Modelling Users Feedback in Crowd-Based Requirements Engineering: An Empirical Study

Nada Sherief[1], Walid Abdelmoez[2], Keith Phalp[1], and Raian Ali[1](✉)

[1] Faculty of Science and Technology, Bournemouth University, Poole, UK
{nsherief,kphalp,rali}@bournemouth.ac.uk
[2] The Arab Academy for Science, Technology and Maritime Transport, Alexandria, Egypt
walid.abdelmoez@aast.edu

Abstract. Most enterprises operate within a complex and ever-changing context. To ensure that requirements keep pace with changing context, users' feedback is advocated to ensure that the requirements knowledge is refreshed and reflects the degree to which the system meets its design objectives. The traditional approach to users' feedback, which is based on data mining and text analysis, is often limited, partly due to the ad-hoc nature of users' feedback and, also, the methods used to acquire it. To maximize the expressiveness of users' feedback and still be able to efficiently analyse it, we propose that feedback acquisition should be designed with that goal in mind. This paper contributes to that aim by presenting an empirical study that investigates users' perspectives on feedback constituents and how they could be structured. This will provide a baseline for modelling and customizing feedback for enterprise systems in order to maintain and evolve their requirements.

Keywords: Users' feedback · Feedback analysis · User involvement · Crowd-Based requirements engineering · Enterprise requirements evaluation

1 Introduction

Requirements management is still one of the most challenging fields in software development [1], has the most impact on project success, and is a major issue for decision makers in enterprises. Requirements are gathered from, yet must still represent, a diverse group of users; they are intrinsically volatile in nature. These issues are exacerbated by the problem that users still typically provide their feedback on the fulfilment of their requirements in a natural language and in an ad-hoc manner, which introduces a great deal of imprecision and ambiguity.

To cope with such a lack of precision, a range of semi-automated techniques have been suggested to handle such user data (this includes techniques such as text mining and/or human facilitator). These techniques may be used to gather, interpret, aggregate, and revise what users say, partly to mitigate for such issues as bias and subjectivity in their textual responses. More effective results can be reached if the feedback is written in a structured format. Structured feedback text would, arguably, allow approaches, such as text processing, to provide more accurate results within less time and with fewer

J. Ralyté et al. (Eds.): PoEM 2015, LNBIP 235, pp. 174–190, 2015.
DOI: 10.1007/978-3-319-25897-3_12

human interventions. If text is structured the requirements extraction process can be more systematic, eliminating complexity and ambiguity found in natural language, and requiring less effort.

Research has identified the need to involve users in requirements engineering, but often focuses on a small number of selected users to give input and feedback in requirements related activities and afterwards in user acceptance testing [2]. Recent research has been focusing on the possibility of utilizing crowdsourcing in requirements engineering [3, 4] to cater for the dynamic contexts and the diversity of users. Moreover, in [5, 6] the collective users' feedback was also encouraged for shaping software adaptation as users are important to communicate certain information that cannot be monitored and captured by automated means and also cannot be fully specified by designers at design time, yet are necessary to plan and support adaptation. Furthermore, authors in [7] stated that the crowd can enrich and keep the precision of engineers' knowledge about software evaluation via their iterative feedback at runtime (i.e. while the software is in use). That is, users' feedback can communicate their opinion on the role of the system in meeting their requirements leading to better users' acceptance of the software. Their acceptance of the product is of a high importance for market success.

However, the literature is still limited in providing engineering approaches to developing systematic feedback acquisition [8, 9]. Our research focuses on the development of a modelling and elicitation framework of crowdsourced feedback at runtime. This includes devising mechanisms to structure such feedback in a way that makes it easy for users to express and engineers to interpret. This will allow the system to prioritize different problems reported by users. Also, it will help in evaluating the overall quality of the system and in taking evolution and maintenance decisions.

In this paper we conduct a two phase empirical study. We follow a qualitative method of two phases including two focus groups in the first phase and three forums' analysis in the second. In the first, we build on the top of our initial findings on the topic in [7] and provide more detailed results on the different aspects of the feedback design and conduct of runtime feedback acquisition. In the second phase study, we undergo a detailed analysis of users' feedback on enterprise software applications by analysing actual users' feedback through examination of their posts and responses on three online forums. We finally discuss how the results inform the process of designing feedback acquisition and increase its efficiency.

2 Research Method

We followed a qualitative approach to explore and understand how users provide feedback and their preferences on the acquisition process. The study had two phases. In the first phase study we took an empirical approach by conducting a two sessions focus group study, which is a popular technique of qualitative research in software engineering [10]. The sessions lasted 2 h and 52 min. Both sessions were audio recorded and transcribed with consent from participants. Our goal was to collect insights and experience from users who have actually given feedback before. Also, both junior and senior software engineers were invited to understand how more high-tech users give feedback and how they think a good feedback should be structured in order to be easily understandable and analysed. The main areas we wanted to explore were:

(RQ1). How users would like feedback to look like, and the criteria that judge whether the feedback is meaningful and useful?
(RQ2). How users would like to be involved in the process of providing feedback, and what encourages them to act as evaluators?

The focus groups were analysed using the thematic mapping approach [11]. The results of the focus groups analysis shown in Fig. 1 gave us a good level of understanding of users' feedback aspects. The resulted thematic areas can be viewed from two different perspectives. In the first perspective, participants gave several insights regarding the structure of the feedback and what are the characteristics they think make their feedback meaningful and useful. These ideas are covered in the environmental and structure thematic areas. In the second perspective, participants gave their perceptions regarding what they expect from a feedback acquisition method. How it can support, motivate and value their feedback. These ideas are covered in the engagement and involvement thematic areas.

Our research goal necessitates building a more concrete description for feedback structures. So in order to get the elaborated view, we conducted another study which involved the analysis of three actual online forums where people give feedback on business software. The main areas we wanted to explore in the second phase study (i.e. the three forums analysis) were:

(RQ1). What are the main concepts that constitute the feedback structure?
(RQ2). What are the designs of the identified feedback concepts?

To start with, we have taken both the environment and structure thematic areas shown in Fig. 1 of the focus groups results as our initial template. This template was edited and enhanced in the forums analysis process to come up with more details on how feedback could be structured in a meaningful and useful manner. We have analysed 200 feedback from 20 different sources found on Microsoft's TechNet, WordPress, and SAP forums. We targeted business software to avoid the noise typically found in general purpose software, as normally users tend to give a more serious and focused feedback, because of the social norms in such kind of forums. Also, business users are best fitted from the motivation perspective, because it has a direct value on their work and performance. We have chosen these three forums in order to target different types of business users with diverse technical capabilities.

We studied actual users' feedback through observation and analysis of their posts and responses on forums. The main advantage of this method is the direct examination of the experience of user's difficulties in the task of expressing their problems and opinions in using the software, the task flow and challenges. Moreover, forums provide a considerable amount of feedback that we have analysed using thematic analysis [11] with the intention to come up with the main concepts that constitute a feedback, and the outlines of the identified concepts.

Using software in the data analysis process has been believed to increase consistency and/or accuracy of qualitative research. In our research, NVivo 10 was used

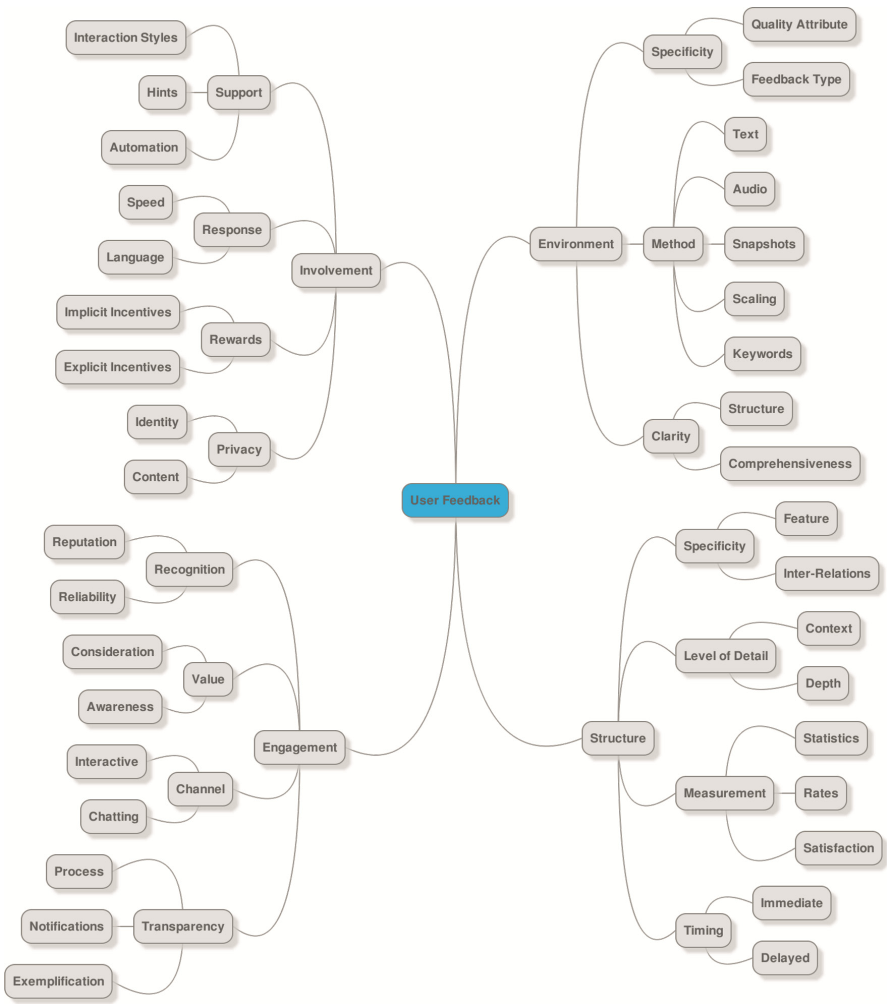

Fig. 1. Focus groups final thematic map

in the data collection and analysis. Moreover, we used multi-coder arrangement [12] to reduce subjectivity and bias. Two researchers performed the coding of the same collection of sources, and analysis of forums. During the analysis new concepts and structures emerged and caused new themes, concepts, categories or codes to be added to the thematic map. After each team discussion, the members refined, merged and/or reorganized the nodes. When a disagreement emerged a third researcher was consulted. This helps validate that a theme is not just emerging from a single coder subjective thinking.

3 Focus Groups Results

Following the recommendation of six stages of analysis [11], four thematic areas were formed, and 15 themes were identified from the analysis, which are shown in Fig. 1. The four thematic areas are: environment, structure, engagement and involvement.

Environment refers to the settings that support users so they feel confident in providing meaningful evaluation feedback. This includes Specificity, Clarity and feedback Method. In detail, participants would like to use a method they prefer to aid them in easily providing feedback. Furthermore, to improve the clarity of feedback, participants pointed out that it is preferable to add reasons and explanations in feedback to help make their viewpoints more **comprehensive**. Also, providing **structure** to the feedback will decrease misinterpretations and ease the analysis of texts afterwards. Specificity can be goal-oriented, which means by specifying the **quality attribute** in the feedback that concerns the user, such as usability, or reliability. Also, specificity can be influenced by the **feedback type** the user would like to provide, as more users tend to give feedback when they need help or when a problem occurs.

Structure refers to the attributes of a feedback which are favourable to be seen by the participants. This includes Specificity, Level of Detail, Measurement, and Timing. In detail, they thought that feedback would be more useful and accurate if it was related to a certain **feature**. It would be useful to be able to correlate feedback according to the **inter-relationships** between the features, because some features may affect the functionality of others. Moreover, it is important to provide the possibility of varying **details** in the feedback to ensure a minimum level of meaningful and useful information, and also to put into consideration other **contextual** aspects that might affect the users while giving their feedback. Furthermore, participants also suggested using simple measurements in a way to aid users in giving their feedback through and re-using the experiences of others. For example, users can **rate** how much others' feedback was meaningful or useful, and accordingly **statistics** can appear to users to show other useful feedback. Also, users can give feedback about their experience with new changes in the software to aid engineers in measuring user **satisfaction**. Finally, it is also important to consider the timing of giving the feedback. Users thought that giving a feedback **immediately** (i.e. at runtime) is important especially in reporting errors or problems, as it helps giving more accurate feedback with detailed explanations, and therefore would affect the structure of the feedback.

Engagement refers to the key merits the acquisition process provides to the involved users that encourage them to take part as evaluators. This includes some key characteristics of engaged users with the process, and also the qualities that are important to the process. This includes Recognition, Value, Channel, and Transparency. In details, participants mentioned that they would like to be recognized through their **reputation**. Reputation may be considered as a component of identity as defined by others. Reputation is a vital factor in any community where trust is important. Also, users would take recommendations, and/or solutions into consideration if they are given from **reliable** users. The reliability of users increases the weight of their feedback. Moreover, users like to be valued in a way in the participation. Participants mentioned that their feedback is valued by knowing that it taken into **consideration** for further analysis and leads to software enhancements. Also, the possibility to learn from others' experiences provides great value to users as it increases their **awareness** by knowing other possible features variations they were no aware about before. Furthermore, Channel reflects the way users want to interact through feedback. They would like the feedback acquisition process to be simple and **interactive**. Also, after giving their feedback they would appreciate if they can **chat** with the analyst to discuss their feedback. Finally, it would increase users' trust if they know the **process** in which their feedback will be handled and considered. Transparency generally implies openness, which can be achieved in different ways. The user can be **notified** through a message that the feedback will be taken into consideration. Transparency may be achieved by giving the user an **example** of other users whom their feedback was taken into consideration and their issue was resolved.

Involvement refers to a variety of aspects that motivate users to participate in the process of feedback acquisition and can directly influence the decisions and activities in using/evaluating the software. This includes Privacy, Rewards, Support, and Response. In detail, privacy issues were raised by participants. Participants differentiated between two aspects in privacy, the privacy of their **identity**, and the privacy of the **content** they provide (i.e. their feedback). Moreover, participants were particularly interested in the rewards mechanism for involvement whether through implicit or explicit incentives. **Implicit incentives** are not based on anything tangible. Social incentives are the most common form of implicit incentives. These incentives allow the user to feel good as an active member of the community for example through increasing their reputation. **Explicit incentives** refer to tangible rewards, for examples financial. Furthermore, the level of support from the feedback system was considered important. Many suggestions were raised about how a feedback acquisition tool can help them. For example, the **interaction styles** *"there can be videos to explain to the users what they can do (in order to provide feedback)"*. The ease of use of the feedback acquisition tool is important. They also suggested that the feedback tool can provide **hints** to the users about its capabilities. Moreover, if there is an **automated** detection in some steps of providing the feedback, this would further ease their job. For example if the tool can automatically detect the feature the user is having trouble with. Finally, the feedback tool response on feedback was also considered important. Two characteristics of system response were discussed, which are the **speed of response** from the system and the **language of response**.

4 Forums Analysis Results

In this section we explain the forums analysis results represented in the final thematic map shown in Fig. 2. The **first** thematic area that was founded from the analysis to the forums is our **novel classification of feedback types** that users provide. We have reached 8 distinct feedback types that users use on forums. In this section we will provide definitions for each type of feedback.

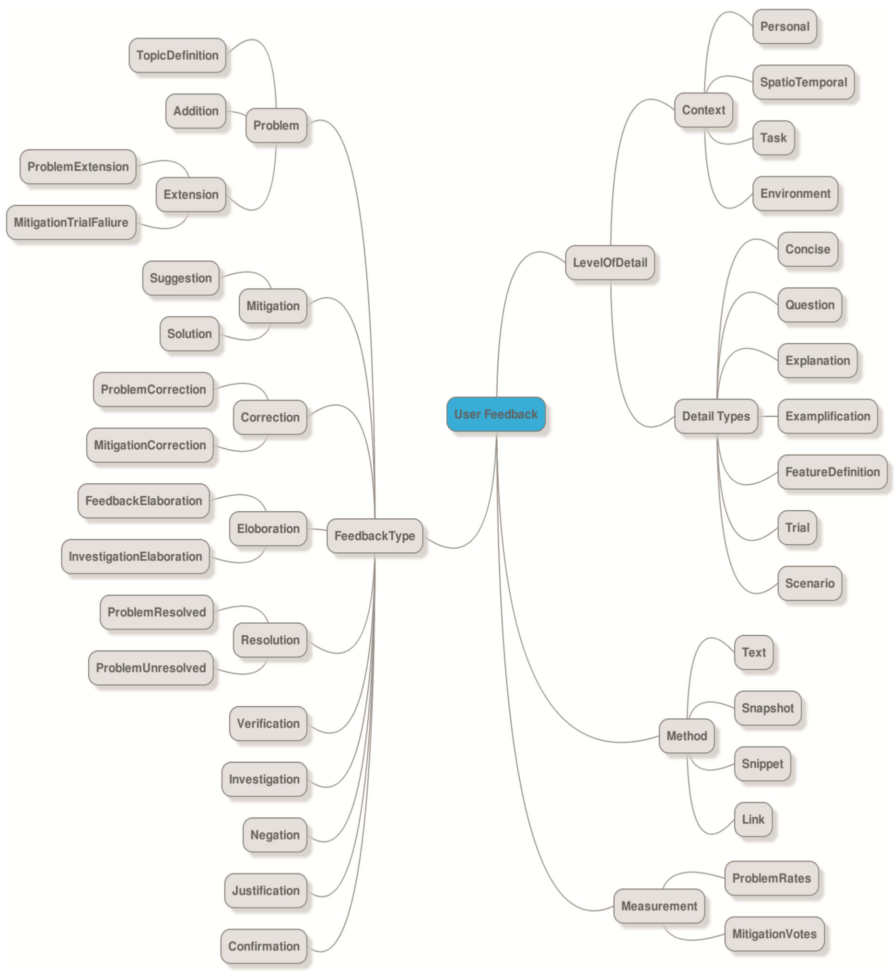

Fig. 2. Forums analysis final thematic map

Before we start defining the meaning of each feedback type, we would like to classify feedback into two types: a ***simple feedback***, and a ***complex feedback***. A simple feedback is a feedback that consists of a single feedback type that a user provides in his post to express a certain meaning, while the complex feedback is a structured feedback that consists of several feedback types that together form a new meaning that can be inferred from its unique structure.

Below if the list of feedback types and subtypes (i.e. cases):

1. **Confirmation or Negation** is a simple feedback type that the users use to agree or disagree on problems or opinions of other users. When these feedback types are unaccompanied with other types in a feedback, it can be inferred as voting for a problem or a given solution.

2. **Investigation** is a simple feedback type used when a user is asking a question to clarify something about another feedback posted by another user. A user may ask about some issues in a problem statement, or unclear steps in a provided solution, or clarify some contextual information that helps explain the problem more.
3. **Elaboration** is a simple feedback type where the user gives extra explanation on a feedback he already posted. There are two cases for giving extra explanations on a feedback:
   a. **Feedback Elaboration** is when a user needs to give more detailed information that he forgot to provide in his main feedback this can be added separately in the feedback where he elaborates. For Example, A user can elaborate on a problem he provided by giving explanation on some trials that he made trying to solve his problem or rephrasing the problem statement.
   b. **Investigation Elaboration** is when a user simply replies on an Investigation by giving detailed explanations to answer the posted question(s).
4. **Justification** is a simple feedback type used when users need to provide reasons to support their feedback. They may give reasons why they provided a solution/ suggestion, or it can be used with confirmations or negations to state reasons why a user agrees or disagrees on a feedback opinion of another user.
5. **Verification** is a complex feedback type where a user gives his opinion on a solution or suggestion he received on the problem that he posted. As a complex type it means that it combines several other feedback types in its structure that are mandatory in its definition. Specifically in order to verify whether a solution or a suggestion was useful or not, this feedback has to reference a certain Mitigation (i.e. Solution or Suggestion) in which the user will be giving his opinion to verify whether it solved the issue or not by using Confirmation or Negation.
6. **Problem** feedback type refers to a certain feature or group of features in the software that the user is having problem with, and a detailed explanation of the problem. Problems may use other feedback types such as Investigations to ask users some questions they need answers for. However, problems in general cannot occur in the same Feedback post with Mitigations or Verifications. In general users who post problems are not the same users who post the Mitigations, and even if this case occurred will not be contained in the same problem post.
   a. **Topic definition** is a simple feedback type that represents the first posted problem in a feedback thread where the user is seeking help. Therefore it does not reference any other feedback in the thread but can be referenced in many other posts.
   b. **Addition** is a complex feedback type where a user votes (i.e. agrees or disagrees) on any posted problem, and adds another problem in his feedback, which is not related to the main problem on which the discussion is held. This means that a feedback thread may contain multiple problems along with the replies. From the definition of this feedback type as a complex type, this implies that it must contain other feedback types in its definition, which in this case are Confirmation or Negations that must reference another problem. Therefore, it cannot reference a feedback post that contains Mitigation, because by definition we use this feedback to add a problem to a problem.

7. **Mitigation** is a complex feedback type that represents a solution or a suggestion that may help a user resolve the problem(s) he has. Since this type is intended to resolve a problem, therefore it has to reference that problem in the solution or suggestion for specificity. Also, for every Mitigation it is always expected that the user who posted the problem will Verify that Mitigation. There are two types of Mitigations:
   a. **Solution** is a well-known procedure or steps that when followed can resolve the problem or issue.
   b. **Suggestion** is a recommendation that a user provides for another user as a trial to resolve his problem. This suggestion may or may not solve the problem. This needs Verification from the problem owner (i.e. the user who posted the problem).
8. **Correction** is used when a user corrects the understanding of another user. There are two cases for this feedback type.
   a. **Problem correction** is a complex feedback type. It occurs when the user corrects the problem of another user. In a problem definition a user must refer to a feature(s) that he is having a problem with. Sometimes the user is using a feature which is not intended for the type of task he is doing, simply due to a lack of understanding of the job a feature should perform. Consequently, other users can provide corrections to this misunderstanding.
   b. **Mitigation Correction** is a ***complex*** feedback type. This type of feedback may occur when a user is trying to correct a Mitigation that was provided for a certain problem. Errors in Mitigations may occur due to the lack of contextual information about the tasks the user is doing or environmental information about the softwares or hardware used while applying Mitigation.

The **second** thematic area that we have reached from our forums analysis is the Level of Detail. Level of Detail represents how much information the user provides in their feedback to express their opinions or problems. The information users provide have two major categories: Detail Types and Context. A single feedback can contain a mix of contextual information and several kinds of details. By Detail Types we mean how deep and specific the user is in expressing their feedback. By Context we mean the information the user may provide about the settings of his use to the software or while providing his feedback, which may affect the problems, mitigations, other users' responses. Therefore, this thematic area is considered a complementary area to the feedback types explained in the section above, as it adds more clarity to the feedback descriptions. Below is **our novel description of the Detail Types**:

1. **Concise.** By literal meaning it is used when users provide very short feedback types with no explanations or details. From the analysis we noticed that it is used mostly, when users tend to confirm or negate by just expressing their agreement or disagreement on a feedback. Moreover, it was never used in problem statements or mitigations, since by nature these specific feedback types need explanation to be meaningful.
2. **Explanation** is the opposite of concise, as in this detail type the user is expected to provide as much details in his feedback to make it meaningful for other users. There is no restriction on the use of this detail category with any feedback type, because it is always acceptable to give more details.

3. **Exemplification** is utilized when the users need to provide examples within this text. In the forums' threads that we have analysed examples are always given within explanations especially problem explanations.
4. **Trials** is used closely with problem description where the problem owner who is explaining the problem, shows that he made many attempts to resolve the problem but have failed to reach a Solution. The user posts these trials as a kind of extra explanation of the problem and how it occurs, and also to avoid getting suggestions from other users with same trials that he already made.
5. **Scenario** is used to explain text in a list. A solution can be explained in steps. These steps if verified by the problem owner can be used as a solution scenario to solve similar problems to other users. Moreover, other users may list the problems they have in the problems statement. Other may suggest mitigation to other users in a form of a list of possible actions to try; sometimes it matters to be in a certain order.
6. **Feature Definition** is used to define a user's perception of the usage of a certain feature. This description is sometimes used in problem statements, which helps other users understand why the user is having a problem (i.e. sometimes users have wrong understanding of the usages of a feature). Moreover, users who provide Mitigation may use it a form to document how they use a feature with certain types of tasks. Finally, it is mostly used when users provide Feedback Type: Correction, specifically Problem Correction, where the user corrects the misunderstanding of another user by providing the correct feature definitions to features referenced in the problem statement.
7. **Question** is a simple detail category that is used with Investigations to indicate the question(s) posted for clarification.

Contextual information can carry valuable information that can help make the feedback more understandable or useful. There are five main categories of contextual information that were captured in the forums analysis that map to [13].

1. **Task:** It captures what the user is doing. This is specifically important when the user is describing a Problem feedback type, because it gives to the other users an idea about the context in which the problem occurred, or describing the frequent jobs that the user is involved in in his daily work which helps give an idea to other users about the importance the feature the user is having problem with.
2. **Spatio-Temporal:** In this kind of context the user specifies information related to place and time. From our forums analysis we have found an angle where such information may play useful role. Cases are when users try to explain the timing relationship between two tasks (i.e. two tasks happening together, or one feature corrupts when a user does a certain action). Another Case is when users try to specify some information about a problem in relation to where it occurs in software for example in a certain interface, or when using a certain module.
3. **Personal:** In this kind of context users express their emotional judgments, stress, or information about their expertise, which is repeated mainly with Negation feedback.
4. **Social:** we mean context information related to a user's role at work, and information about co-workers.

5. **Environmental:** is related to a software or hardware specs, versions, and architectures. Users can provide these kinds of information in a problem statement to specify the software version they are using which may differ in the feature with problem from older or newer ones. Therefore, this adds specificity and usefulness to add such information. Moreover, users can add also environmental context in Mitigations to specify that the suggestion or solution works on a certain version, or works well with a certain hardware configuration.

The **third** thematic area concluded from the forums analysis, it was noted that users use four different methods to provide feedback, which are: text, code snippets, snapshots and links. It was notable that some methods were associated with a certain feedback types. (a) The text method is the most commonly used method in all feedback, and even it is used with other methods such as links or snapshots. However, it is important to note that most users use text written in natural language, which leads to lots of misinterpretations. This motivates our goal in creating a new feedback modelling language that utilizes the same methods the users are used to provide their feedback with, but in a patterned way and with the aid of textual keywords. Therefore, this thematic area is considered a complementary area to the feedback types explained above. (b) Code snippets are used to show fragments of code that have problems, or fragments of code to illustrate mitigation, and same for Snapshots. (c) Finally, a further method used by users to express details in forums, is Links. Links are very useful in providing Mitigations whether solutions or suggestions. Users use them to provide all the information they need by referencing the page that contains manuals or illustration the may help the problem owner. They can also, provide extra notes or explanations in their feedback besides the Link.

Finally, we have concluded from our forums analysis this **fourth** thematic area which is measurement. By measurement we mean measuring problem occurrence frequency or voting for mitigations' usefulness. This can simply be done through confirmations and negations that reference Problems or Mitigation feedback Types. By gathering such relationships between different users' feedback, it will allow the system to (a) prioritize the problems according to its rate of occurrence; (b) Also, when the system arrives to a good Mitigation action, the feedback causing this Mitigation could be reused in similar cases.

5 Threats to Validity

Although we have carefully followed the principles in conducting mixed methods approach, our study would still have five main threats to validity: (a) In the focus groups study users were students, researchers, and engineers recruited from Egypt and UK, which might produce a population bias; (b) a common threat to validity in focus groups study is whether all the participants perceived the questions as intended. We have addressed this issue by providing scripts which went through iterative revisions and modifications by two research members and we have undergone a mock-focus group of 2 participants for questions refinements; (c) while the analysis of forums was effective in identifying and describing concepts that construct users' feedback, it is possible that it did not identify all the important aspects and factors that can affect and influence their behaviour in this regard; (d) The number of analysed feedback from the three different forums (200 feedback) could be found medium considering that numerous number of threads available online, we stopped analysis when we reached the stage of saturation; (e) We have targeted forums where business users provide feedback, future research would further investigate general purpose forums (e.g. products, social media) to discover aspects of feedback in a loosely controlled and more open feedback acquisition environments than the one we studied.

6 Architecture for Structured Feedback Acquisition

In this section we explain how we utilize our findings to propose architecture for structured feedback acquisition as presented in Fig. 3. A set of rules that define feedback elements can be derived from the observations that we have reached from the classifications defined in Sect. 4. We propose architecture for structured feedback acquisition that consists of three main components. First, to formalize the definition of rules we propose developing an ontology that constructs the building blocks of user feedback structure elements, their operation rules, and a set of reserved keywords for each concept.

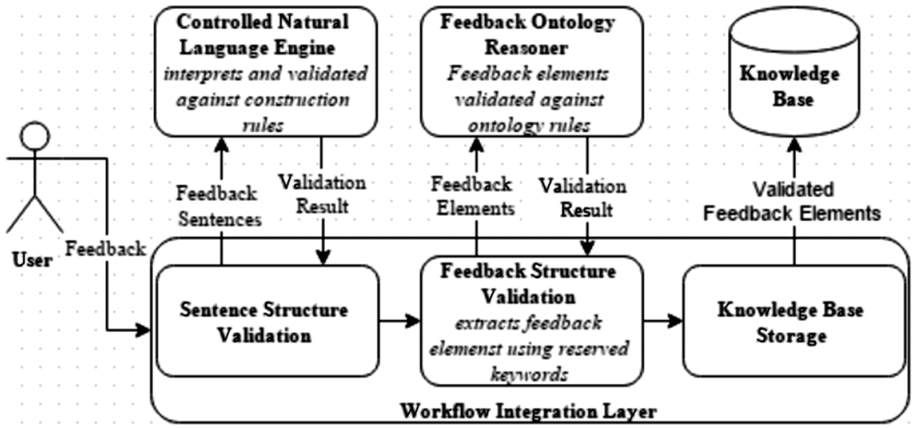
Controlled Natural Language Engine
interprets and validated against construction rules
Feedback Ontology Reasoner
Feedback elements validated against ontology rules
Knowledge Base
User
Feedback
Feedback Sentences
Validation Result
Feedback Elements
Validation Result
Validated Feedback Elements
Sentence Structure Validation
Feedback Structure Validation
extracts feedback elemenst using reserved keywords
Knowledge Base Storage
Workflow Integration Layer

Fig. 3. An architecture for structured feedback modelling

Second, to improve clarity and enable consistent automated semantic analysis of the feedback, a feedback controlled natural language can be employed as an acquisition method for users to provide their feedback. It will restrict the user by general rules such as keeping sentences short and only use the reserved keywords to define textual blocks. This will be achieved by employing an already existing controlled natural language that will act as our text writing foundation that users will use to write their feedback more precisely.

Third, a workflow integration layer orchestrates the workflow between both controlled natural language engine and the ontology reasoner. This layer takes user feedback written in Controlled English, and sends it to the controlled natural language engine that will interpret the text and validate it against the language construction rules. Validated feedback sentences will return to this layer that will extract the feedback elements (that were presented in our findings) using the set of reserved words in the ontology. These elements will be validated against the ontology rules. Finally, this layer will be responsible for storing the validated feedback elements in a knowledge base. This knowledge base will allow engineers to have concrete and formal instances of the feedback extracted by more systematic means, which is more efficient and less error-prone. This can help in evaluating the overall quality of the system, which will help in taking evolution and maintenance decisions.

In our research we suggest employing goal models to represent the stakeholders' goals. We relate it to the feature model to represent both the functional and non-functional requirements of the system. By relating the structured feedback to the feature model, engineers can propagate through the interconnections between them to determine different levels of evaluation information. For example, by looking on the feedback and feature model they can identify most problematic features in the software according to some simple metrics like the no of negative feedback referencing that feature. Or they can look at it from a higher level to see which goals are violated keeping enterprise stakeholders unsatisfied. Stakeholders can be identified and their input can then be used to shape the maintenance and evolution decisions; this ensures their support and improves the quality of the models produced for enterprises. This results in participants having an improved understanding of the problem solving process, and even of their own enterprise.

We will take an example of an actual user feedback on a Problem Extension explained in Sect. 4. The feedback example is from Microsoft's TechNet forum https://goo.gl/CMBDJe: "[Confirmation on existing Problem] *Our office has been struggling with a related problem that maybe you can solve.* [Explanation of the new Extended (related) Problem] *Basically, the same person is repeatedly given a different reviewer name as they work in a document (presumably every time the document is auto saved). For example, if I work for an hour adding edits or comments on a document by the time I'm ready to share it will look like five different people made changes.* [Confirmation on Mitigation that solved part of the problem] *The Inspect Document fix works great to remove all the extra reviewer names, but it changes them all to 'Author'. Do you know how to then either* (*a*) *change 'Author' to the reviewer's actual name or* (*b*) *stop Office from assigning multiple names to the same reviewer? Also, we've tried checking 'Always use these values regardless of sign in' under General to no avail. Thanks - your fix is the closest we've come to a solution and it's greatly appreciated."*

In this example we show how user should write the feedback in a structured format that conforms to the feedback elements and reserved keywords defined by the ontology, the controlled natural language syntax, and the notation that we suggest that will be defined in the intermediate layer.

In this example, the feedback type is problem extension. In order to able to correctly classify this feedback as a problem extension, the user should follow some unique rules that uniquely identifies that type, such as: this feedback should confirm on a previously stated problem in the thread; it should also confirm on a previously stated mitigation in the feedback thread; but also adds a new problem in the feedback content; it should also explain trial that the user has undergone to solve the issue; and provide some environmental context which serves in favour of understanding how the mitigation was applied.

To show the benefits of using a structured format for systematically analysing feedback, we can take the first sentence in the example: "Our office has been struggling with a related problem that maybe you can solve." It can be logically inferred that the user who wrote this feedback agrees on a previously stated problem. However, to reach this conclusion it requires a human interpreter to read, understand and provide such conclusion. But when written: "*I* ***agree*** *on* [***@problem*** *refer to a previously stated problem*]". First, the word "I agree" can be defined as a reserved keyword that indicates that this is a confirmation sentence. Moreover, it accurately refers to a previously stated problem in the thread, which will be validated by the workflow integration layer that handles the communication with the ontology. Therefore, writing the feedback sentence with the new notation removes redundancy, subjectivity, and also provides decisive definitions for the sentences' meanings, and thus can eliminate or remove human interventions.

The next step is the each sentence (i.e. instances content) is validated using the controlled natural language engine. The workflow layer will be responsible for retrieving the sentences that will be sent for validation, and showing the results for the user. In case that the user did not write proper controlled English, this layer will suggest how he can improve his feedback.

7 Related Work

There are several paradigms where the role of users is central such as User centred design [14], User Experience [15], Agile methodology [16], Usability Testing [17]. These techniques can aid the design of enterprise software systems, but they are expensive and time consuming when used for highly variable software designed to be used by a large number of users in contexts that are hardly predictable at design time. Furthermore, our work is similar to End-user Computing [18] in the motivation of involving users and enabling them to change in the system itself to meet their requirements and needs. However, we rely on users to provide feedback in order to decide on maintenance and evolution decisions rather than taking actions.

Recent research has been directed towards involving users in evaluating and modelling evolving requirements for large enterprise softwares. Authors in [19], main contribution is a theoretical understanding of user involvement as a key success factor in implementing and maintaining business intelligence solutions. Moreover, in [20], authors suggest users involvement in developing Business Process Management projects. Their modelling approach involves using User Requirements notation that integrates goals and usage scenarios, from which requirements can evolve. Additionally, in [21] the authors present how strategy maps can be augmented by consumer values to include goals reflecting consumer values, which can be used as requirements for new solutions. All the above work supports the importance of users in driving the enterprise business process as a lifelong activity. However, their work operates on the management of requirements at a rather strategic level to ensure goal satisfaction, and business strategy implementation. In contrast, our work aims to provide engineering approach with concrete constructs to model and acquire feedback and enable their role to take place.

Various works has been done on how to extract requirements from users' feedback. Authors in [22], extract the main topics mentioned in the feedback, along with some sentences demonstrative to those topics using sentiment analysis. Also in [23], have defined a simple domain ontology consisting of generic broad types of feedback and associations. They cluster feedback messages according to the entities they refer to, use natural language parsing and heuristic filtering that can match the detected keywords to domain ontology. Moreover, in [24], the research aims on providing an elicitation approach that can offer new opportunities for users to support them in documenting their needs using a mobile tool. In contrast, and instead of analysing given feedback, e.g. through forums and social networks, our work contributes to forward engineer the acquisition process itself making the analysis more efficient.

When engineering feedback, we need to use a language understood by users and at the same time traceable to the requirements model and knowledge. Goal Model [25], Feature Model [26] and Business Processes [27] seem to be potential models which link the space of the business to the space of users and their understanding of the system.

8 Conclusion and Future Work

This paper has presented a two phase empirical study. The first phase focus group study focused on the different aspects of the activity of interacting with users and acquiring their feedback, which gave a broad perspective for open research challenges. While the second forums analysis study examined actual users' feedback to reach a classification of feedback structures and their elements. The findings can be employed to develop a collaborative architecture that utilizes structured feedback for extracting requirements in a systemized way where the risks resulting from human interventions are minimized. Therefore, our results serve as a foundation step for a holistic approach for the structuring and use of users' feedback for crowdsourced software evaluation. Furthermore, from the feedback classification reached from the empirical study, we can derive new templates that combine multiple feedback and feedback types to form new cases that can inform the engineers by giving them a detailed view of the software's evaluation status from the users' point of view.

Acknowledgments. The research is supported by a European FP7 Marie Curie CIG grant (the SOCIAD project) and Bournemouth University through the Graduate School PGR Development fund.

References

1. Jarke, M., Loucopoulos, P., Lyytinen, K., Mylopoulos, J., Robinson, W.: The brave new world of design requirements. Information Systems **36**(7), 992–1008 (2011)
2. Cleland-Huang, J., Jarke, M., Liu, L., Lyytinen, K.: Requirements management–novel perspectives and challenges. In: Dagstuhl Reports 2, vol. 10, pp. 117–152 (2013)

3. Hosseini, M., Phalp, K., Taylor, J., Ali, R.: Towards crowdsourcing for requirements engineering. In: REFSQ 2014, Germany (2014)
4. Snijders, R., Dalpiaz, F., Hosseini, M., Shahri, A., Ali, R.: Crowd-centric requirements engineering. In: UCC 2014, pp. 614–615. IEEE, London (2014)
5. Ali, R., Solis, C., Omoronyia, I., Salehie, M., and Nuseibeh, B.: Social adaptation: when software gives users a voice. In: ENASE 2012, Poland (2012)
6. Ali, R., Solis, C., Salehie, M., Omoronyia, I., Nuseibeh, B., Maalej, W.: Social sensing: when users become monitors. In: ESEC/FSE 2011, Hungary, pp. 476–479 (2001)
7. Sherief, N., Jiang, N., Hosseini, M., Phalp, K., Ali, R.: Crowdsourcing software evaluation. In: EASE 2014, pp. 19. ACM, London (2014)
8. Almaliki, M., Ncube, C., Ali, R.: The design of adaptive acquisition of users feedback: An empirical study. In: RCIS 2014. IEEE, Morocco (2014)
9. Almaliki, M., Ncube, C. Ali, R.: Adaptive software-based Feedback acquisition: a persona-based design. In: RCIS 2015. IEEE, Greece (2015)
10. Kontio, J., Lehtola, L., Bragge, J.: Using the focus group method in software engineering: obtaining practitioner and user experiences. In: ISESE 2004. IEEE, USA (2004)
11. Braun, V., Clarke, V.: Using thematic analysis in psychology. Qual. Res. Psychol. **3**(2), 77–101 (2006)
12. Crawford, H.K., Leybourne, M.L., Arnott, A.: How we ensured rigor from a multi-site, multi-discipline, multi-researcher study. In: Forum Qualitative Sozialforschung/Forum: Qualitative Social Research, vol. 1, no. 1 (2000)
13. Krogstie, J., Lyytinen, K., Opdahl, A.L., Pernici, B., Siau, K., Smolander, K.: Research areas and challenges for mobile information systems. Int. J. Mobile Commun. **2**(3), 220–234 (2004)
14. Vredenburg, K., Mao, J.Y., Smith, P.W., Carey, T.: A survey of user-centered design practice. In: CHI 2002, pp. 471–478. ACM, Minneapolis, Minnesota, USA (2002)
15. Law, E.L.C., Van Schaik, P.: Modelling user experience - an agenda for research and practice. Interact. Comput. **22**(5), 313–322 (2010)
16. Dybå, T., Dingsøyr, T.: Empirical studies of agile software development: a systematic review. Inf. Softw. Technol. **50**(9–10), 833–859 (2008)
17. Adikari, S., McDonald, C.: User and usability modeling for HCI/HMI: a research design. In: ICIA 2006, pp. 151–154. IEEE (2006)
18. Doll, W.J., Torkzadeh, G.: The measurement of end-user computing satisfaction. MIS Q. **12**, 259–274 (1988)
19. Yeoh, W., Koronios, A.: Critical success factors for business intelligence systems. J. Comput. Inf. Syst. **50**(3), 23–32 (2010)
20. Pourshahid, A., Amyot, D., Peyton, L., Ghanavati, S., Chen, P., Weiss, M., Forster, A.J.: Business process management with the user requirements notation. Electron. Commer. Res. **9**(4), 269–316 (2009)
21. Svee, E.-O., Giannoulis, C., Zdravkovic, J.: Modeling business strategy: a consumer value perspective. In: Johannesson, P., Krogstie, J., Opdahl, A.L. (eds.) PoEM 2011. LNBIP, vol. 92, pp. 67–81. Springer, Heidelberg (2011)
22. Galvis Carreño, L.V., Winbladh, K.: Analysis of user comments: an approach for software requirements evolution. In: ICSE 2013, pp. 582–591. IEEE Press, CA, USA (2013)
23. Schneider, K.: Focusing spontaneous feedback to support system evolution. In: RE 2011. IEEE, Italy (2011)
24. Seyff, N., Graf, F., Maiden, N.: Using mobile RE tools to give end-users their own voice. In: RE 2010, pp. 37–46. IEEE Computer Society, Sydney, Australia (2010)

25. Yu, E.S.: Social modeling and *i**. In: Borgida, A.T., Chaudhri, V.K., Giorgini, P., Yu, E.S. (eds.) Conceptual Modeling: Foundations and Applications. LNCS, vol. 5600, pp. 99–121. Springer, Heidelberg (2009)
26. Kang, K.C., Kim, S., Lee, J., Kim, K., Shin, E., Huh, M.: FORM: a feature-; oriented reuse method with domain-; specific reference architectures. Ann. Softw. Eng. **5**(1), 143–168 (1998)
27. OMG, B.P.M.N., Version 1.0. OMG Final Adopted Specification, Object Management Group (2006)