
- I 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. I use subjective logic [Jøs01] on this model to obtain a prioritization of the goals.

Figure 5.1 shows an overview of my 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. I provide details on this part with the necessary background on Canary in Section 5.2. 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 5.3 provides details on this part with the necessary background on subjective logic.

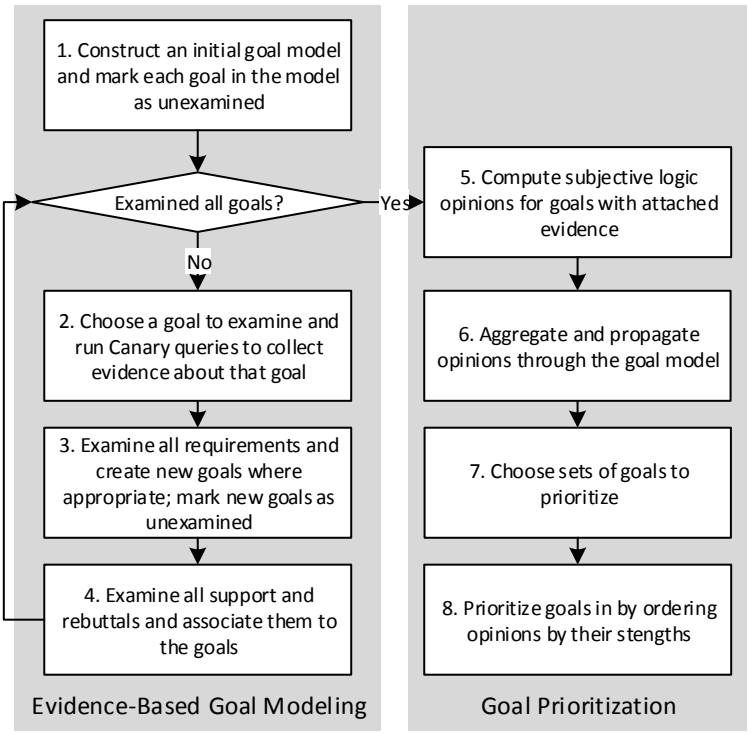



Figure 5.1: Methodology overview

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

## 5.2 Evidence-Driven Goal Models

### 5.2.1 Canary Query Language

Canary, presented in Chapter 4, 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 has first-class abstractions that capture all of the requirements and argumentative entities discussed above. Developers can write powerful queries that leverage entire discussions. The methodology proposed in this thesis exploits Canary for information extraction.

### 5.2.2 Goal-oriented Requirements Language (GRL)

I employ Goal-oriented Requirements Language (GRL) [goa] since it includes all artifacts I investigate. However, my 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 [GMNS02]. 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. Such relationships can be labeled with "+" and "-". The GRL notation I require for this thesis is shown in Figure 5.2.

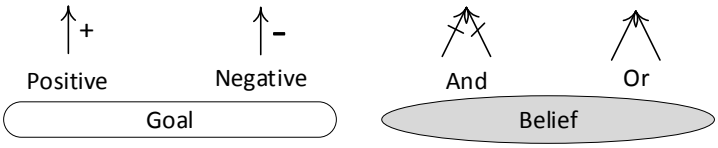



Figure 5.2: Notations used for relations in GRL

During the construction of the goal model it is reasonable to start with the overall, highest level goals. Then, I 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

I illustrate my methodology with a *maps* application; specifically, modeling its *navigation* feature.

### 5.2.3 Modeling Steps

**1. Construct an initial goal model and mark each goal in the model as unexamined.** Figure 5.3 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.

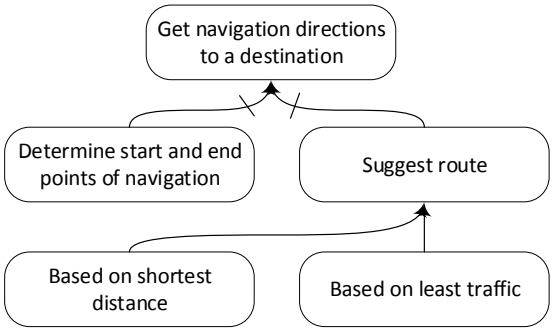



Figure 5.3: An initial goal model for the maps navigation feature

My method relies on examining each goal separately until all goals have been examined. The order of examination is not of crucial importance. In the case of the model in Figure 5.3, 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, I specify a set of queries that can be run for each goal.

Figures 5.4 and 5.5 show sample pieces of evidence (annotated discussions) in the Canary database that are closely related to the model in Figure 5.3. 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 I reverse the process somewhat. The developer will query for requirements based on intuition and textual similarity to the goal specification. For my 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*.

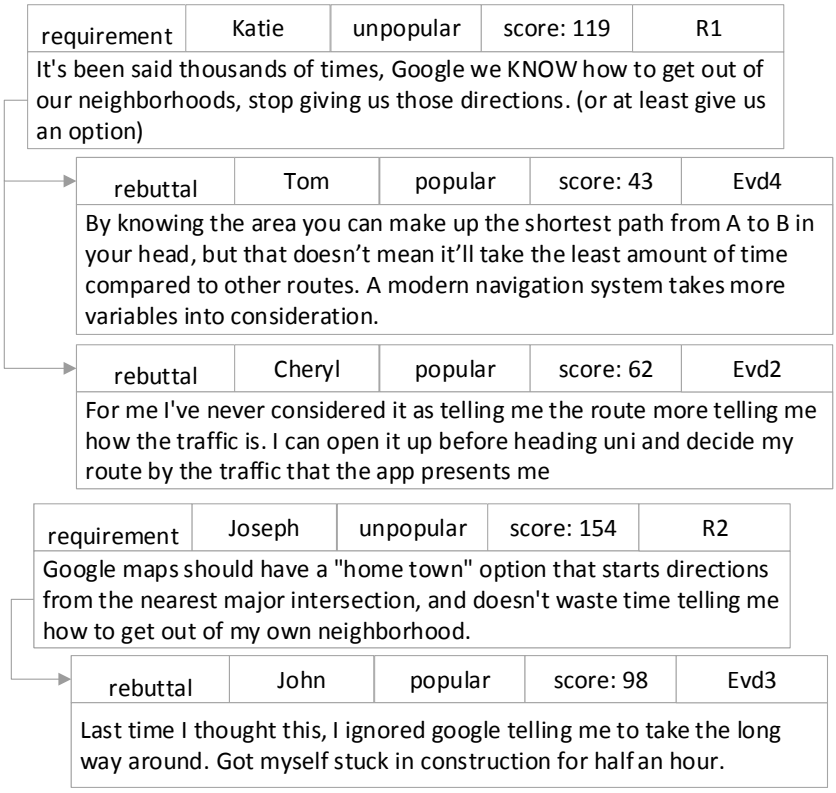



Figure 5.4: Example of evidence related to keyword "directions"

An example of a query that would return a set containing such evidence is in Figure 5.6, where the developers asks for requirements containing *directions*. 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, I stick to basic query types.

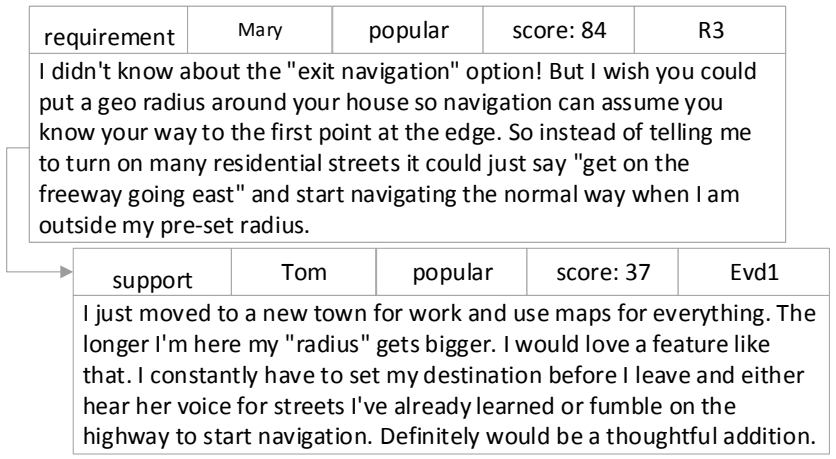



Figure 5.5: Example of evidence related to the keyword "navigation"

```
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 |

Figure 5.6: 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 I 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, I 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 5.6, the developer may find that neither of these requirements can be directly mapped to any of the goals in existing model shown in Figure 5.3. 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 5.7.

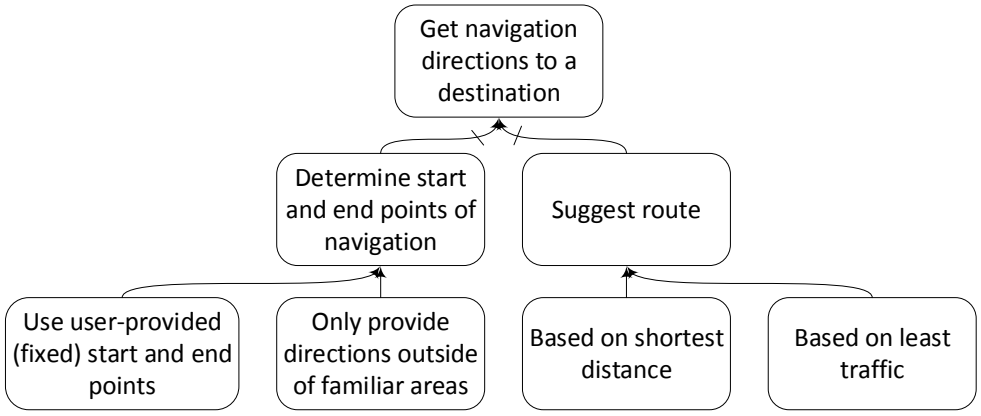



Figure 5.7: Refactoring the goal model and adding new goals

**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, I can go deeper into the discussion by querying the nested interactions. This may gives 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 5.8 shows a rebuttal query to identify conflicting evidence for the goal under consideration. Similarly, as shown in Figure 5.5, the discussions can also have supporting arguments. Figure 5.9 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 |

Figure 5.8: An example query to find rebuttals of requirements related to keyword "directions" and the output

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

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

Figure 5.9: 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 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. My 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 5.10.

For example, I can focus on each piece of evidence in the table in Figure 5.8, 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.

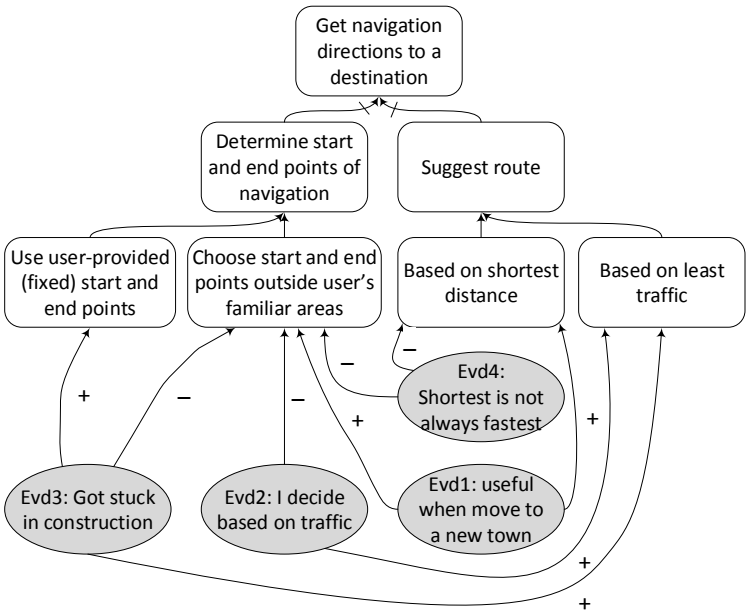



Figure 5.10: Adding argumentative evidence that relates to more than one goal

## 5.3 Goal Prioritization

Once I attach crowdsourced evidence to a goal model, I exploit that evidence to prioritize goals. My 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.

I exploit *subjective logic* [Jøs01], 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. My 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 my scenario very well in that I am 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.

### 5.3.1 Subjective Logic

To be self-contained, I provide a brief summary of subjective logic primitives pertinent to this thesis. Additional details can be found in the original works [Jøs01, JHP06].

#### 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:

$$b = \frac{r}{r+s+2}; \quad d = \frac{s}{r+s+2}; \quad u = \frac{2}{r+s+2} \tag{5.1}$$

#### Mapping to Probability

An opinion $\omega$ can be mapped to a probability expectation value, $E(\omega)$ (see Figure 8 in [JHP06]). I require this mapping in a later stage for ordering opinions.

$$E(\omega) = b + au. \tag{5.2}$$

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 [JHP06]).

#### Logical Operators

The subjective logic framework describes the following operators (among others) to facilitate evidence aggregation and propagation.

**Negation**:

$$b_{\neg x} = d_x; \quad d_{\neg x} = b_x; \quad u_{\neg x} = u_x; \quad a_{\neg x} = 1 - a_x. \tag{5.3}$$

**Conjunction**:

$$\begin{aligned} b_{x\wedge y} &= b_x b_y; \quad d_{x\wedge y} = d_x + d_y - d_x d_y; \\ u_{x\wedge y} &= b_x u_y + u_x b_y + u_x u_y; \\ a_{x\wedge y} &= \frac{b_x u_y a_y + u_x a_x b_y + u_x a_x u_y a_y}{b_x u_y + u_x b_y + u_x u_y}. \end{aligned} \tag{5.4}$$

**Disjunction**:

$$\begin{aligned} b_{x\vee y} &= b_x + b_y - b_x b_y; \quad d_{x\vee y} = d_x d_y; \\ u_{x\vee y} &= d_x u_y + u_x d_y + u_x u_y; \\ a_{x\vee y} &= \frac{u_x a_x + u_y a_y - b_x u_y a_y - u_x a_x b_y - u_x a_x u_y a_y}{u_x + u_y - b_x u_y - u_x b_y - u_x u_y}. \end{aligned} \tag{5.5}$$

#### 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, I assume that the ordering is arbitrary.

### 5.3.2 Prioritization Method

Let us resume from Figure 5.10, where I had a crowd-informed goal model with associated evidence. In order to reason about priorities in this model based on subjective logic, I proceed with the next four steps of my method.

**1. Compute opinions for goals with attached evidence.** First, for each goal with at least one piece of associated evidence, I compute an opinion ($\omega$) for the goal. To do so, first I compute the amount of supporting (positive) evidence, $r$, and rebutting (negative) evidence, $s$. Further, to compute $r$ and $s$, I employ the scores associated with each piece of evidence. For example, consider the scores in Figure 5.11. Note that these scores are derived from scores in Figures 5.8 and 5.9, 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 5.11, for the goal "Choose start and end points outside user's familiar areas," $r = 4$ and $s = 20$. Given $r$ and $s$, I can employ the evidence mapping functions in Equation 5.1 to compute the $b$, $d$, and $u$ parameters of the opinion (I assume the apriori base rate $a = 0.5$). Figure 5.11 shows the computed opinions.

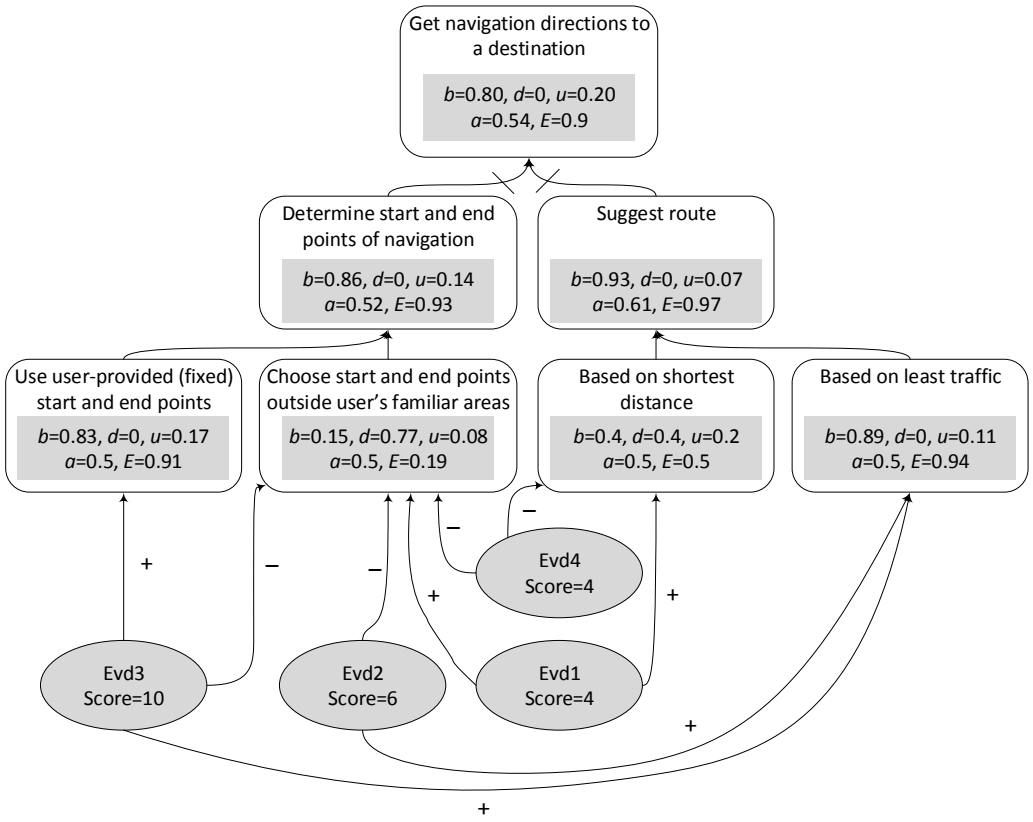



Figure 5.11: Example of a goal model with opinions

**2. Aggregate and propagate opinions through the goal model.** In the previous step, I 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, I employ the aggregation operators described in Equations 5.3–5.5 to aggregate evidence. Figure 5.11 shows examples of opinion aggregation for two OR and one AND decompositions.

**3. 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 5.11, I see two opportunities for prioritization—one for each OR decomposition.

**4. Prioritize goals by ordering opinions.** Once I compute opinions for goals in a model, goals can be prioritized based on the rules described Section 5.3.1. In Figure 5.11, for the OR decomposition on the right, I can prioritize "Based on least traffic" over "Based on shortest distance." Similarly, for the OR decomposition on the left, I 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.

## 5.4 Discussion

I described crowd-informed goal models that yield prioritization. The model construction is guided by an incremental yet novel methodology. I 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. I 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.

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

Without a proper systematic evaluation it is difficult to tell what the direct benefits of this methodology can be but the following is a reasonable hypothesis. The methodology can be useful to software development practitioners, especially such that are interested in developing market-driven software. To re-iterate, market-driven development is such that it develops software that is to be released in an open market and used by a non-discrete crowd of users as opposed to bespoke software, which is developed with specific users in mind to be used in a specific setting. Validating goals and requirements can be a challenging process when there is no explicit end user. The intended users of this methodology can either have their product already on the market and apply the methodology to discussions about their specific application. This would enable them to gain a high-level overview of the way their crowd of users perceives their product. It would give them a numerical representation of what works in a satisfactory manner and what doesn't. Information gained from this methodology can be used to plan future releases and prioritize requirements for such. Alternatively, a practitioner can use it in a more generic way by applying it to a set of discussions about an application they have yet to begin development on or on discussions of their competitor applications. This would allow them to identify weak points for existing solutions by eliciting goals that are not adequately or at all satisfied according to the available evidence. They can also easily identify strong points of competitors that they need to replicate in order to stay afloat on the market.

A software development practitioner that decides to apply this methodology may have one of two entry points, first they may already have a definition of a goal model for their product. In this case they would wish to enrich and validate it using evidence from the crowd. Enrichment of existing artifacts with crowd evidence would improve the validity of the model and may elicit new goals that inspire future development directions inspired by the trends in the market. Enriching and updating goal models using crowdsourced data driven approach would allow developers to react quickly to a changing market. Second they may not have a goal model at all and wish to produce one from scratch using my methodology. Goal models have a number of benefits and providing practitioners with a

semi-automated tool to generate such would improve the documentation of a software project and improve communication within the team.

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. My 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 available in the database of evidence.

Finally, having a goal with no evidence associated to it can pose a challenge to my 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 to investigate the evidence further. Incomplete evidence can cause problems in other aspects of the methodology, such as model creation. Measuring the "completeness" of a goal model is non-trivial, but I 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.

# 6

# Conclusion

This chapter aims to summarize, examine, discuss and thus conclude my work. It is divided into sections, the first four of which explore each of my research questions. Next, I explore the implications and future directions of my work.

## 6.1 The Structure of Discussion

The fundamental constituent of this thesis, the principle on the basis of which all other work has been built is laid down for examination in **RQ1**. The research question aims to examine whether online discussions about software applications have a naturally occurring underlying structure that can be captured using abstractions borrowed from requirements engineering and argumentation. In order to explore this research question I undertook a series of experiments.

In the first stage of this thesis I explored the structure, shape, and content of online discussions using an application of the grounded theory research methodology, as presented in Chapter 3. I carried out a case study about Google Maps, detailed in Chapter 3.1.1, on Reddit.com, which is a Web forum, briefly presented in Chapter 3.1.2. I set out to highlight and bring forth important artifacts relevant to requirements engineering in the interactions between users on forums. In Chapter 3.3, I also critically evaluate the effectiveness of goal modeling techniques to capture the information contained within feedback in social media with the purpose of enhancing requirements modeling with notions that capture user interactions.

In order to achieve that I conducted an experiment in grounded theory, in which manually explored Reddit.com in search for discussions between users regarding Google Maps. The result of my application of the grounded theory method, presented in detail in Chapter 3.2.2, were several key intuitions regarding the information of value to requirements engineering available in social media. Such intuitions include *requirements*, *observations made by end-users* (explicit statements of tacit knowledge), *lack of awareness of existing solutions*, *expressions of sentiment towards previously discovered artifacts* (users tend to express their agreement that a requirement is necessary or that a solution is not good

enough), and *community support* (well defined, pertinent artifacts are recognized by the community using the collaborative voting system). My intuitions of content in social media are driven by example. All statements are defended using a real world example taken from an organic discussion that was found on the website Reddit.com. This can be seen in Chapter 3.2, where I discuss the examples given in Figure 3.2 and Figure 3.3.

In Chapter 3.3, I extracted two discussions from the forum and built a goal model of Google Maps using the domain knowledge gained solely from those discussions. Comparing the information captured in the resulting model to the amount of information available in the source, I concluded that requirements modeling can benefit from enhancements that capture user interactions by taking advantage of several key elements of social media that involve quantitative metrics. Examples of that are *community support and sentiment*, *user reputation*, and *controversy of entities*. In addition, I also discovered that the interaction between users on the forum is structured as argumentation and that combining multiple sources of domain knowledge in the same model enriches my understanding of the domain.

The results gained from such experiments is of qualitative nature. This is a known weakness of case studies, grounded theory and design science. Wohlin et al. [WRH+12] address this critique by stating that knowledge is not only statistical significance. This is an important point and one must bear it in mind whilst reading the findings of my research. Wohlin et al. present the case study methodology as well suited for many kinds of software engineering research, as the objects of study are contemporary phenomena, which are hard to study in isolation. Methodologies of similar nature to case studies provide deeper understanding of the phenomena under study in its real context, such as online discussions about software products.

In Chapter 3.6, I undertook an experiment that applied the intuitions gained from this application of grounded theory to practice. In this experiment I aimed to provide empirical evidence that using annotations inspired by my previous findings can serve as a good annotation schema for discussions. The annotation labels taken into account were *requirement*, *solution*, *support* and *rebuttal*. As study units, I selected five online discussions from two social forums, each involving user discussions about Google Maps and related software applications. To determine the validity of my intuition, two software engineers, acting as expert annotators, annotated each discussion in three rounds. It is worth noting that one of the software engineers was myself and the second was another, more experienced academic colleague. In the first round, the two experts annotated the discussions independently without seeing each other‘s annotations. In the second round, they saw each other‘s annotations and updated their annotations, independently. In the third round, the experts discussed their annotations, resolved differences, and settled on one set of annotations as the *ground truth*. As result I came up with a dataset that allowed me to present in this thesis empirical statistics on the contents of online discussions in social media and to validate my efforts in the application of grounded theory.

It is worth noting that while many precautions were taken to prevent any cognitive biases influencing the results of the above mentioned study the experiment was carried out by two human workers who are a subject to cognitive bias. On top of that they were both involved in the design, implementation and write up of the paper that the work was published in.

The results of this experiment, presented in Chapter 3.6.3, show that naturally occurring in social media discussions contain a significant portion of information that can be labeled as interesting and valuable to software engineering practitioners. In this context, as interesting and valuable I mean labels such as requirement, solution, support and rebuttal. Plots that summarize my empirical observations uncovered in my experiments can be seen in Figures 3.10 — 3.16. I also explore the distribution of interesting labels in different depths of the discussions. It is worth noting that the discussions I picked as a case study were not random, they were picked because they showed signs that they contain information that could be of use to requirements engineering. The selection was performed using simple tools, such as the search bars available on Reddit.com and any other web forum and can easily be made systematic and rigorous. It is also worth noting that for all discussions I examined a majority of comments were still labeled as "noise". This is to be expected, since online discussions are driven by lay people participating in an ad-hoc way for no obvious incentive. The content labeled as noise is not of any use to the methodologies that I've devised or to any RE practitioners that may be interested. Regardless, the remaining content, labeled with as one of the other more meaningful annotations makes up for nearly half of the total content. With that, I was able to build a number of tools that can aid RE and many more uses can be found. Noise doesn't contribute to the value of discussions, but it also doesn't take away from it. Noise is an immutable part of the information found in such crowd-based sources. I used the dataset that was produced to present a deep investigation into the structure of online discussions. I showed using graphs and plots the distribution of various labeled artifacts in relation to their level in the hierarchical comment-reply structure of online discussions. My investigation showed that the valuable information tends to be close to the top of the hierarchical structure of discussions, indicating that people in casual environments tend to steer off topic with time, which also makes intuitive sense.

I also present graphical representations of my data that justify my choice of social media outlet in Figures 3.5 — 3.9. I plot the discussions I've analyzed as graphs in order to show how those discussions are structured and make the argument that social media outlets enable for much more interesting discussions.

## 6.2 Scaling the Solution

My next research interest lead me to addressing **RQ2**, a question seeking to find the accurate and efficient are crowd sourcing techniques at producing annotations for online discussions about software applications. In order to answer this question I conducted an experiment employing Amazon Mechanical Turk (MTurk) users. As study units, I selected five online discussions from two social forums, each involving user discussions about the Google Maps and related software applications. Table 3.1 summarizes these discussions. AMT is a crowdsourcing platform where lay users can be commissioned to perform HITs, human intelligence tasks. Splitting the discussions yielded 38 microtasks. I sought to acquire annotations from two users for each microtask so as to get a reliable estimate. Accordingly, I launched 76 HITs (human intensive tasks) on Amazon MTurk and collected annotations from 44 unique MTurk users.

In the main task, first, I asked users to read about the core concepts of requirements, solutions, supports and rebuttals from a document I provided with the aid of examples. I

instructed users to select any comment or a part of a comment in the provided discussion and annotate it as one of the four entities: *requirement*, *solution*, *support*, or *rebuttal*.

The results of this experiment are presented in detail in Chapter 3.7.3. **RQ2** aims to find the effectiveness of the crowd at annotating discussions. Effective, in this context refers to accuracy and efficiency.

The **accuracy** of the annotations is presented in Chapter 3.7.3. The accuracy of the annotations was measured against a list of annotations gathered from expert annotators, as explained in Chapter 3.6. Table 3.3 shows a table comparing the expert- and user-labels, aggregating counts across all five discussions. Table 3.4 shows the mean and variance of the per-discussion precision, recall, and $F_1$ scores. Overall, my experiment revealed that a crowd of lay users can produce annotations over online discussions with high accuracy.

The **efficiency** of the annotators is presented in Chapter 3.7.3. Figure 3.17 (top-most box plot) shows the distribution of the durations reported by the users. The mean amount of time participants spent on the main task is about 35 minutes. However, the variance in time spent is high.

I had a few returning users in my dataset ($n = 10$). The bottom two box plots in Figure 3.17 compare the durations reported by these users for the first task and the second tasks. I find that users take significantly less time the second time ($p = 0.02$; measured via Wilcoxon's ranksum test, excluding outliers).

## 6.3 Methodical Queries

In the second stage of my research, I developed and presented Canary; a tool-supported approach for querying requirements-related artifacts from user discussions. The centerpiece of the approach is a high-level query language in which requirement analysts can pose simple but useful queries to take advantage of the social features of online discussions. My query language has a translation into SQL, which means that queries can be executed against discussions stored in relational databases. I implemented a compiler and demonstrated the results of a few Canary queries on a database of real discussions. Analysts and developers may use Canary to inform their reasoning when compiling the list of formal requirements.

Canary annotations are simpler than some requirements models in the literature. In particular, I did not consider conflict, priority, positive and negative contributions, and assumptions [BPG+04, vL09] and argumentation [Tou58a]. Including these concepts in Canary would require considering their meaning in the context of user discussions. For instance, prioritization could be achieved with a suitable notion of popularity. The trade-off between richer requirements models is more complex annotation. It is conceivable that there would be a proliferation of conceptual models with various levels of technical sophistication and associated query languages and tools. My choice was additionally validated by my previous work in grounded theory that formed the foundation of my preferred set of abstractions. My work on the Canary methodology and query language came in response to **RQ3**, my research question which aims to examine whether it would be valuable to create a tool, such as a query language for the purpose of automating and

assisting the extraction of valuable information from potentially large online discussions. When presented the Canary methodology was illustrated with real world examples extracted directly from social media and presented to the reader without augmentation. The main source of examples is presented in Figure 4.9 where I show an example discussion that I use to illustrate the framework. Following that in Chapter 4.3.2, I start a running example of how to apply the Canary methodology on that example discussion so as to extract pertinent information from it. This example-driven approach was done intentionally with the purpose of demonstrating to the reader the value of the methodology and language as they were unfolding.

## 6.4 Crowd-Informed Models

In the third, last stage of my research, I described crowd-informed goal models that yield prioritization. The model construction is guided by an incremental yet novel methodology. I 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. I 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.

This part of my research addresses **RQ4**, my research question which aims to explore whether Canary queries can be used to augment and enrich existing Requirements Engineering artifacts, such as goal models. I demonstrate the value of the methodology via a running example, but lack a formal evaluation. The presentation of the methodology is intertwined with the illustration of its application to real world projects using examples extracted from existing discussions from social media. This example-driven approach is presented in Chapter 5.2.2. The methodology augments goal modeling, which is an existing requirements engineering artifact that has a high potential of being widely adopted by industrial practitioners. I plan to conduct a full-scale user study to evaluate the practical value of the approach. I intend to evaluate whether my method (1) yields richer goal models and (2) simplifies the modeling process compared to other goal modeling approaches.

## 6.5 Implications

The primary target beneficiaries of my research are requirement engineering practitioners that work in market-driven software development. Due to the inherent difficulties in end user contact in such projects, especially in early stages of the development project, they can benefit most from my findings, methodologies and tools. The group of end users for such projects can be referred to as a crowd to begin with, as a crowd is defined in Chapter 2.2.1. It is worth noting that before decisive conclusions can be made about the benefit of my work for practitioners a user study is mandatory. The theory on design science, detailed in Chapter 1.2.3, is adamant about that. In my example-driven approach, I've demonstrated how my research can provide value for practitioners. My tools can reduce large unstructured discussions to volumes and formats easily comprehensible by humans. Thus my work can open entirely new channels of communication between end users and

practitioners where the voice of the crowd does not go unheard and the cacophony of many voices speaking at once is handled gracefully.

Techniques aimed at the inclusion of user feedback of various forms are more than likely to play an increasingly larger role in modern software development. This is only becoming more viable with the recent rise of various machine learning, artificial intelligence and other data-driven approaches. My work is based on natural language, a type of data which is difficult to process without the use of machine learning. Despite that the a large part of the payoff and contributions of my work are based on making use of the data after it has been organized and structured, which in this case is largely applying annotations to the underlying natural language data. Advancements and interest in data-driven computation make it reasonable to assume that acquiring such annotations will become increasingly trivial in the future. It is reasonable to assume that my work will inform and inspire the appropriation of machine learning techniques on natural language discussions for the purposes of requirements engineering. While there are many directions that my research can be taken in, as explained next in Chapter 6.6, my techniques and tools can also inspire completely new, innovative ways of leveraging labeled data. Additionally, my exploration into the abstractions that can be used to structure discussions can also be used to inform and inspire more sophisticated forms of annotation.

## 6.6 Future Directions

The insights of my initial experiments can be used to motivate research in designing a requirements model. The notions supported by the model can capture user interaction as context to traditional requirements artifacts, such as goals or requirements. Social media is a highly interactive environment, where users are encouraged to evaluate, rate, and vote on each other‘s contributions to the community. Such a model would require abstractions that can be informed by my findings. Many different sources and combinations of abstraction can be experimented with for such models.

In my work I found argumentation to emerge as a candidate source of abstractions that can capture the qualitative requirements-related information in social media, and can be expanded to accommodate the quantitative metrics mentioned above. In my work I borrowed two of the most basic argumentation-inspired abstractions — support and rebuttal. The simplicity of my choice of abstractions is due to considerations of the complexity involved in annotating discussions using more sophisticated models. Further research is needed on the applicability of argumentation towards understanding user feedback in social media, possibly with more complex models, or with different choice of fundamental abstractions. I provide an introduction of argumentation in Chapter 2.2.2, making the abundant choice of abstractions obvious.

The interactivity of social media makes the content within it dynamic. As a result, using it as a source of information can have some interesting implications on the value of the information I extract. New content emerges in social media at a fast paste, so how long after data is created it is still relevant? When is it most relevant? How do the dynamics of social media affect the design of a requirements model intended to capture interactions between the users of that media?

Analyzing the vast quantity of information in social media manually proved to be challenging and error-prone. An interesting area for future research is tool support for assistance and automation of the analysis of feedback in social media. Support can be provided for the finding, filtering, and extraction of information, as well as reducing the amount of manual labor needed for its analysis by applying natural language processing or crowdsourcing techniques.

I empirically explore one possible direction in the face of Amazon Mechanical Turk. While promising in results, it's not without its methodological weaknesses that can in themselves be a future direction of exploration. The overall methodology is currently labour intensive. I had to extract data from online forums, and reproduce it in a format suitable for annotation by MTurk users, and then load the annotated data into the database. I created several custom tools to help me with the tasks, e.g., for extracting data from Reddit using its published API and scripts to load annotated data into a database. A future direction will be to build an automated tool chain.

An important future direction is to augment Canary with automated annotation techniques based on NLP. Argumentation mining [BHDM07] has recently been applied to social media [PC14, BŠ14a] and Canary may be able to exploit argumentation mining toward automating annotations. Another interesting application of NLP would be to use NLP as the underlying query processing engine. Such an engine might, for example, detect discussions about similar requirements in two or more distinct discussions and merge the interaction from both discussion to calculate the output of the query.

Automation can also benefit other parts of my research, such as the crowd-informed goal models and the Canary query language. Approaches similar to Robeer et al. [RLvdW$^+$16] can be used to generate an initial goal model from existing requirements artifacts in the first step of my method. This would ease the adoption of my methodology. Further, I currently extract evidence via Canary queries and intuitive textual similarity. A better approach would be to study the applicability of NLP techniques based on semantic similarity for gathering all relevant evidence from the available sources. Techniques such as fuzzy matching have come a long way and can be adopted to improve Canary as well.

Relying on the developer to write queries for the extraction of evidence to inform goal models has other risks associated to it, such as confirmation and other types of cognitive biases. My 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 available in the database of evidence.

Finally, having a goal with no evidence associated to it can pose a challenge to informing goal models with crowd evidence. 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 to investigate the evidence further. Incomplete evidence can cause problems in other aspects of the methodology, such as model creation. Measuring the "completeness" of a goal model is non-trivial, but I 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 seek

out additional external sources to add to the database and attach to the model.

# Bibliography


[ACD+08] Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 international conference on web search and data mining*, pages 183–194. ACM, 2008. [Cited on page 2]

[ADG10] Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requirements Engineering*, 15(4):439–458, 2010. [Cited on pages 17 and 26]

[AG06] Vincenzo Ambriola and Vincenzo Gervasi. On the systematic analysis of natural language requirements with CIRCE. *Automated Software Engineering*, 13(1):107–167, 2006. [Cited on page 14]

[AGM11] Yudistira Asnar, Paolo Giorgini, and John Mylopoulos. Goal-Driven Risk Assessment in Requirements Engineering. *Requirements Engineering*, 16(2):101–116, 2011. [Cited on page 16]

[All03] George Allan. A critique of using grounded theory as a research method. *Electronic journal of business research methods*, 2(1):1–10, 2003. [Cited on page 5]

[ASIM14] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. A Systematic Literature Review of Software Requirements Prioritization Research. *Information and software technology*, 56(6):568–585, 2014. [Cited on page 17]

[AYV10] Tiago L Alves, Christiaan Ypma, and Joost Visser. Deriving metric thresholds from benchmark data. In *2010 IEEE International Conference on Software Maintenance*, pages 1–10. IEEE, 2010. [Cited on page 52]

[BB18] Alexandre Boucher and Mourad Badri. Software metrics thresholds calculation techniques to predict fault-proneness: An empirical comparison. *Information and Software Technology*, 96:38–67, 2018. [Cited on pages 51 and 52]

[BHDM07] Erik Boiy, Pieter Hens, Koen Deschacht, and Marie-Francine Moens. Automatic sentiment analysis in on-line text. In *ELPUB*, pages 349–360, 2007. [Cited on pages 56 and 76]

[BL11] Dejana Bajic and Kelly Lyons. Leveraging social media to gather user feedback for software development. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering*, pages 1–6. ACM, 2011. [Cited on pages 2 and 13]

[BPG+04] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. [Cited on pages 28, 56, and 73]

[BŠ14a] Filip Boltužić and Jan Šnajder. Back up your stance: Recognizing arguments in online discussions. In *Proceedings of the First Workshop on Argumentation Mining*, pages 49–58, 2014. [Cited on pages 56 and 76]

[BS14b] Travis D. Breaux and Florian Schaub. Scaling requirements extraction to the crowd: Experiments with privacy policies. In *Proceedings of the IEEE 22nd International Requirements Engineering Conference*, pages 163–172, Karlskrona, 2014. [Cited on page 13]

[CKI88] Bill Curtis, Herb Krasner, and Neil Iscoe. A field study of the software design process for large systems. *Communications of the ACM*, 31(11):1268–1287, 1988. [Cited on page 2]

[CS11] Amit K Chopra and Munindar P Singh. Colaba: Collaborative design of cross-organizational processes. In *2011 Workshop on Requirements Engineering for Systems, Services and Systems-of-Systems (RESS)*, pages 36–43. IEEE, 2011. [Cited on page 26]

[CSPG17] Adelina Ciurumelea, Andreas Schaufelbühl, Sebastiano Panichella, and Harald C Gall. Analyzing reviews and code of mobile apps for better release planning. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, pages 91–102. IEEE, 2017. [Cited on page 14]

[CTIB15] Justin Cheng, Jaime Teevan, Shamsi T. Iqbal, and Michael S. Bernstein. Break it down: A comparison of macro-and microtasks. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 4061–4064, Seoul, 2015. ACM. [Cited on page 35]

[CVL12] Antoine Cailliau and Axel Van Lamsweerde. A Probabilistic Framework for Goal-Oriented Risk Analysis. In *20th IEEE International Requirements Engineering Conference*, pages 201–210. IEEE, 2012. [Cited on page 15]

[Dav03] Alan M Davis. The Art of Requirements Triage. *Computer*, 36(3):42–49, 2003. [Cited on pages 1 and 17]

[DVLF93] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-Directed Requirements Acquisition. *Science of computer programming*, 20(1-2):3–50, 1993. [Cited on page 1]

[EFM18] Yehia Elrakaiby, Alessio Ferrari, and John Mylopoulos. Care: A refinement calculus for requirements engineering based on argumentation semantics. In *26th International Requirements Engineering Conference (RE)*, pages 364–369. IEEE, 2018. [Cited on page 16]

[EJ12] Christof Ebert and Michael Jastram. Reqif: Seamless requirements interchange format between business partners. *IEEE software*, 29(5):82–87, 2012. [Cited on page 15]

[EY10] Hesam Chiniforooshan Esfahani and Eric Yu. A repository of agile method fragments. In *International Conference on Software Process*, pages 163–174. Springer, 2010. [Cited on page 15]

[EYC10] Hesam Chiniforooshan Esfahani, Eric Yu, and Jordi Cabot. Situational Evaluation of Method Fragments: An Evidence-Based Goal-Oriented Approach. In *International Conference on Advanced Information Systems Engineering*, pages 424–438. Springer, 2010. [Cited on page 15]

[FDR+12] João Fernandes, Diogo Duarte, Claudia Ribeiro, Carla Farinha, João Madeiras Pereira, and Miguel Mira da Silva. iThink: A game-based approach towards improving collaboration and participation in requirement elicitation. *Procedia Computer Science*, 15:66–77, 2012. [Cited on page 18]

[FSM+18] 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. [Cited on page 13]

[GDA15] Eduard C Groen, Joerg Doerr, and Sebastian Adam. Towards crowd-based requirements engineering a research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 247–253. Springer, 2015. [Cited on pages 10 and 12]

[Ger13] Ricardo Gacitua Mark Rouncefield Peter Sawyer Leonid Kof L. Ma P. Piwek et al. Gervasi, Vincenzo. Unpacking tacit knowledge for requirements engineering. In *Managing Requirements Knowledge*, pages 23–47. Springer, 2013. [Cited on page 23]

[GK18] Eduard Groen and Matthias Koch. Crowd-based requirements engineering (whitepaper). 11 2018. [Cited on page 10]

[GKH+17] Eduard C Groen, Sylwia Kopczyńska, Marc P Hauer, Tobias D Krafft, and Joerg Doerr. Users—The Hidden Software Product Quality Experts?: A Study on How App Users Report Quality Aspects in Online Reviews. In *IEEE 25th International Requirements Engineering Conference*, pages 80–89. IEEE, 2017. [Cited on page 13]

[GMNS02] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with Goal Models. In *International Conference on Conceptual Modeling*, pages 167–181. Springer, 2002. [Cited on page 59]

[goa] Goal-Oriented Requirements Engineering. `https://www.cs.toronto.edu/km/GRL/`. Accessed: 2018-06-13. [Cited on page 59]

[Gre80] Günther Grewendorf. Argumentation in der sprachwissenschaft. *Zeitschrift für Literaturwissenschaft und Linguistik*, 10(38):129, 1980. [Cited on page 11]

[Gro17] Object Management Group. Unified modeling language UML, version 2.5.1. `https://www.omg.org/spec/UML/2.5.1`, December 2017. [Cited on page 1]

[GRW12] Phil Greenwood, Awais Rashid, and James Walkerdine. Udesignit: Towards social media for community-driven design. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 1321–1324. IEEE, 2012. [Cited on page 13]

[GSA+17] 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*, 34(2):44–52, 2017. [Cited on pages 1, 2, 10, 11, and 12]

[GZ14] Vincenzo Gervasi and Didar Zowghi. Supporting traceability through affinity mining. In *Proceedings of 22nd International Requirements Engineering Conference*, pages 143–152. IEEE, 2014. [Cited on page 14]

[HAC+16] Jennifer Horkoff, Fatma Başak Aydemir, Evellin Cardoso, Tong Li, Alejandro Maté, Elda Paja, Mattia Salnitri, John Mylopoulos, and Paolo Giorgini. Goal-oriented requirements engineering: a systematic literature map. In *IEEE 24th International Requirements Engineering Conference*, pages 106–115. IEEE, 2016. [Cited on page 17]

[HAC+17] Jennifer Horkoff, Fatma Başak Aydemir, Evellin Cardoso, Tong Li, Alejandro Maté, Elda Paja, Mattia Salnitri, Luca Piras, John Mylopoulos, and Paolo Giorgini. Goal-oriented requirements engineering: an extended systematic mapping study. *Requirements Engineering*, pages 1–28, 2017. [Cited on page 17]

[HLMN08] Charles B. Haley, Robin C. Laney, Jonathan D. Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008. [Cited on page 57]

[HML15] Jennifer Horkoff, Neil Maiden, and James Lockerbie. Creativity and goal modeling for software requirements engineering. In *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition*, pages 165–168. ACM, 2015. [Cited on page 13]

[HPTA14] Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali. The four pillars of crowdsourcing: A reference model. In *Research Challenges in Information Science (RCIS), 2014 IEEE Eighth International Conference on*, pages 1–12. IEEE, 2014. [Cited on page 2]

[HY16] Jennifer Horkoff and Eric Yu. Interactive Goal Model Analysis for Early Requirements Engineering. *Requirements Engineering*, 21(1):29–61, 2016. [Cited on page 16]

[IBM] IBM Rational DOORS product overview. `https://www.ibm.com/support/knowledgecenter/SSYQBZ_9.6.1/com.ibm.doors.requirements.doc/topics/c_welcome.html`. Accessed: 2016-03-15. [Cited on pages 15 and 43]

[Jac00] Michael Jackson. *Problem Frames: Analyzing and structuring software development problems*. Addison-Wesley Longman, Boston, 2000. [Cited on page 1]

[JGW10] Sami Jantunen, Donald C Gause, and Ragnar Wessman. Making sense of product requirements. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 89–92. IEEE, 2010. [Cited on page 26]

[JHP06] Audun Jøsang, Ross Hayward, and Simon Pope. Trust Network Analysis with Subjective Logic. In *Proceedings of the 29th Australasian Computer Science Conference - Volume 48*, ACSC '06, pages 85–94, Hobart, Australia, 2006. [Cited on page 65]

[JM15] Timo Johann and Walid Maalej. Democratic mass participation of users in requirements engineering? In *Proceedings of 23rd IEEE International Requirements Engineering Conference*, pages 256–261, 2015. [Cited on page 13]

[Jøs01] Audun Jøsang. A Logic for Uncertain Probabilities. *International Journal of Uncertainity, Fuzziness, and Knowledge-Based Systems*, 9(3):279–311, June 2001. [Cited on pages 57, 64, and 65]

[JSM+17] Timo Johann, Christoph Stanik, Walid Maalej, et al. Safe: A simple approach for feature extraction from app descriptions and app reviews. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 21–30. IEEE, 2017. [Cited on page 14]

[KDR+07] Lena Karlsson, Åsa G Dahlstedt, Björn Regnell, Johan Natt och Dag, and Anne Persson. Requirements engineering challenges in market-driven software development–an interview study with practitioners. *Information and Software technology*, 49(6):588–604, 2007. [Cited on pages 2, 10, and 26]

[KG17] Martina Z Huber Kolpondinos and Martin Glinz. Behind points and levels—the influence of gamification algorithms on requirements prioritization. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 332–341. IEEE, 2017. [Cited on page 18]

[KHS98] Erik Kamsties, Klaus Hörmann, and Maud Schlich. Requirements engineering in small and medium enterprises. *Requirements engineering*, 3(2):84–90, 1998. [Cited on page 26]

[KM17] Zijad Kurtanović and Walid Maalej. Mining User Rationale from Software Reviews. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 61–70. IEEE, 2017. [Cited on page 15]

[KMCS17] Georgi M Kanchev, Pradeep K Murukannaiah, Amit K Chopra, and Pete Sawyer. Canary: Extracting Requirements-Related Information from Online Discussions. In *Requirements Engineering Conference*, pages 31–40. IEEE, 2017. [Cited on pages 28, 37, and 45]

[KMP+17a] Fitsum Kifetew, Denisse Munante, Anna Perini, Angelo Susi, Alberto Siena, and Paolo Busetta. Dmgame: A gamified collaborative requirements prioritisation tool. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 468–469. IEEE, 2017. [Cited on page 18]

[KMP+17b] Fitsum Meshesha Kifetew, Denisse Munante, Anna Perini, Angelo Susi, Alberto Siena, Paolo Busetta, and Danilo Valerio. Gamifying collaborative prioritization: Does pointsification work? In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 322–331. IEEE, 2017. [Cited on page 18]

[LDLB16] Philipp Lombriser, Fabiano Dalpiaz, Garm Lucassen, and Sjaak Brinkkemper. Gamified requirements engineering: Model and experimentation. In *Proceedings of Requirements Engineering: Foundation for Software Quality*, pages 171–187. Springer, 2016. [Cited on page 18]

[Lef03] Dean Leffingwell. *Managing Software Requirements: a Use Case Approach*. Pearson Education India, 2003. [Cited on page 17]

[LF12] Soo Ling Lim and Anthony Finkelstein. Stakerare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation. *IEEE Transactions on Software Engineering*, 38(3):707–735, 2012. [Cited on page 12]

[LMSM10] Sotirios Liaskos, Sheila A McIlraith, Shirin Sohrabi, and John Mylopoulos. Integrating Preferences Into Goal Models for Requirements Engineering. In *18th IEEE International Requirements Engineering Conference*, pages 135–144. IEEE, 2010. [Cited on page 1]

[LPR93] Mitch Lubars, Colin Potts, and Charles Richter. A review of the state of the practice in requirements modeling. In *Proceedings of IEEE International Symposium on Requirements Engineering*, pages 2–14. IEEE, 1993. [Cited on pages 9 and 25]

[LQF10] Soo Ling Lim, Daniele Quercia, and Anthony Finkelstein. StakeNet: Using social networks to analyse the stakeholders of large-scale software projects. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering*, pages 295–304, 2010. [Cited on page 12]

[LVL04] Emmanuel Letier and Axel Van Lamsweerde. Reasoning About Partial Goal Satisfaction for Requirements and Design Engineering. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 53–62. ACM, 2004. [Cited on page 16]

[MAS16a] 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 *Proceedings of the 24th IEEE International Requirements Engineering Conference*, pages 176–185, 2016. [Cited on pages 2 and 13]

[MAS16b] 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 *Proceedings of the 24th IEEE International Requirements Engineering Conference*, pages 176–185, Beijing, September 2016. [Cited on page 13]

[MAS17] Pradeep K. Murukannaiah, Nirav Ajmeri, and Munindar P. Singh. Toward Automating Crowd RE. In *IEEE 25th International Requirements Engineering Conference*, pages 512–515, Lisbon, September 2017. [Cited on page 13]

[MHR09] Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. When users become collaborators: Towards continuous and context-aware user input. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 981–990. ACM, 2009. [Cited on pages 9 and 10]

[MKTS15a] Pradeep K. Murukannaiah, Anup K. Kalia, Pankaj R. Telang, and Munindar P. Singh. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference*, pages 156–165, 2015. [Cited on pages 13 and 57]

[MKTS15b] Pradeep K. Murukannaiah, Anup K. Kalia, Pankaj R. Telang, and Munindar P. Singh. Resolving goal conflicts via argumentation-based analysis of competing hypotheses. In *Proceedings of the 23rd IEEE International Requirements Engineering Conference*, pages 156–165, Ottawa, August 2015. [Cited on page 16]

[MM11] Laurie McLeod and Stephen G MacDonell. Factors that affect software systems development project outcomes: A survey of research. *ACM Computing Surveys (CSUR)*, 43(4):24, 2011. [Cited on page 1]

[MN15] Walid Maalej and Hadeer Nabil. Bug report, feature request, or simply praise? On automatically classifying app reviews. In *Proceedings of 23rd IEEE International Requirements Engineering Conference*, pages 116–125, 2015. [Cited on page 14]

[MNJR16] Walid Maalej, Maleknaz Nayebi, Timo Johann, and Guenther Ruhe. Toward data-driven requirements engineering. *IEEE Software*, 33(1):48–54, 2016. [Cited on pages 2 and 14]

[MPG15] Itzel Morales-Ramirez, Anna Perini, and Renata S. S. Guizzardi. An ontology of online user feedback in software engineering. *Applied Ontology*, 10(3-4):297–330, 2015. [Cited on page 13]

[MPP15] Itzel Morales-Ramirez, Dimitra Papadimitriou, and Anna Perini. Crowd intent: Annotation of intentions hidden in online discussions. In *Proceedings of 2nd IEEE/ACM International Workshop on CrowdSourcing in Software Engineering*, pages 24–29, 2015. [Cited on page 13]

[MRMK+17] 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 *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 424–429. IEEE, 2017. [Cited on page 17]

[MSJ+16] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. A Survey of App Store Analysis for Software Engineering. *IEEE Transactions on Software Engineering*, 2016. [Cited on page 14]

[MT86] Patricia Yancey Martin and Barry A Turner. Grounded theory and organizational research. *The Journal of Applied Behavioral Science*, 22(2):141–157, 1986. [Cited on page 5]

[MVM+15] Itzel Morales-Ramirez, Matthieu Vergne, Mirko Morandini, Anna Perini, and Angelo Susi. Exploiting online discussions in collaborative distributed requirements engineering. In *Proceedings of the Eighth International i*Workshop, iStar 2015, Ottawa, Canada, August 24-25, 2015.*, pages 7–12, 2015. [Cited on page 13]

[NE00] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: A roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000. [Cited on page 1]

[NS11] Mohd Hairul Nizam Nasir and Shamsul Sahibuddin. Critical success factors for software projects: A comparative study. *Scientific research and essays*, 6(10):2174–2186, 2011. [Cited on page 1]

[PB13] Dennis Pagano and Bernd Brugge. User Involvement in Software Evolution Practice: A Case Study. In *35th International Conference on Software Engineering*, pages 953–962. IEEE, 2013. [Cited on pages 1 and 12]

[PC14] Joonsuk Park and Claire Cardie. Identifying appropriate support for propositions in online user comments. In *Proceedings of the First Workshop on Argumentation Mining*, pages 29–38, 2014. [Cited on pages 56 and 76]

[PLA+14] Piotr Pruski, Sugandha Lohar, Rundale Aquanette, Greg Ott, Sorawit Amornborvornwong, Alexander Rasin, and Jane Cleland-Huang. Tiqi: Towards natural language trace queries. In *Requirements Engineering Conference (RE), 2014 IEEE 22nd International*, pages 123–132. IEEE, 2014. [Cited on page 15]

[PM13a] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *Proceedings of the 21st IEEE International Requirements Engineering Conference*, pages 125–134, 2013. [Cited on page 14]

[PM13b] Dennis Pagano and Walid Maalej. User feedback in the appstore: An empirical study. In *Proceedings of 2013 21st IEEE International Requirements Engineering Conference*, pages 125–134. IEEE, 2013. [Cited on page 26]

[PS13] Andreas Peldszus and Manfred Stede. From argument diagrams to argumentation mining in texts: A survey. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 7(1):1–31, 2013. [Cited on page 11]

[PSG+16] Sebastiano Panichella, Andrea Di Sorbo, Emitza Guzman, Corrado Aaron Visaggio, Gerardo Canfora, and Harald C. Gall. Ardoc: app reviews development oriented classifier. In *Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, FSE 2016, November 13-18, 2016*, pages 1023–1027, 2016. [Cited on page 14]

[PSRB07] 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 *Fifth International Workshop on Comparative Evaluation in Requirements Engineering*, pages 23–35. IEEE, 2007. [Cited on page 17]

[RB05] Björn Regnell and Sjaak Brinkkemper. Market-driven requirements engineering for software products. In *Engineering and managing software requirements*, pages 287–308. Springer, 2005. [Cited on pages 2, 10, and 26]

[RD92] Balasubramaniam Ramesh and Vasant Dhar. Supporting systems development by capturing deliberations during requirements engineering. *IEEE Transactions on Software Engineering*, 18(6):498–510, June 1992. [Cited on pages 10 and 57]

[RLvdW+16] 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 *IEEE 24th international Requirements engineering conference*, pages 196–205. IEEE, 2016. [Cited on page 76]

[Saa99] Thomas L Saaty. Fundamentals of the Analytic Network Process. In *Proceedings of the 5th international symposium on the analytic hierarchy process*, pages 12–14, 1999. [Cited on page 17]

[SC94] Anselm Strauss and Juliet Corbin. Grounded theory methodology. *Handbook of qualitative research*, 17:273–85, 1994. [Cited on page 5]

[SDB+15] Remco Snijders, Fabiano Dalpiaz, Sjaak Brinkkemper, Mahmood Hosseini, Raian Ali, and Atilla Ozum. REfine: A gamified platform for participatory requirements engineering. In *Proceedings of 1st International Workshop on Crowd-Based Requirements Engineering*, pages 1–6. IEEE, 2015. [Cited on page 18]

[SF14] Klaas-Jan Stol and Brian Fitzgerald. Two's company, three's a crowd: a case study of crowdsourcing software development. In *Proceedings of the 36th International Conference on Software Engineering*, pages 187–198. ACM, 2014. [Cited on page 10]

[SGM10] Norbert Seyff, Florian Graf, and Neil Maiden. Using mobile RE tools to give end-users their own voice. In *Proceedings of the 18th IEEE International Requirements Engineering Conference*, pages 37–46, 2010. [Cited on page 2]

[SRC05] Pete Sawyer, Paul Rayson, and Ken Cosh. Shallow knowledge as an aid to deep understanding in early phase requirements engineering. *IEEE Transactions on Software Engineering*, 31(11):969–981, November 2005. [Cited on page 14]

[SS13] Alistair Sutcliffe and Pete Sawyer. Requirements elicitation: Towards the unknown unknowns. In *Proceedings of 21st International Requirements Engineering Conference*, pages 92–104. IEEE, 2013. [Cited on pages 2 and 23]

[SSK99] Pete Sawyer, Ian Sommerville, and Gerald Kotonya. Improving market-driven re processes. In *VTT SYMPOSIUM*, volume 195, pages 222–236. VTT; 1999, 1999. [Cited on page 26]

[ST15] Richard Berntsson Svensson and Maryam Taghavianfar. Selecting creativity techniques for creative requirements: An evaluation of four techniques using creativity workshops. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 66–75. IEEE, 2015. [Cited on page 13]

[STC+15] 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 Applications*, 6(1):1–16, 2015. [Cited on page 12]

[Tou58a] Stephen E. Toulmin. *The Uses of Argument*. Cambridge University Press, Cambridge, UK, 1958. [Cited on pages 10, 56, and 73]

[Tou58b] Stephen Edelston Toulmin. *The use of argument*. Cambridge University Press, 1958. [Cited on page 11]

[vL01] Axel van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, pages 249–262, 2001. [Cited on page 1]

[vL09] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley, Chichester, UK, 2009. [Cited on pages 28, 56, and 73]

[Wie09] Roel Wieringa. Design science as nested problem solving. In *4th International Conference on Design Science Research in Information Systems and Technology*, page 8. ACM, 2009. [Cited on page 6]

[Wie14] Roel J Wieringa. *Design Science Methodology for Information Systems and Software Engineering*. Springer, 2014. [Cited on pages 4 and 5]

[WM17] Grant Williams and Anas Mahmoud. Mining twitter feeds for software user requirements. In *2017 IEEE 25th International Requirements Engineering Conference (RE)*, pages 1–10. IEEE, 2017. [Cited on page 14]

[WRB11] Krzysztof Wnuk, Björn Regnell, and Brian Berenbach. Scaling up requirements engineering–Exploring the challenges of increasing size and complexity in market-driven software development. In *Requirements Engineering: Foundation for Software Quality*, pages 54–59. Springer, 2011. [Cited on pages 2, 10, and 26]

[WRH+12] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012. [Cited on pages 42 and 71]

[WSF+10] Jon Whittle, William Simm, Maria-Angela Ferrario, Katerina Frankova, Laurence Garton, Andrée Woodcock, Baseerit Nasa, Jane Binner, and Aom Ariyatum. VoiceYourView: Collecting real-time feedback on the design of public spaces. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, pages 41–50, Copenhagen, 2010. [Cited on page 2]

[YFT+15] Yijun Yu, Virginia N.L. Franqueira, Thein Than Tun, Roel J. Wieringa, and Bashar Nuseibeh. Automated analysis of security requirements through risk-based argumentation. *Journal of Systems and Software*, 106:102–116, 2015. [Cited on page 10]

[YM98] Eric Yu and John Mylopoulos. Why goal-oriented requirements engineering. In *Proceedings of the 4th International Workshop on Requirements Engineering: Foundations of Software Quality*, volume 15, page 15–22, 1998. [Cited on page 24]

[Yu97] Eric S. K. Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Proceedings of the 3rd IEEE International Symposium on Requirements Engineering*, pages 226–235, 1997. [Cited on page 1]

## .1 Canary User Study

This appendix provides additional documents that helped to conduct user study of this thesis. Additionally it contains a technical report of the Canary compiler.

This part of the appendix shows the documents that aided the study of Chapter 3.

The documents are presented in the following order.

- First I present the demographic results of the study.
- The consent form used for the study.
- The participant information sheet.
- The participant instructions used in the study.
- Next I present the questionnaire the the participants of the study were asked to fill out as part of the study.
- In order to conduct the study I had to acquire an ethics approval. Lastly I present the approval documents of the ethics committee in Lancaster University Faculty of Science and Technology.

| | | |
|---|---|---|
| Gender | Female | 48.89% |
| | Male | 51.11% |
| Age | 18 to 24 | 11.11% |
| | 25 to 34 | 42.22% |
| | 35 to 45 | 28.89% |
| | 45 to 54 | 13.33% |
| | 55 or older | 4.44% |
| English proficiency | Very low | 0.00% |
| | Low | 0.00% |
| | Medium | 0.00% |
| | High | 6.67% |
| | Very high | 93.33% |
| Highest level of education | High school degree or equivalent (e.g., GED) | 6.67% |
| | Some college but no degree | 28.89% |
| | Bachelor degree | 37.78% |
| | Graduate degree | 20.00% |
| | Postgraduate degree | 6.67% |
| Familiarity with CS and SE concepts | Very low | 4.44% |
| | Low | 8.89% |
| | Medium | 44.44% |
| | High | 28.89% |
| | Very high | 13.33% |
| Social forum visit | Every day | 46.67% |
| | Once every two or three days | 20.00% |
| | About once a week | 22.22% |
| | About once a month or less often than that | 11.11% |

Lancaster University

***Consent Form***

Please read this information

You are invited to participate in a survey, entitled "Canary crowd evaluation" The study is being conducted by Georgi Kanchev and Amit Chopra from the Computing Department of the Faculty of Science and Technology in Lancaster University, and Pradeep Murukannaiah from Rochester Institute of Technology.

Lancaster University
Bailrigg
Lancaster
United Kingdom
LA1 4YW

The purpose of this study is to prove crowdsourcing as a feasible way of acquiring annotations over online discussions from social media. Risks to participants are considered minimal. There will be no costs for participating. You will be paid for each task you complete, but will not otherwise benefit from participating. Your Amazon account information will be kept while we collect data for payment purposes only. A limited number of research team members will have access to the data during this process. This information will be stripped from the final dataset.

Your participation in this survey is voluntary. You may decline to answer any question and you have the right to withdraw from participation at any time without penalty. If you wish to withdraw from the study or have any questions, contact the investigator listed above.

If you have any questions, please email Georgi Kanchev at g.kanchev@lancaster.ac.uk.

☐ I Agree

☐ I do not Agree

Lancaster University

# Participant Information Sheet

***Canary crowd evaluation***

My name is Georgi Kanchev and I am conducting this research in the Computing Deprtment of the Faculty of Science and Technology in Lancaster University, Lancaster, United Kingdom.

**What is the study about?**
The purpose of this study is to prove crowdsourcing as a feasible way of acquiring annotations over online discussions from social media. The discussions we have selected are about software applications. We will have expert software developers identify and annotate different features that people have discussed. Then we will have random, anonymous members of the crowd (AMT users) do the same. By comparing the two results we will be to assess the feasibility of using crowdsourcing for such purposes.

**Why have I been approached?**
You have been approached because the study requires information from a crowd of users. We chose Amazon Mechanical Turk as our mediator to connect us to crowd members.

**Do I have to take part?**
No. It's completely up to you to decide whether or not you take part in our study. You are free to quit in any time.

**What will I be asked to do if I take part?**
If you decide you would like to take part, you would be asked to read through an online discussion extracted from social media. While you are reading it we will ask you to annotate (highlight) sections of the discussion that contain one of four objects of interest to us.

- *Requirement*. You should highlight text as containing a requirement when it describes a non-existing feature in the application that is being discussed.
- *Solution*. You should highlight text as containing a solution when it describes an existing feature in the application that is being discussed.
- *Support*. You should highlight text as containing a support when it contains positive sentiment towards another object of interest
- *Non-support (rebuttal).* You should highlight text as containing a non-support when it contains negative sentiment towards another object of interest

More detail on this can be found in the instructions document.

**Will my data be Identifiable?**
The information you provide is confidential. The data collected for this study will be stored securely and only the researchers conducting this study will have access to this data:
Your Amazon account information will be kept while we collect data for payment purposes only. A limited number of research team members will have access to the data during this process. This information will be stripped from the final dataset.

**What will happen to the results?**
The results will be summarised and reported in a dissertation/thesis of one of the participating researchers (Georgi Kanchev) and will be summarized and submitted for publication in an academic or professional journal.

**Are there any risks?**
There are no risks anticipated with participating in this study. However, if you experience any distress following participation you are encouraged to inform the researcher and contact the resources provided at the end of this sheet.

**Are there any benefits to taking part?**
You will be paid for each task you complete through the Amazon Mechanical Turk payment system, but will not otherwise benefit from participating.

**Who has reviewed the project?**
This study [is being reviewed] by the Faculty of Science and Technology Ethics Committee, and [will be approved] by the Faculty of Science and Technology Ethics Committee at Lancaster University.

**Where can I obtain further information about the study if I need it?**
If you have any questions about the study, please contact:
Georgi Kanchev at email g.kanchev@lancaster.ac.uk
Or
Dr. Amit Chopra at email a.chopra1@lancaster.ac.uk

Thank you for taking the time to read this information sheet.

My name is Georgi Kanchev (g.kanchev@lancaster.ac.uk) and I am conducting this study as part of my PhD in the Computing Department of the Faculty of Science and Technology in Lancaster University, Lancaster, United Kingdom.

# What will I be asked to do if I take part?

You will be asked to read through an excerpt of an online discussion in which users are discussing some software application and highlight and annotate certain kinds of text in the discussion. We ask you to highlight four kinds of text: (1) requirement, (2) solution, (3) support, and (4) rebuttal. We describe each of these below.

You are not allowed to participate in this study in more than **two HITs**. Any HIT you submit after your second one will be rejected, regardless of its validity. **Please beware.**

## 1. Requirement

Each discussion usually begins with a user describing either some limitation or problem in how the application currently works or some enhancement that he or she would like to see in the application in the future. In either case, the user is implicitly or explicitly expressing a desire about how the application should work. This is called as a **requirement**. For example, imagine a discussion that could be about Amazon.com. A user says

*"On Amazon, every account requires a different email address. This is very painful as I often need to create temporary Amazon accounts, but I don't want to create a new email account for each. They should allow me to create different accounts with the same email address."*

**Requirements may be expressed anywhere in the discussion, not just at the beginning, so pay careful attention.** For example, deep down in the discussion, some other user expresses a requirement when she says

*"Amazon should allow me to link a new Amazon account with an existing Amazon account. I just don't get it why they are so slow on supporting this functionality."*

A few typical (but not exhaustive) templates of expressions of requirement are

*It would like be really cool if Amazon allowed/supported/ had a way of…*

*I want to be able to…*

*I tried doing something but could not find a way…*

*How do I do…*

*Google Maps should have/allow/support… …*

*I want to be able to … …*

A**ny** expression that may express desires towards what the system should do is a requirement.

Whenever you spot text that expresses a requirement, **highlight the text and label it as requirement.** We would also like you to note your **confidence** in the annotation using one of 5 levels - very low, low, medium, high, or very high. So an annotation for a requirement in which you are very confident would look like "requirement, very high".

## 2. **Solution**

Solutions often come up in response to expressions of requirement towards the system. In solutions, users refer to already existing ways that the requirement can be satisfied. It is common for users expressing solutions to briefly describe, in a tutorial-like fashion, how to accomplish the required task. Consider an example of solution shown below.

*"You can actually add multiple accounts linked to the same email if you make a special request for it. You have to go to Account-> Settings -> make a request. From then you have to fill out the form and just wait. They will give you a password for the new account and you'll still log in with the same email, but different password."*

**Solutions usually arise in the discussion after requirements.**

Common (but no exhaustive) templates for solutions are:

*You can…*

*You should…*

*There is a way of doing that…*

*A possible way of achieving that is…*

*This is already implemented, go to…*

*There is a workaround to solve that problem…*

Whenever you spot text that expresses a solution**, highlight the text and label it as Solution.** Remember to also include your **confidence** level (very low, low, medium, high, or very high).

## 3. Support

A support or a supporting argument is a term used to infer positive relationship between two statements. You should highlight text as containing a support when its primary purpose is to express support for an earlier comment in the discussion. Typically, such text will contain positive sentiment towards the object of interest that it is a response to (in the comment-reply structure of the discussions).

Sometimes support expressions are straight-forward as the one below:

*"This is a brilliant idea, I support that!"*

Or some are indirect as in:

*"I had never thought of it this way but it is entirely possible that you are right. I have no other option but to agree with you."*

*"Well, isn't that just a marvellous idea?"*

Examples can also be considered a way of supporting, for example:

*"Having two amazon accounts under the same email address would have been very useful to me the other day when my son wanted to use my account, but I didn't want to give him access to ALL my payment methods and purchase history"*

A supporting argument must support the comment it is a reply to.

Whenever you spot text that expresses a support**, highlight the text and label it as Support**, followed by your **confidence** level (very low, low, medium, high, or very high).

## 4. Rebuttal

A rebuttal or a rebutting argument is a term used to infer negative relationship between two statements. Rebuttals will try to contradict or disprove previous statements, sometimes by offering a counter argument. Typically, a rebuttal contains negative sentiment towards the object of interest that it is a response to (in the comment-reply structure of the discussions).

Rebuttals can also be very easy to notice and simple, like:

*"This is a horrible idea, I completely disagree!"*

Or more complex, like:

*"Even though you bring forward many compelling arguments to support your idea as valid, I fail to see your point of view and the value this idea will add to the application."*

A rebuttal argument must rebut the comment it is a reply to.

Supports and Rebuttals sometimes arise in response to each other forming a series of arguments. Make sure to follow the arguments in the discussion to the end.

Whenever you spot text that expresses a rebuttal**, highlight the text and label it as Rebuttal**, and your **confidence** level (very low, low, medium, high, or very high).

**Never use more than one annotation** in any comment. Sometimes, you may find that more than one annotation fits a comment**.** In such a case, give the text **one annotation that fits best.**

On the other hand some comments might not contain any object of interest at all, in which case you should leave it without annotation.

Please download and read through the full participant information sheet from here.

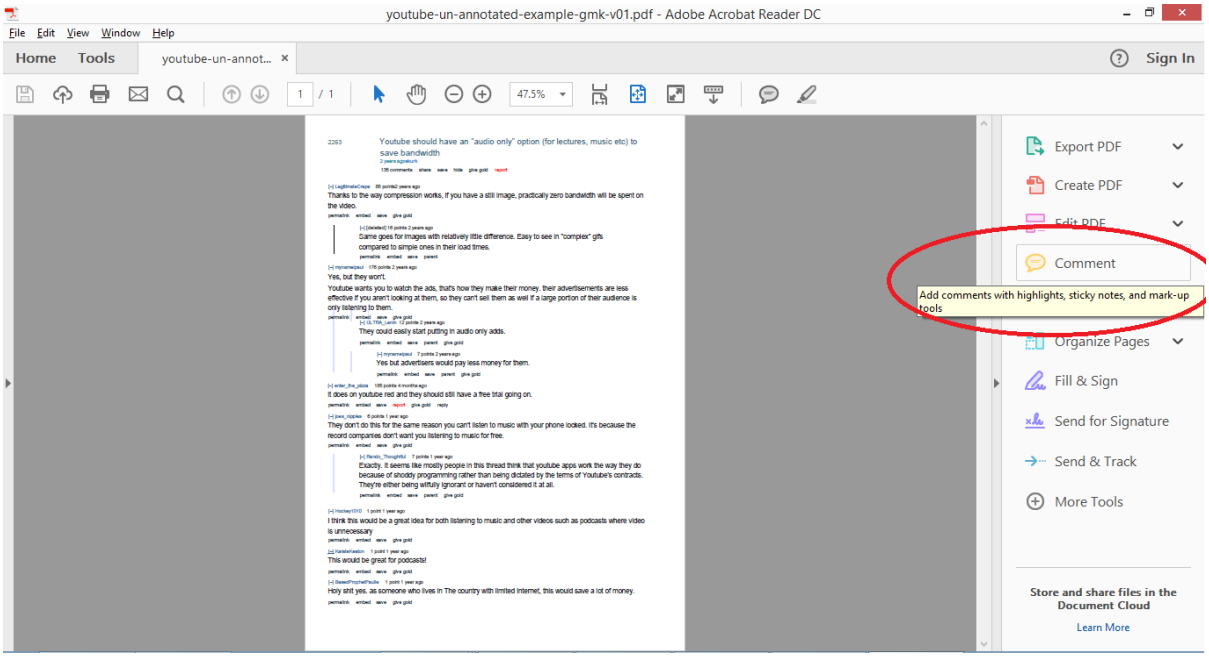
An annotated example of a discussion can be downloaded from here. Note that in the examples every comment contains annotations. This only for illustration purposes and the discussion allocated to you may not contain information worth annotating.

## How do I perform annotations?

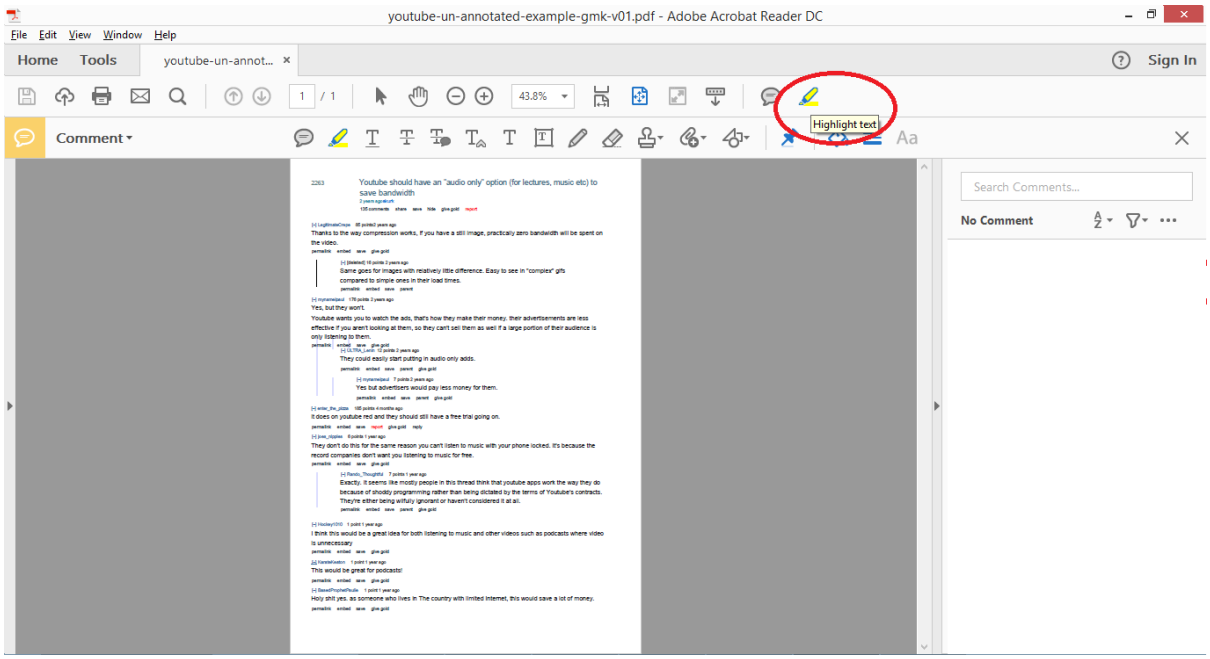
**Step 1.** In order to complete this task, you will need to have Adobe Acrobat Reader DC installed. Adobe Reader is free to install and can be acquired from the following link: https://get.adobe.com/uk/reader/.

For the examples below we use Adobe Acrobat Reader (15.023.20056) to illustrate via screenshots what the annotating process should look like and what we are looking for in a successfully finished task.

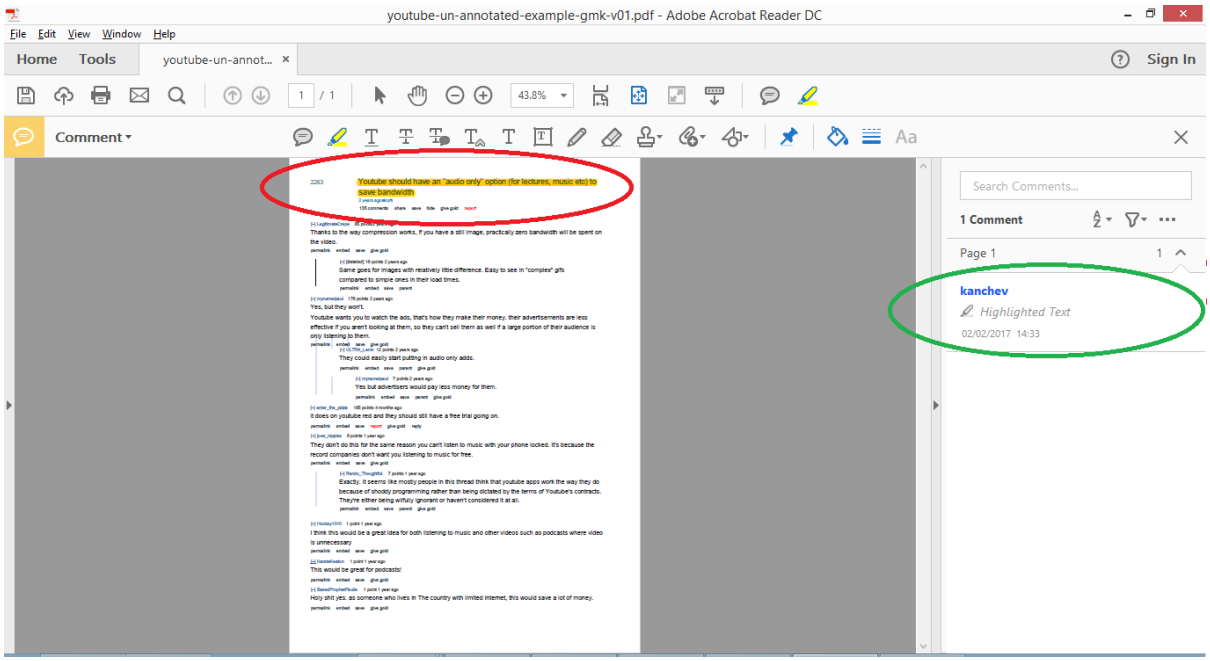
**Step 2.** Click on the comments tab on the sidebar of Adobe Reader as illustrated below.



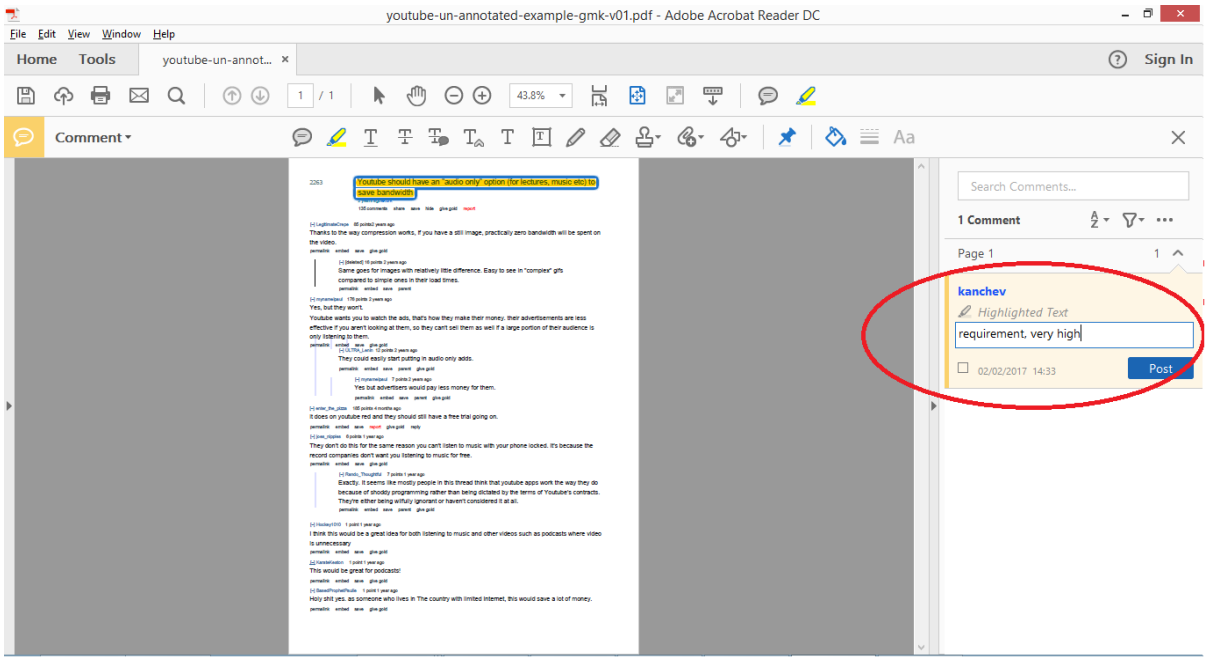

**Step 3.** Click on the text highlighter icon on the toolbar menu in order to start highlighting sections of the discussion you think contain objects of interest (annotations to the pdf).



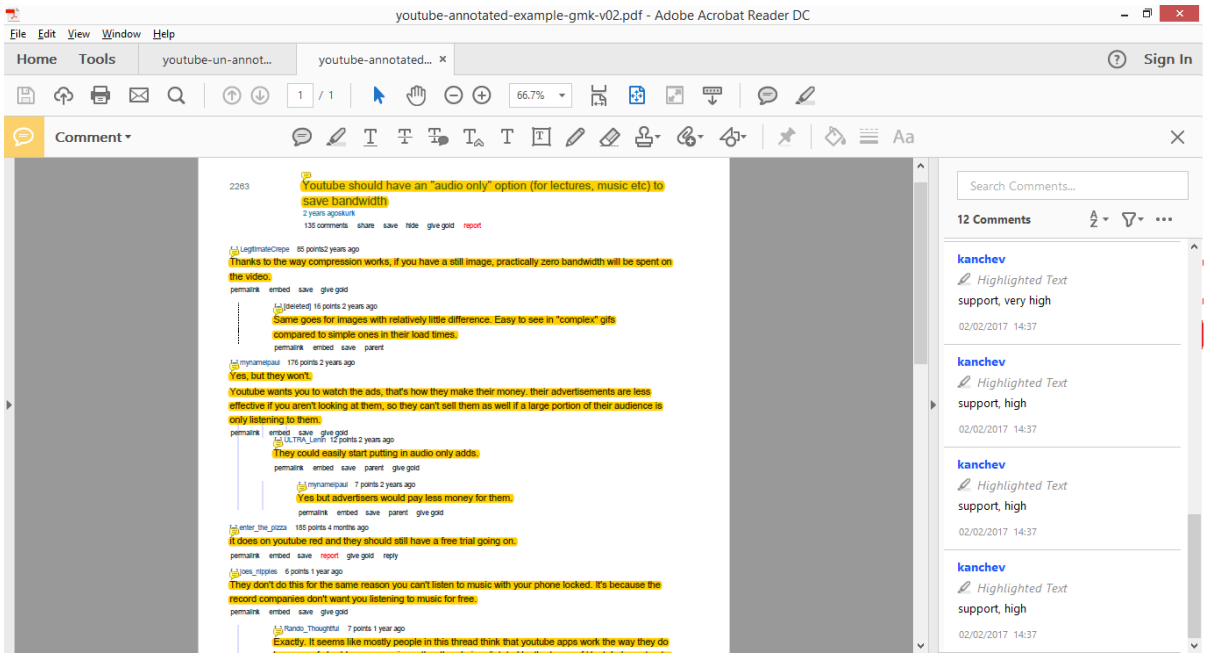

**Step 4.** Click and drag over text to highlight it. The text you have highlighted should show up in yellow, as circled in red. Circled in green is the new entity in the comments tab on the left. You will use that in the next step.




**Step 5.** Double clicking on the new entity in the comment section (circled in red) will allow you to write a comment. In this step, you should specify exactly which object of the four objects of interest we are looking for you have observed in the highlighted text and your confidence level. When you have specified click the "post" button (blue colour, located in the red highlight in the image above). The four objects of interest we are looking for are explained above in the "**What will I be asked to do if I take part?**" section.




**Step 6.** A finished document will contain annotated highlights with all the objects of interest you have observed in the discussion given to you. Make sure to be objective, thorough, and as accurate as possible when highlighting the discussion allocated to you. Then save the changes to the document and re-upload it to complete the task.




**Important:** You should highlight **all instances** of the above objects of interest in the entire document we provide you. The document may contain zero or more instances of each of the four types of objects of interest.

**Thank you!**

## Participant Information

My name is Georgi Kanchev and I am conducting this study as part of my PhD in the Computing Department of the Faculty of Science and Technology in Lancaster University, Lancaster, United Kingdom.

I would like to emphasize that each worker is only allowed to participate in TWO HITs of this kind (coming from this requester). Any more will be rejected automatically, regardless of their validity.

Please download and read through the full participant information sheet from here.

## Consent Form

Please donwload and read the consent form from here. If you have any questions, please email Georgi Kanchev at g.kanchev@lancaster.ac.uk.

Before you proceed please agree to the consent form. If you diagree, please abort you participation to this study.

☐ I Agree

☐ I do not Agree

Please fill out the following questions **before** you proceed to the main task.

**1. What gender do you identify with from the listed:**

- ○ Male
- ○ Female
- ○ Other

**2. Which of the following categories includes your age?**

- ○ 18 to 24
- ○ 25 to 34
- ○ 35 to 45
- ○ 45 to 54
- ○ 55 or older

**3. What language do you speak most often?**

**4. How proficient are you in reading and writing in English?**

- Very low
- Low
- Medium
- High
- Very high

**5. What is the highest level of school you have completed or the highest degree you have received?**

- Less than high school degree
- High school degree or equivalent (e.g., GED)
- Some college but no degree
- Bachelor degree
- Graduate degree
- Postgraduate degree

**6. If you went to college, what was your major?**

**7. Have you worked for a technology company?**

- No
- Yes; for less than a year
- Yes; for one to five years
- Yes; for more than five years

**8. How do you rate your familiarity of concepts related to Computer Science, Information Technology, and Software Engineering?**

- Very low
- Low
- Medium
- High
- Very high

**9. How often do you use social media (e.g., Facebook, Google Plus, Flickr, Twitter, LinkedIn, etc.)?**

- Every day
- Once every two or three days
- About once a week
- About once a month or less often than that

- Never used social media

**10. How often do you visit social forums (e.g., Reddit, Stackoverflow, Quora, etc.)**

- Every day
- Once every two or three days
- About once a week
- About once a month or less often than that
- Never used social media

**Main Task**

Please download detailed instructions on how to succesfully complete the main task here.

Then download the pdf file of the discussion you will be asked to annotate from here.

An example of a fully annotated discussion can be seen here. Note that in this example all comments are annotated as containing an object of interest. This is for illustrative purposes only and the discussion allocated to you may not contain as many objects.

To successfully complete the main task please upload your work (annotated pdf file) using (right click and open in new window) the following link. Make sure to type in your worker ID correctly.

If you encounter any problems during the upload or you believe for any reason that the upload didn't work please email the file at g.kanchev@lancaster.ac.uk

Please fill out the following questions **after** you you have finished the main task.

**11. How long did the main tasks (excluding pre and post surveys) take in hours:minutes?**

**12. How do you rate the difficulty of the main tasks, overall?**

- Very low
- Low
- Medium
- High
- Very high

**13. How well did you understand the concept of a requirement?**

- Very low
- Low
- Medium
- High

- Very high

**14. How well did you understand the concept of a solution?**

- Very low
- Low
- Medium
- High
- Very high

**15. How well did you understand the concept of a support?**

- Very low
- Low
- Medium
- High
- Very high

**16. How well did you understand the concept of a rebuttal?**

- Very low
- Low
- Medium
- High
- Very high

**17. How well did you understand the annotation instructions?**

- Very low
- Low
- Medium
- High
- Very high

**18. Please provide any additional comments you have below.**

End! Thank you for participating in this study!

Lancaster University

## *Faculty of Science and Technology Research Ethics Committee (FSTREC)*
## *Lancaster University*

### *Application for Ethical Approval for Research*

**This form should be used for all projects by staff and research students, whether funded or not, which have not been reviewed by any external research ethics committee.** If your project is or has been reviewed by another committee (e.g. from another University), please contact the FST research ethics officer for further guidance.

In addition to the completed form, you need to submit **research materials** such as:

i. Participant information sheets
ii. Consent forms
iii. Debriefing sheets
iv. Advertising materials (posters, e-mails)
v. Letters/emails of invitation to participate
vi. Questionnaires, surveys, demographic sheets that are non-standard
vii. Interview schedules, interview question guides, focus group scripts

Please note that **you DO NOT need to submit pre-existing questionnaires or standardized tests** that support your work, but which cannot be amended following ethical review. These should simply be referred to in your application form.

Please submit this form and any relevant materials **by email as** a **SINGLE attachment** to *fst-ethics@lancaster.ac.uk*

---

## Section One

### *Applicant and Project Information*

***Name of Researcher: Georgi Kanchev***

***Project Title: Canary crowd evaluation***

***Level: PhD***

***Supervisor: Dr. Amit Chopra***

***Researcher's Email address: g.kanchev@lancaster.ac.uk***
***Telephone: +359 7934 440 685***
***Address: 59 Dale Street, Lancaster, Lancashire, United Kingdom LA1 3AP***

***Names and appointments/position of all further members of the research team:***

***Is this research externally funded? If yes,***

***ACP ID number:***
***Funding source:***
***Grant code:***

***Does your research project involve any of the following?***

☑ Human participants (including all types of interviews, questionnaires, focus groups, records relating to humans, use of internet or other secondary data, observation etc.)

☐Animals - the term animals shall be taken to include any non-human vertebrates, cephalopods or decapod crustaceans.

☐Risk to members of the research team e.g. lone working, travel to areas where researchers may be at risk, risk of emotional distress

☐Human cells or tissues other than those established in laboratory cultures

☐Risk to the environment

☐Conflict of interest

☐Research or a funding source that could be considered controversial

☑ Social media and/or data from internet sources that could be considered private

☐any other ethical considerations

**Yes – complete the rest of this form**

**No – go to Section Five**

## Section Two

### *Type of study*

☒ Includes *direct* involvement by human subjects. ***Complete all sections apart from Section 3.***

☐ Involves *existing documents/data only*, or the evaluation of an existing project with no direct contact with human participants. ***Complete all sections apart from Section 4.***

### *Project Details*

**1. Application is for an individual study ☑ for a programme of studies ☐**

**2. Anticipated project dates (month and year)**
**Start date: End date:**

**3. Please briefly describe the background to the research (no more than 150 words, in lay-person's language):**

Requirements engineering (RE) is a sub-part of software engineering that specifically deals with understanding user needs. Traditional RE practices implore various techniques, such as collocated interviews, modelling, and prototyping. The result of the RE process is an explicit, written list of user requirements. In our research we are interested in studying ways of methodically incorporating crowd-generated information into the RE process.

Software users often discuss their problems and requirements with a piece of software, e.g., Google Maps, on social media sites such as Reddit and Google Forums. However, much of this information is not fed back easily into the improvement of the software, primarily because it is voluminous and not easy to search and categorize. We are currently devising a methodological approach to structure and query (search) such information.

**4. Please state the aims and objectives of the project (no more than 150 words, in lay-person's language):**

We aim to take advantage of the voluminous data generated by user interaction in social media. This project is focused on extracting software requirements, their potential solutions, and arguments about them from discussions on social media site such as Reddit. We are developing a language which can be used to easily and systematically search a database of such artifacts. This tool will enable developers to know, e.g., the most discussed and the most popular requirements taking advantage of social media features such as the discussion and upvotes/downvotes, and so on.

In this project we will directly evaluate the feasibility of using the crowd (an anonymous group of random users) to annotate discussions from social media in the form of English natural language. The annotations that we would require the users to apply over the data fall into four categories: requirement, solution, support, and rebuttal.

**5. Methodology and Analysis:**

One of the challenges we face is acquiring annotations. Online discussions are publicly available to us in social media, but they are completely unstructured. We have come up with a way of structuring the data using annotations, but applying them to the raw data requires intelligence.

This particular approval is for testing the feasibility of using the crowd as a method for gathering annotations. To test this, we will make the crowd annotate available discussions about certain applications from Reddit on Amazon Mechanical Turk (AMT). AMT defines itself as a crowdsourcing Internet marketplace enabling individuals and businesses (known as Requesters) to coordinate the use of human intelligence to perform tasks that computers are currently unable to do. In other words, we can use AMT to delegate small tasks to random people. AMT provides a very convenient way to dispatch these tasks to people and to reward them with a monetary compensation for each completed task.

Below we describe the entire process of this study in detail.

Step 1. Take a forum about an application such as Google Maps from Reddit (a social media site), extract its data using their own publicly available API (the data on the forum is **public, not private**).

Step 2. Annotate plain text comments into categories such as requirements, solutions, supportive arguments, and unsupportive arguments and store them. These annotations we will do ourselves, as experts in RE. We will take these annotations as ground truth.
Step 3. Split the discussions into small, manageable chunks and prepare them to be dispatched as AMT tasks.
Step 4. Dispatch the tasks using AMT and collect crowd-generated annotations.
Step 5. Evaluate the crowd annotations against the ground truth (expert annotations). High precision and recall would prove using crowdsourcing as a feasible approach to acquire annotations.
Step 6. Run queries against this data in Canary, our query language, and see check if useful data is returned.

## Section Three

### *Secondary Data Analysis*

**Complete this section if your project involves *existing documents/data only*, or the evaluation of an existing project with no direct contact with human participants**

1. Please describe briefly the data or records to be studied, or the evaluation to be undertaken.

2. How will any data or records be obtained?

3. Confidentiality and Anonymity: If your study involves re-analysis and potential publication of existing data but which was gathered as part of a previous project involving direct contact with human beings, how will you ensure that your re-analysis of this data maintains confidentiality and anonymity as guaranteed in the original study?
No personal information will be collected from the members of the crowd who participate in the study.

4. What plan is in place for the storage of data (electronic, digital, paper, etc)? Please ensure that your plans comply with the Data Protection Act 1998.

5. What are the plans for dissemination of findings from the research?

6a. Is the secondary data you will be using in the public domain? YES/NO
6b. If NO, please indicate the original purpose for which the data was collected, and comment on whether consent was gathered for additional later use of the data.

7.What other ethical considerations (if any), not previously noted on this application, do you think there are in the proposed study? How will these issues be addressed?

8a. Will you be gathering data from discussion forums, on-line 'chat-rooms' and similar online spaces where privacy and anonymity are contentious? YES/NO

If yes, your project requires full ethics review. Please complete all sections.

## Section Four

## *Participant Information*

**Complete this section if your project includes *direct* involvement by human subjects.**

1. Please describe briefly the **intended human participants** (including number, age, gender, and any other relevant characteristics):

Age and gender are irrelevant for our purposes and will not be taken into account during the recruitment process. Since AMT requires payment per allocated task, the exact number of allocated tasks will depend on the amount of funding we attract. The minimum task allocations we would aim for is around 800. Since we are using AMT as a mediator, we don't have direct control over selecting the participants.

2. How will participants be **recruited** and from where?

We will use a service called Amazon Mechanical Turk (AMT) to recruit random anonymous members of the crowd. AMT doesn't provide you with a way to select individuals in your crowd, so we have no control over it.

3. Briefly describe your **data collection methods**, drawing particular attention to any potential ethical issues.

Data will be collected via a web application designed and developed for the purposes of this research alone. The data will be backed up on the provided by university Box storage. Additionally the data will be stored in an SQL database on Georgi Kanchev's university issued laptop for testing queries. The laptop is password protected and encrypted. No particular ethical issues are evident.

**4. Consent**

4a. Will you take all necessary steps to **obtain the voluntary and informed consent** of the prospective participant(s) or, in the case of individual(s) not capable of giving informed consent, the permission of a legally authorised representative in accordance with applicable law? **YES**

If yes, please go to question 4b. If no, please go to question 4c.

4b. Please explain the procedure you will use for **obtaining consent**?. If applicable, please explain the procedures you intend to use to gain permission on behalf of participants who are unable to give informed consent.

The first page of the online survey will be the consent document. The online consent has all of the elements of a regular consent, but it will not require a signature. Participants will either click an "I Agree" or an "I do not Agree" box. The "I Agree" box will take them into the survey. The "I do not agree" box will thank them for their time and take them away from the survey. The body of the consent form can be found attached at the end of this form.

4c. If it will be necessary for participants to take part in the study **without their knowledge and consent at the time**, please explain why (for example covert observations may be necessary in some settings; some experiments require use of deception or partial deception – not telling participants everything about the experiment).

5. Could participation cause **discomfort** (physical and psychological eg distressing, sensitive or embarrassing topics), **inconvenience or danger beyond the risks encountered in normal life**? Please indicate plans to address these potential risks. State the timescales within which participants may withdraw from the study, noting your reasons.

No. A task will take no more than fifteen minutes. AMT users are always free to leave the task unfinished.

6. How will you protect participants' **confidentiality and/or anonymity** in data collection (e.g. interviews), data storage, data analysis, presentation of findings and publications?

No personal data about AMT users will be collected except user identification from AMT. These identifications are necessary for successful payment to the user. Even though the id can be considered as personally identifiable information, we have no plans on keeping the ids after payment. After that they will be replaced by anonymous ids in our records. We will not keep any identifiable information about any AMT user. A short questionnaire with demographic and opinion information will be collected from participants at the beginning and end of their participation. No identifiable information will be required in the questionnaire. The questions are also submitted with this form, attached at the end.

7. Do you anticipate any ethical constraints relating to **power imbalances or dependent relationships**, either with participants or with or within the research team? If yes, please explain how you intend to address these?

No.

8. What potential **risks may exist for the researcher** and/or research team? Please indicate plans to address such risks (for example, noting the support available to you/the researcher; counselling considerations arising from the sensitive or distressing nature of the research/topic; details of the lone worker plan you or any researchers will follow, in particular when working abroad.

None.

9. Whilst there may not be any significant direct **benefits to participants** as a result of this research, please state here any that may result from participation in the study.

None.

10. Please explain the **rationale for any incentives/payments** (including out-of-pocket expenses) made to participants:

Amazon Mechanical Turk has a well-established rewarding system for the users of their crowd. Each task is allocated a set amount of money to be paid to the account of the user. We will allocate ~$2,50 for each task (each task will be of proportionate size to the reward).

11. What are your plans for the **storage of data** (electronic, digital, paper, etc.)? Please ensure that your plans comply with the Data Protection Act 1998.

All data will be backed up in Box and will also be stored in an encrypted SQL database with a secure password on a university-administered computer. The data will be stored in those locations so that it is accessible to the participating researches to carry out their work on it. Upon completion of Georgi Kanchev's PhD (expected March 2018), copies of the data on his personal computer will be erased. A copy of the data will remain on the University's repository for 10 years for replication and further research. Dr. Amit Chopra (Kanchev's supervisor) will be responsible for this copy.

23/06/2016

12. Please answer the following question *only* if you have not completed a Data Management Plan for an external funder.

12.a How will you make your data available under open access requirements?

A copy of the raw data will also be deposited in Lancaster University's institutional data repository and made freely available with an appropriate data license. Lancaster University uses Pure as the data repository which will hold, manage, preserve and provide access to datasets produced by Lancaster University research. We will keep the data in public access for a period of 10 years, so that anyone who is interested can replicate our research for rebuttal purposes, or further extend our studies.

12b. Are there any restrictions on sharing your data for open access purposes?

None.

13. Will **audio or video recording** take place? ☑ no ☐ audio ☐ video

13a. Please confirm that portable devices (laptop, USB drive etc) will be **encrypted** where they are used for identifiable data. If it is not possible to encrypt your portable devices, please comment on the steps you will take to protect the data.

13b. What arrangements have been made for **audio/video data storage**? At what point in the research will tapes/digital recordings/files be destroyed?

13c. If your study includes video recordings, what are the implications for participants' anonymity? Can anonymity be guaranteed and if so, how? If participants are identifiable on the recordings, how will you explain to them what you will do with the recordings? How will you seek consent from them?

14. What are the plans for dissemination of findings from the research? If you are a student, mention here your thesis. Please also include any impact activities and potential ethical issues these may raise.

The findings of this research study will be collected and published into a conference or journal paper. Further after that the findings will be used in the PhD thesis of one of the participating researchers, Georgi Kanchev. There are no foreseeable ethical issues raised by our dissemination, as we will not be collecting any personal information from participants.

15. What particular ethical considerations, not previously noted on this application, do you think there are in the proposed study? Are there any matters about which you wish to seek guidance from the FSTREC?

We believe all ethical considerations have been previously mentioned on this application.

## Section Five

## *Additional information required by the university insurers*

If the research involves either the nuclear industry or an aircraft or the aircraft industry (other than for transport), please provide details below:

## Section Six

## *Declaration and Signatures*

I understand that as Principal Investigator/researcher/PhD candidate I have overall responsibility for the ethical management of the project and confirm the following:

- I have read the Code of Practice, Research Ethics at Lancaster: a code of practice and I am willing to abide by it in relation to the current proposal.
- I will manage the project in an ethically appropriate manner according to: (a) the subject matter involved and (b) the Code of Practice and Procedures of the University.
- On behalf of the University I accept responsibility for the project in relation to promoting good research practice and the prevention of misconduct (including plagiarism and fabrication or misrepresentation of results).
- On behalf of the University I accept responsibility for the project in relation to the observance of the rules for the exploitation of intellectual property.
- If applicable, I will give all staff and students involved in the project guidance on the good practice and ethical standards expected in the project in accordance with the University Code of Practice. (Online Research Integrity training is available for staff and students here.)
- If applicable, I will take steps to ensure that no students or staff involved in the project will be exposed to inappropriate situations.

☑ Confirmed

**Please note:** If you are not able to confirm the statement above please contact the FST Research Ethics Committee and provide an explanation.

**Student applicants:**
Please tick to confirm that you have discussed this application with your supervisor, and that they agree to the application being submitted for ethical review ☒
***Students must submit this application from your Lancaster University email address, and copy your supervisor in to the email in which you submit this application***

**All applicants (Staff and Students) must complete this declaration:**
**I confirm that I have sent a copy of this application to my Head of Department** (or their delegated representative) . **Tick here to confirm** ☒
**Name of Head of Department** *(or their delegated representative)*

23/06/2016

Professor Jon Whittle

**Applicant electronic signature**: Click here to enter text. Date 29.09.2016

Georgi Kanchev

***Presurvey Questions***

1. What is your gender?

   Male|Female|Other

2. Which of the following categories includes your age?

   18 to 24|25 to 34|35 to 45|45 to 54|55 or older

3. What language do you speak most often?

   Text, e.g., English, Spanish, Hindi, Mandarin, etc.

4. How proficient are you in reading and writing in English?

   Very low|Low|Medium|High|Very high

5. What is the highest level of school you have completed or the highest degree you have received?

   Less than high school degree|High school degree or equivalent (e.g., GED)|Some college but no degree|Bachelor degree|Graduate degree

6. If you went to college, what was your major?,

   Text, e.g., computer science, mechanical engineering, psychology, music, law, etc.

7. Have you worked for a technology company?

   No|Yes; for less than a year|Yes; for one to five years|Yes; for more than five years, 1

8. How do you rate your familiarity of concepts related to Computer Science, Information Technology, and Software Engineering?

   Very low|Low|Medium|High|Very high

9. How often do you use social media (e.g., Facebook, Google Plus, Flickr, Twitter, LinkedIn, etc.)?

   Almost every day|About once a week| About once a month or less often than that| Never

used social media

10. How often do you visit social forums (e.g., Reddit, Stackoverflow, Quora, etc.)

    Almost every day|About once a week| About once a month or less often than that| Never used social forums

***Postsurvey Questions***

1. How long did the main tasks (excluding pre and post surveys) take?

   Text, duration in hours:minutes, e.g., 00:30 for thirty minutes

2. How do you rate the difficulty of main tasks?

   Very easy|Easy|Medium|High|Very high

3. Please provide any additional comments you have below

   <Text>

***Consent Form***

Please read this information

You are invited to participate in a survey, entitled "Canary crowd evaluation" The study is being conducted by Georgi Kanchev and Amit Chopra from the Computing Department of the Faculty of Science and Technology in Lancaster University, and Pradeep Murukannaiah from Rochester Institute of Technology.

Lancaster University
Bailrigg
Lancaster
United Kingdom
LA1 4YW

The purpose of this study is to prove crowdsourcing as a feasible way of acquiring annotations over online discussions from social media. Risks to participants are considered minimal. There will be no costs for participating. You will be paid for each task you complete, but will not otherwise benefit from participating. Your Amazon account information will be kept while we collect data for payment purposes only. A limited number of research team members will have access to the data during this process. This information will be stripped from the final dataset.

Your participation in this survey is voluntary. You may decline to answer any question and you have the right to withdraw from participation at any time without penalty. If you wish to withdraw from the study or have any questions, contact the investigator listed above.

If you have any questions, please email Georgi Kanchev at g.kanchev@lancaster.ac.uk.

☐ I Agree

☐ I do not Agree

## *Participant Information Sheet*

**Participant Information Sheet**

***Canary crowd evaluation***

My name is Georgi Kanchev and I am conducting this research in the Computing Deprtment of the Faculty of Science and Technology in Lancaster University, Lancaster, United Kingdom.

**What is the study about?**
The purpose of this study is to prove crowdsourcing as a feasible way of acquiring annotations over online discussions from social media. The discussions we have selected are about software applications. We will have expert software developers identify and annotate different features that people have discussed. Then we will have random, anonymous members of the crowd (AMT users) do the same. By comparing the two results we will be to assess the feasibility of using crowdsourcing for such purposes.

**Why have I been approached?**
You have been approached because the study requires information from a crowd of users. We chose Amazon Mechanical Turk as our mediator to connect us to crowd members.

**Do I have to take part?**
No. It's completely up to you to decide whether or not you take part in our study. You are free to quit in any time.

**What will I be asked to do if I take part?**
If you decide you would like to take part, you would be asked to read through an online discussion extracted from social media. While you are reading it we will ask you to annotate (highlight) sections of the discussion that contain one of four objects of interest to us.

- *Requirement*. You should highlight text as containing a requirement when it describes a non-existing feature in the application that is being discussed.
- *Solution*. You should highlight text as containing a solution when it describes an existing feature in the application that is being discussed.
- *Support*. You should highlight text as containing a support when it contains positive sentiment towards another object of interest
- *Non-support (rebuttal).* You should highlight text as containing a non-support when it contains negative sentiment towards another object of interest

**Will my data be Identifiable?**
The information you provide is confidential. The data collected for this study will be stored securely and only the researchers conducting this study will have access to this data:
Your Amazon account information will be kept while we collect data for payment purposes only. A limited number of research team members will have access to the data during this process. This information will be stripped from the final dataset.
**What will happen to the results?**

The results will be summarised and reported in a dissertation/thesis of one of the participating researchers (Georgi Kanchev) and will be summarized and submitted for publication in an academic or professional journal.

**Are there any risks?**
There are no risks anticipated with participating in this study. However, if you experience any distress following participation you are encouraged to inform the researcher and contact the resources provided at the end of this sheet.

**Are there any benefits to taking part?**
You will be paid for each task you complete through the Amazon Mechanical Turk payment system, but will not otherwise benefit from participating.

**Who has reviewed the project?**
This study [is being reviewed] by the Faculty of Science and Technology Ethics Committee, and [will be approved] by the Faculty of Science and Technology Ethics Committee at Lancaster University.

**Where can I obtain further information about the study if I need it?**
If you have any questions about the study, please contact:
Georgi Kanchev at email g.kanchev@lancaster.ac.uk
Or
Dr. Amit Chopra at email a.chopra1@lancaster.ac.uk

Thank you for taking the time to read this information sheet.

## .2 Canary Compiler Technical Report

This part of the appendix provides the technical report that details the implementation in MySQL the Canary query language presented in Chapter 4.

# Canary: MySQL Query Definitions

## I. Formal Syntax and Semantics

In this section, we will describe the language formally.

### A. Syntax

Grammar 1 defines the syntax of Canary. The queries can be a mixture of both requirements engineering entities, such as *requirement* or *solution*, and argumentative entities, such as *support* or *rebuttal*.

⟨*query*⟩ ::= ⟨*expr*⟩
| ⟨*arg_expr*⟩

⟨*expr*⟩ ::= ⟨*req_expr*⟩
| ⟨*sol_expr*⟩

⟨*req_expr*⟩ ::= requirement
| requirement where ⟨*condition*⟩
| ⟨*aggregator*⟩ ( ⟨*req_expr*⟩ )

⟨*sol_expr*⟩ ::= solution ( ⟨*req_expr*⟩ )
| solution ( ⟨*req_expr*⟩ ) where ⟨*condition*⟩
| ⟨*aggregator*⟩ ( ⟨*sol_expr*⟩ )

⟨*arg_expr*⟩ ::= ⟨*arg_entity*⟩ ( ⟨*expr*⟩ )
| ⟨*arg_entity*⟩ ( ⟨*expr*⟩ ) where ⟨*condition*⟩

⟨*arg_entity*⟩ ::= support
| rebuttal

⟨*aggregator*⟩ ::= popular
| unpopular
| controversial
| discussed

Grammar 1: Syntax of Canary

### B. Semantics

Listing 1 shows a relational database schema that realizes the conceptual model of Canary in Figure 1. In the schema, all attributes beginning comID in all relations other than comment are foreign key references to comID in comment.

Listing 1: Example schema identifying the underlying database

```
\\social
  user(iduser, username, urole, reputation)
    key iduser
  comment(idcomment, text, score, iduser,
      created, id, parent)
    key idcomment
\\requirement
  requirement(idrequirement, idcomment,
      start_char, end_char)
    key(reqID)
  solution(idrsolution, idcomment,
      start_char, end_char)
    key(solID)
\\argumentation
  support(idsupport, idcomment, start_char,
      end_char)
    key(suppID)
  rebuttal(idrebuttal, idcomment, start_char,
      end_char)
    key(rebID)
```

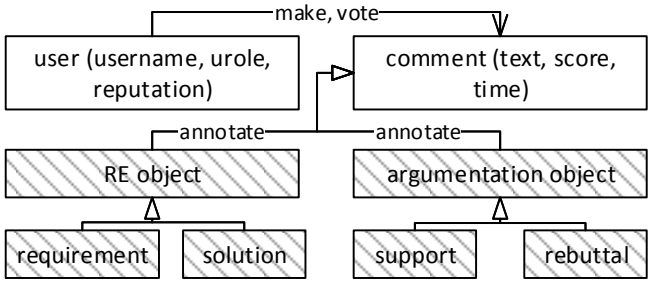



Fig. 1: Conceptual model of information considered in Canary

Figure 2 shows an example discussion that instantiates the framework. In the example, users are discussing features and requirements of Google Maps. The example is extracted from Reddit. John suggests a requirement about being able to save addresses in Google Maps. The requirements gets a score of 805. Mary expresses support for the requirement. Henry and Patrick both propose solutions for the requirement (and get a score of their own, which is a result of summing upvotes and downvotes). Henry's solution attracts rebuttal comments in addition to a score. Patrick's solution attracts support and a score. In general, each comment may attract up and down votes (resulting in a positive/negative score), as explained above.

Figure 3 shows a partial diagram of the schema. The diagram is only partial because some tables don't have all records listed, such as *user*, and *comment*. The records in those tables are repetitive and do not contribute to the conceptual understanding of the schema, so they were left out for simplicity.

The semantics of every expression in the language of Grammar 1 is given as an SQL query. Formally, for any such expression $x$ in Grammar 1, the function $\mathsf{SQL}(x)$ gives the SQL query that $x$ maps to. Below, we define SQL inductively from the simplest expressions to the most complex ones. For concreteness, we give the SQL queries in the MySQL dialect of SQL. MySQL requires that subqueries that appear in the

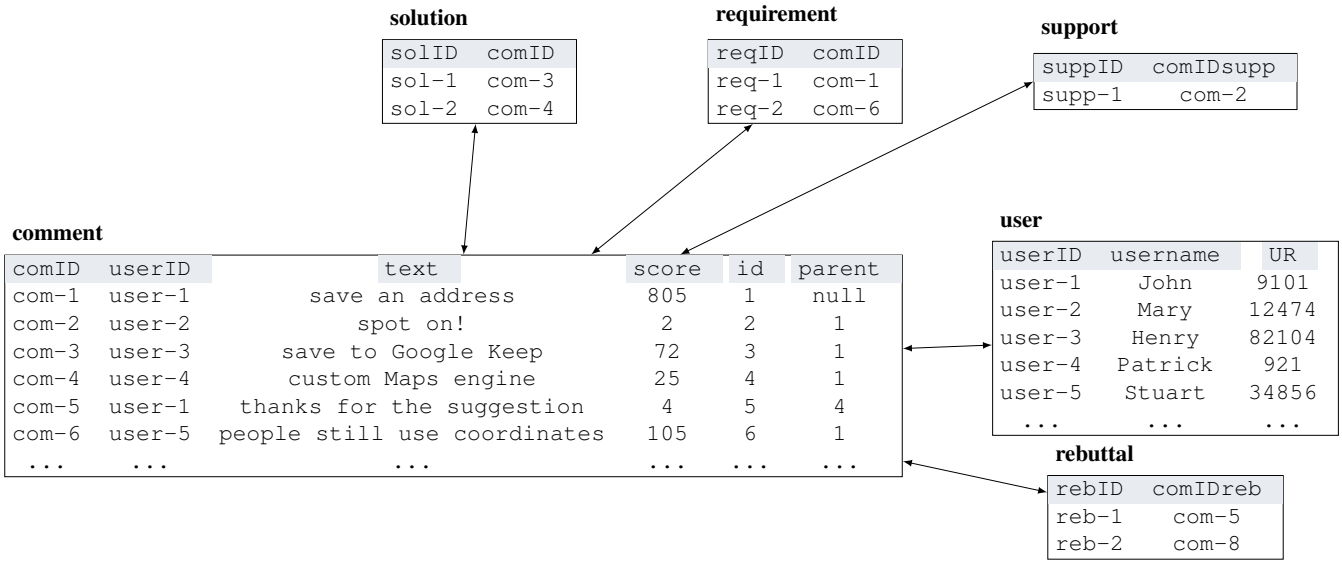



Fig. 3: Snapshot of data in the schema

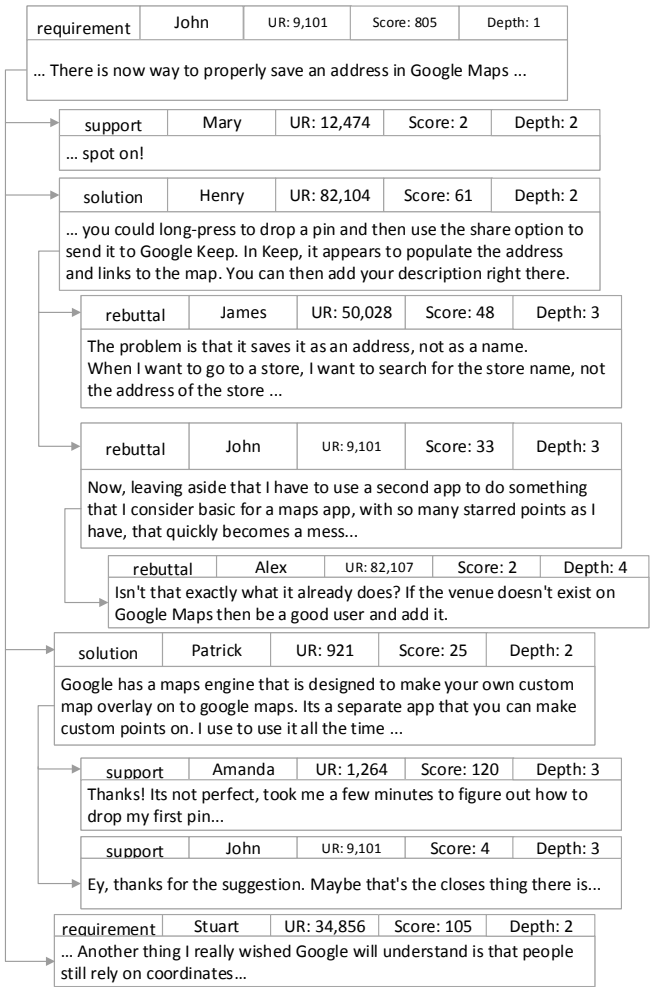



Fig. 2: Example discussion following information framework

from part of an SQL query be named. For this reason, we introduce a function new that generates a new name (a GUID) every time it is invoked.

SQL(comment). Gives the SQL query to return all comments along with information about the users who made them. As Listing 2 shows, it expresses a join of the user and comment relations.

Listing 2: SQL(comment)

```
select * from comment as query1 = new()

join user as query2 = new()
  on query1.iduser = query2.iduser;
```

SQL(propagate(x)). In order for propagation to work we need to create a hierarchical query in MySQL. Querying hierarchical data is a common problem in MySQL practice. Hierarchical data is data that has parent-child relationship. In the case of the comment data rows in our database schema we keep all hierarchical data in the same table. Namely the *id* and *parent* fields. Both fields must be incremental integer IDs. An important assumption for this algorithm to work is that id will always have a higher value than id. I.e. a parent must always become a record in the database before it's child. This fits our data source nicely, because that is always the case in terms of a comment-reply tree.

For hierarchical queries to work we need a helper function that traverses the parent-child tree and returns all valid children. In our case valid children means such that hold positive interaction.

Listing 3: SQL(propagate(x))

```
CREATE DEFINER=`root`@`localhost` FUNCTION
    `propagate`(input INT) RETURNS int(11)
  READS SQL DATA
BEGIN
  DECLARE _id INT;
  DECLARE _parent INT;
```

```
DECLARE _next INT;
DECLARE parent_sentiment INT;

DECLARE id INT;
DECLARE start_with INT;
DECLARE level INT;

DECLARE null_counter INT;
DECLARE max_level INT;


DECLARE curr_is_support INT;
DECLARE curr_is_requirement INT;
DECLARE curr_is_solution INT;
DECLARE curr_is_rebuttal INT;


DECLARE par_is_support INT;
DECLARE par_is_requirement INT;
DECLARE par_is_solution INT;
DECLARE par_is_rebuttal INT;

DECLARE prop_exist INT;

SET id = input;
SET start_with = input;
SET level = 0;
SET max_level = 5;

SET _parent = id;
SET _id = -1;
set null_counter = 0;

CREATE TEMPORARY TABLE if not exists
    propagation (
 idcomment VARCHAR(45) NOT NULL
 , id INT(11)
 , parent INT(11)
 , idrequirement INT(11)
 , idsolution INT(11)
 , idsupport INT(11)
 , idrebuttal INT(11)
 , sentiment INT(11)
 ) ENGINE=MEMORY;

SELECT COUNT(*)
INTO @prop_exist
FROM propagation
WHERE true;

if @prop_exist = 0 then

 INSERT INTO propagation
 (idcomment, id, parent, idrequirement,
     idsolution, idsupport, idrebuttal,
     sentiment)


 SELECT c2.idcomment, c2.id, c2.parent,
     requirement.idrequirement,
     solution.idsolution,
     support.idsupport,
     rebuttal.idrebuttal, 0
  from comment as c2
   left join requirement on c2.idcomment =
       requirement.idcomment
   left join solution on c2.idcomment =
       solution.idcomment
   left join support on c2.idcomment =
       support.idcomment
   left join rebuttal on c2.idcomment =
       rebuttal.idcomment
  order by c2.id asc;

end if;

CREATE TEMPORARY TABLE if not exists temp (
 tree_item INT(11)
 , parent INT(11)
 , level INT(11)
 , start_with INT(11)
 , sentiment INT(11)
) ENGINE=MEMORY;


UPDATE propagation set sentiment = 1 where
    id = start_with;

SELECT MIN(idsupport), MIN(idrebuttal),
   MIN(idsolution), MIN(idrequirement)
INTO par_is_support, par_is_rebuttal,
    par_is_solution, par_is_requirement
FROM propagation
WHERE id = start_with;


insert into temp (tree_item, parent, level,
    start_with, sentiment) values
    (start_with, null, 0, input, 1);

 LOOP

   SELECT MIN(propagation.id)
   INTO id
   FROM propagation
   WHERE propagation.parent = _parent
     AND propagation.id > _id;

   IF id IS NOT NULL OR _parent =
       start_with THEN

 SELECT MIN(sentiment)
 INTO parent_sentiment
 FROM propagation
 WHERE propagation.id = _parent;


 SELECT MIN(idsupport), MIN(idrebuttal),
    MIN(idsolution), MIN(idrequirement)
 INTO curr_is_support, curr_is_rebuttal,
     curr_is_solution, curr_is_requirement
 FROM propagation
 WHERE propagation.id = id;

     SET level = level + 1;
     if level > max_level then
   set max_level = level;
 end if;


 insert into temp (tree_item, parent,
     level, start_with, sentiment) values
     (id, _parent, level, input, 0);
```

```
    if parent_sentiment = 1 then

     IF curr_is_support is not null or
     (curr_is_requirement is not null  and
         par_is_requirement is not null) or
     ( curr_is_solution and
         par_is_requirement is not null) then
     UPDATE  propagation set sentiment = 1
         where propagation.id = id;
     UPDATE temp set sentiment = 1 where
         temp.tree_item = id;

     elseif curr_is_rebuttal is not null then
     UPDATE  propagation set sentiment = 2
         where propagation.id = id;
     UPDATE temp set sentiment = 2 where
         temp.tree_item = id;
     else
     UPDATE temp set sentiment = 0 where
         temp.tree_item = id;
   end if;

   ELSEIF parent_sentiment = 2 THEN

     IF curr_is_rebuttal is not null then
      UPDATE  propagation set sentiment = 1
          where propagation.id = id;
      UPDATE temp set sentiment = 1 where
          temp.tree_item = id;

     elseif curr_is_support is not null then
      UPDATE  propagation set sentiment = 2
          where propagation.id = id;
      UPDATE temp set sentiment = 2 where
          temp.tree_item = id;
      else
      UPDATE temp set sentiment = 0 where
          temp.tree_item = id;
     end if;

   end if;

   set null_counter = 0;
       SET _parent = id;
   SET _id = -1;
     END IF;


   IF id IS NULL THEN
     set null_counter = null_counter +1;

     SET level := level - 1;
     SELECT  comment.id, comment.parent
     INTO  _id, _parent
     FROM  comment
     WHERE id = _parent;
   END IF;

   if (null_counter > max_level) THEN
     return null;
   end if;

   END LOOP;
END
```

SQL(requirement). Gives the SQL query to return all the comments that are requirements. Notice that the requirement relation has a foreign key *comIDreq* that links it to the appropriate comment. Doing a join on requirement and SQL(propagation) would give us all comments annotated as a requirement and their propagated values, as shown in Listing 4.

Listing 4: SQL(requirement)

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT SQL ( propagate ) AS id
  FROM (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select idcomment, body,
  sum(Case When sentiment = 1 then score Else
      0 End) as pos_score,
  sum(Case When sentiment = 1 then 1 Else 0
      End) as pos_count,
  sum(Case When sentiment = 2 then score Else
      0 End) as neg_score,
  sum(Case When sentiment = 2 then 1 Else 0
      End) as neg_count,
  count(*) as all_count
    from temp

  join comment on temp.tree_item = comment.id
  group by start_with
```

SQL(x) where $\phi$. Gives the SQL query to return all $x$ that satisfy the condition $\phi$. In essence, $\phi$ acts as a selection filter, as shown in Listing 5.

Listing 5: SQL(x) where <condition>

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT SQL ( propagate ) AS id
  FROM (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    and comment.body regexp 'rely on
        coordinates'
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select idcomment, body,
  sum(Case When sentiment = 1 then score Else
      0 End) as pos_score,
  sum(Case When sentiment = 1 then 1 Else 0
      End) as pos_count,
```

```
  sum(Case When sentiment = 2 then score Else
      0 End) as neg_score,
  sum(Case When sentiment = 2 then 1 Else 0
      End) as neg_count,
  count(*) as all_count
    from temp

  join comment on temp.tree_item = comment.id
  group by start_with
```

SQL(solution(x)). Since solutions must be related to a requirement to make sense, we must link them to the requirements they satisfy. We do that by making a join between the solutions table and SQL(requirement), in this case represented as SQL(x). A solution must also be linked to its semantic meaning, as a SQL(propagation) record. Listing 6 is the SQL code for the definition.

Listing 6: SQL(solution(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT SQL ( propagate ) AS id
  FROM (
    SELECT comment.id from comment
    join solution on comment.idcomment =
        solution.idcomment
    where solution.idsolution is not null and
        comment.parent in
    (
      SELECT comment.id from comment
      join requirement on comment.idcomment =
          requirement.idcomment
      where requirement.idrequirement is not
          NULL
    )
    ) query0
WHERE   id IS NOT NULL
) ho;

select idcomment, body,
  sum(Case When sentiment = 1 then score Else
      0 End) as pos_score,
  sum(Case When sentiment = 1 then 1 Else 0
      End) as pos_count,
  sum(Case When sentiment = 2 then score Else
      0 End) as neg_score,
  sum(Case When sentiment = 2 then 1 Else 0
      End) as neg_count,
  count(*) as all_count
    from temp

  join comment on temp.tree_item = comment.id
  group by start_with
```

SQL(support(x)). Similarly, in order to find supporting arguments for an RE entity, we first join the support with the propagation and then join it with the RE entity itself (in this case represented as SQL(x), using a common foreign key. Listing 7 is the SQL code for the definition.

Listing 7: SQL(support(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT SQL ( propagate ) AS id
  FROM (
    SELECT comment.id from comment
    join support on comment.idcomment =
        support.idcomment
    where support.idsupport is not null and
        comment.parent in
    (
      SELECT comment.id from comment
      join requirement on comment.idcomment =
          requirement.idcomment
      where requirement.idrequirement is not
          NULL
    )
    ) query0
WHERE   id IS NOT NULL
) ho;

select idcomment, body,
  sum(Case When sentiment = 1 then score Else
      0 End) as pos_score,
  sum(Case When sentiment = 1 then 1 Else 0
      End) as pos_count,
  sum(Case When sentiment = 2 then score Else
      0 End) as neg_score,
  sum(Case When sentiment = 2 then 1 Else 0
      End) as neg_count,
  count(*) as all_count
    from temp

  join comment on temp.tree_item = comment.id
  group by start_with
```

SQL(rebuttal(x)). Finding rebuttal arguments for an RE entity is analogous: we join the rebuttal table with the propagation and then we join it with the RE entity itself (represented as SQL(x)), using a common foreign key. Listing 8 is the SQL code for the definition.

Listing 8: SQL(rebuttal(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT SQL ( propagate ) AS id
  FROM (
    SELECT comment.id from comment
    join rebuttal on comment.idcomment =
        rebuttal.idcomment
    where rebuttal.idrebuttal is not null and
        comment.parent in
    (
      SELECT comment.id from comment
      join requirement on comment.idcomment =
          requirement.idcomment
      where requirement.idrequirement is not
          NULL
    )
    ) query0
WHERE   id IS NOT NULL
) ho;

select idcomment, body,
  sum(Case When sentiment = 1 then score Else
      0 End) as pos_score,
```

```
  sum(Case When sentiment = 1 then 1 Else 0
      End) as pos_count,
  sum(Case When sentiment = 2 then score Else
      0 End) as neg_score,
  sum(Case When sentiment = 2 then 1 Else 0
      End) as neg_count,
  count(*) as all_count
    from temp

  join comment on temp.tree_item = comment.id
  group by start_with
```

Canary supports aggregators such as *discussed*, *popular*, *unpopular*, and *controversial* in order to allow a selection of requirements (analogously solutions) based on aggregate metrics.

SQL(discussed(x)). Let's take *discussed* requirements first. We define a requirement to be discussed if the sum of the number of replies (support and rebuttals) and their score is greater than some threshold (for simplicity, we give a vote the same weight as a reply.) Listing 9 is the SQL code for the definition. We find the number of supports and rebuttals by propagating the comment. The resulting relation has both counts of support and rebuttal comments as attributes, and we apply the threshold condition $\alpha$ so we get only those $x$ with values above threshold. Note that the value of $\alpha$ will be configured by the analyst. For the purposes of this paper, we set $\alpha = 200$.

Listing 9: SQL(discussed(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT  propagate(vars.id) AS id
  FROM    (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select * from
(
  (
  select *,
    case when neg_score=0 or neg_count = 0
        then ( pos_score * pos_count ) end as
        prop_score_pos,
    case when pos_score=0 or pos_count = 0
        then (neg_score * neg_count)  end as
        prop_score_neg ,
    case when neg_score>0 and neg_count > 0
        and pos_score > 0 and pos_count > 0
        then
    ( ( pos_score * pos_count ) / (neg_score
        * neg_count) ) end as prop_score_accum
  from (
    (select idcomment, body,
        sum(Case When sentiment = 1 then score
            Else 0 End) as pos_score,
        sum(Case When sentiment = 1 then 1 Else
            0 End) as pos_count,
        sum(Case When sentiment = 2 then score
            Else 0 End) as neg_score,
        sum(Case When sentiment = 2 then 1 Else
            0 End) as neg_count,
        count(*) as all_count
          from temp

        join comment on temp.tree_item =
            comment.id
        group by start_with
    ) as propagated
   )
  ) as query22
) where all_count > 200
```

The sets of *popular*, *unpopular*, and *controversial* records are all subsets of *discussed*. That is, they are all discussed entities, where either the positive sentiment dominates (popular), or the negative sentiment dominates (unpopular), or there is a balance between the two (controversial). We define positive sentiment as the sum of the number of support and upvotes and negative sentiment as the sum of the number of rebuttals and downvotes. In order to calculate them, we encapsulate SQL(discussed(x)) in a select statement and apply the appropriate selection filer to it.

SQL(popular(x)). Gives all discussed $x$ where the ratio of positive to negative sentiment is greater than $\beta$. Again, $\beta$ is configurable by the analyst. For this paper, we set it to 1.15, as shown in Listing 10.

Listing 10: SQL(popular(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT  propagate(vars.id) AS id
  FROM    (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select * from
(
  (
  select *,
    case when neg_score=0 or neg_count = 0
        then ( pos_score * pos_count ) end as
        prop_score_pos,
    case when pos_score=0 or pos_count = 0
        then (neg_score * neg_count)  end as
        prop_score_neg ,
    case when neg_score>0 and neg_count > 0
        and pos_score > 0 and pos_count > 0
        then
    ( ( pos_score * pos_count ) / (neg_score
        * neg_count) ) end as prop_score_accum
```

```
  from (
    (select idcomment, body,
      sum(Case When sentiment = 1 then score
          Else 0 End) as pos_score,
      sum(Case When sentiment = 1 then 1 Else
          0 End) as pos_count,
      sum(Case When sentiment = 2 then score
          Else 0 End) as neg_score,
      sum(Case When sentiment = 2 then 1 Else
          0 End) as neg_count,
      count(*) as all_count
        from temp

      join comment on temp.tree_item =
          comment.id
      group by start_with
    ) as propagated
   )
  ) as query22
) where prop_score_pos > 1.15 or
    prop_score_accum > 1.15
```

SQL(unpopular(x)). Gives all discussed $x$ where the ratio of positive to negative sentiment is less than $\theta$. Again, $\theta$ is configurable by the analyst. For this paper, we set it to 0.85, as shown in Listing 11.

Listing 11: SQL(unpopular(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT  propagate(vars.id) AS id
  FROM    (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select * from
(
  (
  select *,
    case when neg_score=0 or neg_count = 0
        then ( pos_score * pos_count ) end as
        prop_score_pos,
    case when pos_score=0 or pos_count = 0
        then (neg_score * neg_count)  end as
        prop_score_neg ,
    case when neg_score>0 and neg_count > 0
        and pos_score > 0 and pos_count > 0
        then
    ( ( pos_score * pos_count ) / (neg_score
        * neg_count) ) end as prop_score_accum
  from (
    (select idcomment, body,
      sum(Case When sentiment = 1 then score
          Else 0 End) as pos_score,
      sum(Case When sentiment = 1 then 1 Else
          0 End) as pos_count,
      sum(Case When sentiment = 2 then score
          Else 0 End) as neg_score,
      sum(Case When sentiment = 2 then 1 Else
          0 End) as neg_count,
      count(*) as all_count
        from temp

      join comment on temp.tree_item =
          comment.id
      group by start_with
    ) as propagated
   )
  ) as query22
) where prop_score_neg < 0.85 or
    prop_score_accum < 0.85
```

SQL(controversial(x)). Gives all discussed $x$ where the ratio of positive to negative sentiment lies between $\theta$ and $\beta$, as shown in Listing 12.

Listing 12: SQL(controversial(x))

```
drop table if exists temp;

SELECT  *
FROM    (
  SELECT  propagate(vars.id) AS id
  FROM    (
    SELECT comment.id from comment
    join requirement on comment.idcomment =
        requirement.idcomment
    where requirement.idrequirement is not
        NULL
    ) vars
  WHERE   id IS NOT NULL
  ) ho;

select * from
(
  (
  select *,
    case when neg_score=0 or neg_count = 0
        then ( pos_score * pos_count ) end as
        prop_score_pos,
    case when pos_score=0 or pos_count = 0
        then (neg_score * neg_count)  end as
        prop_score_neg ,
    case when neg_score>0 and neg_count > 0
        and pos_score > 0 and pos_count > 0
        then
    ( ( pos_score * pos_count ) / (neg_score
        * neg_count) ) end as prop_score_accum
  from (
    (select idcomment, body,
      sum(Case When sentiment = 1 then score
          Else 0 End) as pos_score,
      sum(Case When sentiment = 1 then 1 Else
          0 End) as pos_count,
      sum(Case When sentiment = 2 then score
          Else 0 End) as neg_score,
      sum(Case When sentiment = 2 then 1 Else
          0 End) as neg_count,
      count(*) as all_count
        from temp

      join comment on temp.tree_item =
          comment.id
      group by start_with
    ) as propagated
   )
```

```
  ) as query22
) where ( ( prop_score_neg > 0.85 or
    prop_score_accum > 0.85) and ((
    prop_score_pos < 1.15 or prop_score_accum
    < 1.15)) )
```

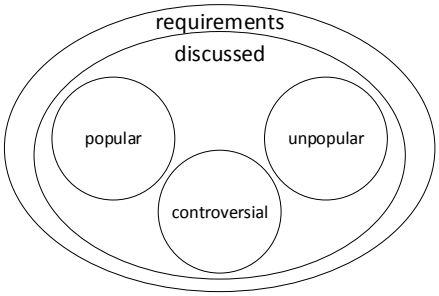



Fig. 4: Euler diagram showing the aggregator sets in Canary

Let $R$ be the set of all requirements. Let $D, P, U, T$ be the sets of discussed, popular, unpopular, and controversial requirements. Then $D \subseteq R$. Also, $P \subseteq D$, $U \subseteq D$, and $T \subseteq D$. The sets of $P, U$, and $T$ have no common members by our definitions above. This relation can be seen in Figure 4. An analogous diagram can be made for solutions.

Popularity is measured using ratios. The algorithm used in the current Canary algorithm is rather naive, but future improvements on the language can utilize a more sophisticated algorithm. The calculation is done but summing quantitative data about positive interaction and dividing it by the sum of negative interaction.

$$\text{Popularity Score} = \frac{\sum_{i=0}^{n} \text{PV(i)} \times \text{n}}{\sum_{i=0}^{m} \text{NV(i)} \times \text{m}}$$

Where $n$ is the number of children with positive semantics and PV(i) the number of votes for a given child $i$, and $m$ is the number of children with negative semantics and NV(i) the number of votes for a given child $i$. In Table I we presents the assumptions we can make based on the ratio.

| pop ratio | aggregator | assumption |
|---|---|---|
| <1 | unpopular | negative interaction has prevalence |
| 1 | controversial | balance between positive and negative |
| >1 | popular | positive interaction has prevalence |

Table I: Aggregator assumptions