Analysing User Feedback and Finding Experts: Can Goal-Orientation Help?

Matthieu Vergne, Itzel Morales-Ramirez, Mirko Morandini, Angelo Susi, and Anna Perini

Fondazione Bruno Kessler
Software Engineering Research Unit
Via Sommarive 18, 38123 Trento, Italy

Abstract. Goal-oriented approaches in requirements engineering aim at understanding stakeholders' needs, modelling the intentions, dependencies and expectation to be met by a system-to-be, or by a new release of an existing system in a software evolution process. In the context of software evolution, we consider user feedback, as commonly available in user forums and bug-trackers, as the information artefact that impacts specific requirements and design of the software. In this position paper, we argue about the advantages that goal-orientation can bring in this context when addressing the following issues: i) the analysis of user feedback, usually expressed as free or semi-structured text; and ii) the identification of the most expert users that can contribute to requirements evolution. We define the problems, state the emerging research questions and present contributions with the help of an illustrative example.

1 Introduction

Goal-oriented (GO) approaches in requirements engineering (RE) aim at understanding stakeholders' needs, by modelling the intentions, dependencies and expectations to be met by a system-to-be or by a new release of an existing system. They are widely adopted in requirements elicitation activities based on face-to-face meetings with stakeholders (e.g. users and analysts), e.g. in [1], supporting the communication while acquiring and refining requirements. After releasing a software application, the activities of acquiring and analysing requirements for the purpose of system maintenance and evolution remain still important, especially as long as users provide their feedback to the analysts team, which is the case in many open-source projects. In the context of our research, we refer to user feedback as an information artefact communicated as free or semi-structured text, that expresses the users' perspective upon experiencing the use of a software application. This feedback is usually analysed against the system's software architecture and code for the purpose of bug fixing and general system maintenance. Yet, feedback can also be used for deriving the requirements of next versions [2].

Currently, as far as we know, there is no link between this feedback and a requirements model that supports a classification of feedback and the organisation of it. We are investigating on user feedback from a RE perspective, with the aim of defining a structured approach for managing it. We believe that requirements models can help analysts to structure user feedback and to identify the most expert users (among the forum's participants) who could be engaged for clarifying or evaluating new requirements.

The main issues to be faced towards this objective are the separation of relevant and noisy feedback and the definition of criteria for ranking forum participants against expertise levels. We identified various specific challenges in analysing feedback, including: i) coping with the heterogeneous abstraction levels in which it can be expressed; ii) identifying the key concepts in feedback that can possibly be linked to an existing requirements specification; and iii) evaluating its impact on system requirements [3]. Concerning expert finding, challenges are: iv) identifying relevant pieces of knowledge and their relationships for the purpose of modelling the expertises of the users; and v) building a consistent model that we can query to infer the most expert people in the group under consideration.

In this paper, we focus on the challenges *ii)* of feedback analysis and *v)* of expert finding, discussing about advantages given by adopting a GO approach. We make the strong assumption that, for a given software application, a GO requirements specification is kept aligned along the phases of system maintenance and evolution.

We envisage a process in which the feedback is analysed and correlated to the key concepts in a project's requirements model. In a second step, the links obtained from this analysis can be exploited, among other data, to identify expert users that can contribute effectively to the requirements analysis.

To approach the first challenge we propose a meta-model that defines the underlying structure of user feedback, and relate it to the conceptual entities of the GO modelling language Tropos [4], while for the second challenge we exploit Markov networks for processing the information available in forums and goal models in a probabilistic way, which will support making inferences about users' expertise.

In Section 2, we state the research objectives and describe an illustrating scenario. Section 3 presents the contributions along the two lines of feedback analysis and expert finding, while Section 4 concludes and outlines future work.

2 Research objectives and motivating example

2.1 Objectives

We consider the analysis of user feedback may be guided by a meta-model, thus identifying its purpose and impact on a goal model. On the other hand, the combination of this feedback with the goal model may lead to identify the potential experts on relevant topics, supporting a better requirements refinement process. Consequently, we want to discuss:
– how we can benefit from a *user feedback meta-model*, centred around the concept of *purpose* and a set of *bridging links* to the Tropos meta-model, to understand to what extent the user feedback impacts on a requirements specification;
– how probabilistic models such as *Markov networks* can be used to exploit the information provided in a GO approach in order to identify expert users participating in forum discussions, and on which we can rely for the improvement of the system.

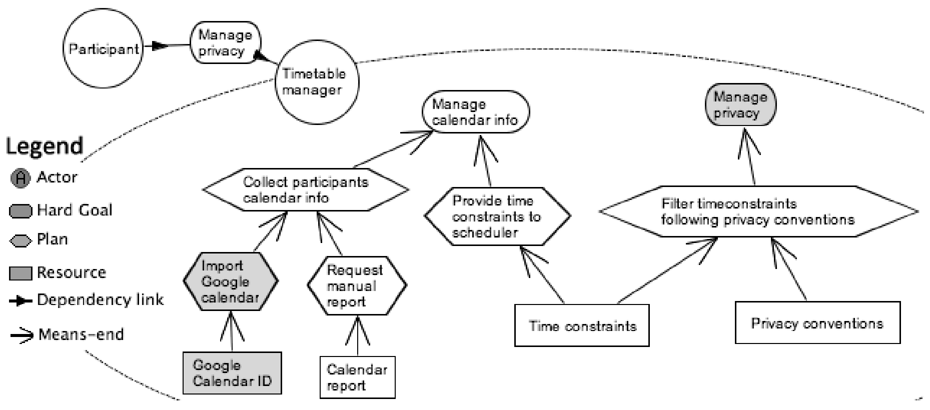

Fig. 1. Internal goals of the *Timetable manager* actor (Tropos notation).

2.2 Motivating example

We envision a software application, called OPEN-MEET, that schedules internal meetings in research groups. This application has been developed adopting the Tropos GO approach for domain analysis and requirements modelling. An excerpt of the model is depicted in Figure 1, showing the functionalities performed by the actor "Timetable manager" (a sub-component of OPEN-MEET). The system schedules meetings by considering the personal calendar of all the people in the group (the participants), as well as the availability of meeting resources. To improve the app, its users can provide feedback in a forum (bug reports, suggestions, wishes, etc.). To do so, they write their concerns, in such a forum, by giving a short title and a description. Examples of user feedback are:
-Stefania "It would be nice to have an option to specify certain days of the week on which I'm fully available" (Concerning calendar information that users should be able to provide).
-Paolo "I do not want my full Google calendar to be considered, only the periods related to my working time" (Concerning privacy issues).

3 Scientific contributions

3.1 Analysis of User Feedback

We want to analyse the user feedback and understand whether it impacts on requirements expressed in a goal model. In a previous work [3] we reviewed the literature on feedback and derived a conceptualisation to describe feedback's rationale and structure. We extend this conceptualisation in terms of a meta-model with *bridging links* to the Tropos meta-model, as described in [4]. Excerpts of both meta-models are shown in Figure 2, left side depicts the conceptual entities that drive the analysis of feedback, right side shows Tropos excerpt. User feedback is elaborated in natural language and typically based on some *topics*[1]. From our perspective, user feedback can have one or more *purposes*. We present only three purposes: (1)Improvement, (2)Clarification and (3)StrategicBehaviour. Number 1 refers to users' wish of using a better app, 2 for requesting further details (both ways), and 3 refers to the propagation of the impact through system's functional and non-functional requirements that can receive an action, for instance modification. The *bridging links* are high-level expressions of concepts contained in feedback. For example, FunctionImprovement may refer to a specific behaviour of the app, e.g. "print calendar". QualityImprovement may refer to a judgement about the operation of the app, e.g. "calendar interface". ExecutionImprovement may refer to the steps for performing a function, e.g. "first select day". Therefore, we see that the *bridging links* are linked to a HardGoal, SoftGoal, Plan, respectively. Finally, in the feedback excerpt "It would be nice... an option", it may refer to a new functionality because of the expression highlights a wish, i.e. a *goal*. The entity OwnedBehaviour is connected to the entity Decomposition to indicate that an Action might be applied to an element of Tropos by following the means-end links, i.e. to propagate a likely impact.

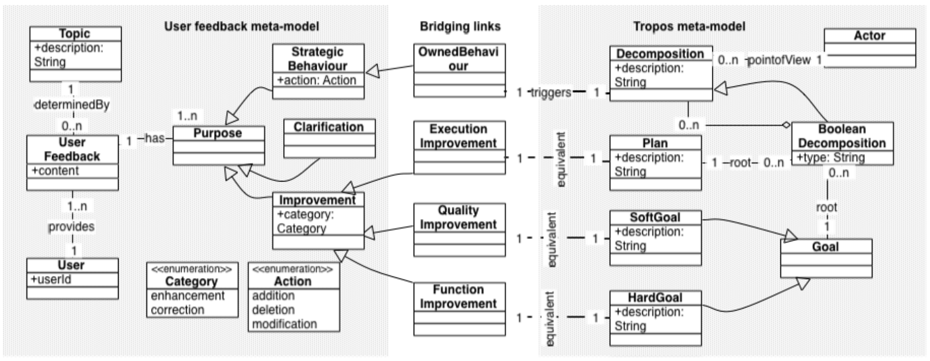

Fig. 2. Excerpt of the user feedback meta-model with bridging links to the Tropos meta-model.

Recalling the running scenario, Paolo's feedback could be classified under the *topic* "Manage privacy", which turns out to be a *goal* (see Figure 1). The meta-model may help in guiding the identification of the *purpose(s)* contained in such a feedback, thus leading to the impacted fragment in the model. In our example, the purpose is a written expression recognised by the starting words "I don't want". This expresses a wish of avoiding something. The key concepts "full Google calendar" and "to be considered" point out that there is a request to avoid considering the whole calendar. As Paolo's concern may be a correction, the entity OwnedBehaviour triggers first the propagation with the link modification to a functionality in the model that contains the text *calendar*, i.e. the goal "Manage calendar info". By following the means-end links, the analyst can deduce that the feedback "...full Google calendar..." may refer to modify a *plan*. So the impacted fragment to be analysed is the plan related to the goal "Manage calendar info". Since the feedback was provided under the topic "Manage privacy", it needs to be

[1] The elaboration of the *topic* based on system's behaviour is an issue under research.

classified as Clarification and discuss it with the users to validate requirements change. But at this point the analyst will be able to identify likely impacted fragments (elements shown in Figure 1[2] in grey).

3.2 GO-based Expert Finding

By investigating expert finding techniques in RE[3], we have identified several types of relationships to exploit, as for instance the concepts related to a broad topic or the topics a user knows about. In this section, we show that the goal model and the analysis of the feedback can provide such relationships, although there is uncertainty issues, in order to identify *expert users* for the discussion process. Similarly to [5], we build a *Markov network*, able to exploit these relations and manage uncertainty, with the information provided by the goal model and the previous feedback (in the forum), relating the users to different *pieces of knowledge* (e.g. concepts and topics). The analysis of the current user feedback is then used to identify the pieces of knowledge to consider, to query the network to infer the probability that a given user suits to the discussion, and finally to rank the users regarding these probabilities.

Concretely, we extract pieces of knowledge from the goal model, which are *concepts* (in the labels of the goals, plans and so on), *topics* (concepts in not-leaf elements like the goal "Manage privacy") and abstractions of the users (i.e. actors) that we will call *roles*. We can also infer their *relationships* through this goal model: for instance, as a goal decomposition relates a parent goal to its sub-goals, it also relates the topics of the former to the concepts of the latter. Then, we exploit the previous feedback contained in the forum, identifying further relations between topics and concepts, but also the *users* who have been involved and their relations to the corresponding topics and concepts (we can use other sources to identify user-to-role relations).

Consequently, we have several relevant pieces of knowledge, namely users, roles, topics and concepts, and several ways to relate them: each user can play some roles, know about some topics and use some concepts; playing a specific role can lead to know about some topics and concepts; knowing about a specific topic can lead to know some concepts. Thus, considering that each one corresponds to a *node* in a graph, we can then relate all the nodes of one kind to all the nodes of another kind, making the graph almost fully connected (only nodes of the same kind are not related, e.g. a user is not related to another user). The amount of information we have for each relation can represents its *strength*, like in a weighted graph, and more this strength is high more the data is reliable. For example, if a lot of feedback on the topic "Manage privacy" mentions "Google calendar" but a few mentions "lunch", the link between the former will be stronger than for the latter. Finally, the strong paths linking the users to the different pieces of knowledge indicate which users are better to involve, considering a set of topics or concepts we are looking for.

We model this kind of graph via Markov networks, where the nodes are *random variables* and the relations correspond to *potential functions*. In our case, the random variables are binary, identifying whether a user/role/topic/concept is concerned

[2] This is a late requirements model that represents the results on the elaboration of feedback.

[3] *Infer Informational Capabilities by Relating Expertises in Requirements Engineering*. Technical report. Matthieu Vergne, 2013. Document available on request.

by the discussion process or not, and the potential functions compute the strength of the edges depending on the state of the related nodes. With such a model, we can infer the *probability* (which is the normalised product of the potential functions) for a specific user to be of interest given some identified pieces of knowledge, formalised as $P(user|roles, topics, concepts)$. Computing this probability for all the users, we can rank them to identify who are the best ones to recommend for the discussion. For instance, if the analyst identifies with the feedback the topic "Manage privacy" and the concept "Google calendar", he computes $P(x_i|\text{Manage privacy}, \text{Google calendar})$, where x_i corresponds to Stefania, Paolo or any other user, and finds the most suitable users by looking at the highest probabilities.

4 Conclusion and future work

In this paper, we provided a preliminary discussion about the advantages of linking a meta-model of user feedback with the conceptual entities of the Tropos meta-model. We also discussed how a GO approach can be exploited to model the relevant knowledge in a Markov network, and how it relates to the available users to infer the most probable experts.

So far, the proposed meta-model supports a more focused analysis of feedback, by revealing a hidden structure that can be exploited to identify the fragments of a requirements model which are affected by the feedback. The expert finding, on the other hand, supports the understanding of how this model should be impacted, using the feedback analysis to identify expert users to discuss with. However, the analysis is performed manually and the use of the meta-model is merely focused on giving a structure to user feedback, while the Markov network still need to be parametrised and assessed with some quality measures.

As future work, we plan to support a semi-automatic identification of feedback purpose, by classifying it with the help of feedback patterns combined with a supervised learning process, and try to identify parameters that provide a good compromise between ranking quality and computation time.

References

1. Morales-Ramirez, I., Vergne, M., Morandini, M., Sabatucci, L., Perini, A., Susi, A.: Where did the requirements come from? a retrospective case study. In: ER Workshops. Volume 7518 of LNCS., Springer (2012) 185–194
2. Schneider, K.: Focusing spontaneous feedback to support system evolution. In: RE, IEEE (2011) 165–174
3. Morales-Ramirez, I.: On exploiting end-user feedback in requirements engineering. Doctoral Symposium at REFSQ13. ISSN 1860-2770. (April 2013)
4. Giorgini, P., Mylopoulos, J., Perini, A., Susi, A.: The Tropos Methodology and Software Development Environment. In Yu, Giorgini, Maiden, Mylopoulos, eds.: Social Modeling for Requirements Engineering, MIT Press (2010) 405–423
5. Hu, D.H., Yang, Q.: Cigar: Concurrent and interleaving goal and activity recognition. In Fox, D., Gomes, C.P., eds.: AAAI, AAAI Press (2008) 1363–1368