SOCIAL ADAPTATION
When Software Gives Users a Voice

Raian Ali[1], Carlos Solis[2], Inah Omoronyia[3], Mazeiar Salehie [3] and Bashar Nuseibeh[3,4]
[1] *Bournemouth University, UK,* [2] *FEXCO, Killorglin, Ireland.* [3] *Lero - Limerick University, Ireland.* [4] *Open University, UK*

Keywords: Requirements Engineering, Requirements-Driven Adaptation, Social Adaptation

Abstract: Adaptive systems are characterized by the ability to monitor changes in their volatile world and react to monitored changes when needed. The ultimate goal of adaptation is that users' requirements are always met correctly and efficiently. Adaptation is traditionally driven by the changing state of the system internal and its surrounding environment. Such state should be monitored and analyzed to decide upon a suitable alternative behaviour to adopt. In this paper, we introduce another driver for adaptation which is the users' collective judgement on the alternative behaviors of a system. This judgmenet should be infered from the individual users' feedback given iteratviely during the lifetime of a system. Users' feedback reflects their main interest which is the validity and the quality of a system behaviour as a means to meet their requirements. We propose *social adaptation* which is a specific kind of adaptation that treats users' feedback, obtained during the software lifetime, as a primary driver in planning and guiding adaptation. We propose a novel requirements engineering modelling and analysis approach meant for systems adopting social adaptation. We evaluate our approach by applying it in practice and report on the results.

1 INTRODUCTION

Self-adaptive systems are designed to autonomously monitor and respond to changes in the operational environment and the system's own state (Laddaga, 1997). Each change can trigger a different response to satisfy or maintain the satisfaction of certain requirements (Salehie and Tahvildari, 2009). Self-adaptive systems have basic design properties (called *self-**). Self-protection property means the ability to monitor security breaches and act to prevent or recover from their effects. Self-optimization means the ability to monitor the resources availability and act to enhance performance. Self-healing means the ability to monitor faults that have occurred or could occur and cure or prevent their effects. Finally, self-configuration means the ability to monitor changes related to all of the other properties and add, alter, or drop upon certain software entities (Murch, 2004).

Self-adaptivity is highly reliant on the system ability to autonomously monitor the drivers of adaptation (security breaches, the available resources, faults and errors, etc.). In (Ali et al., 2011c), we argued that there are adaptation drivers which are unmonitorable by relying on solely automated means. The judgement of users on the validity and quality of each of the alternative behaviours of a system, is an example of that. Such judgement is an important driver for adaptation which enables a system to know how the validity and quality of its alternative behaviours are socially evaluated. The feedback of individual users is the main ingredient to obtain and process as a preliminary step for a collective planing of adaptation. We define *Social Adaptation* as the system autonomous ability to analyse users' feedback and choose upon an alternative behaviour which is collectively shown to be the best for meeting requirements in a context.

Obtaining and processing social feedback to support adaptation decision helps to accelerate adaptation. This is particularly true for highly variable systems which incorporate a large number of alternatives. For such systems, a large number of users will be required to validate all of the system alternatives. Establishing such validation as a design time activity which is directed by designers, as often done in usability testing and user-centred design (Dumas and Redish, 1999; Vredenberg et al., 2001) is highly expensive, time-consuming and hardly manageable. Social adaptation allows the actual users to act as validators and give feedback at runtime so that the system can analyse the set of users' feedback and validate upon each system alternative by analysing how it is collectively judged. This is particularly important when there is a large space of system alternatives already implemented and enacted and their quality judgement is likely to change frequently over time.

Social adaptation advocates the repetitive obtaining and analysis of social feedback to keep adaptation up-to-date. Due to the dynamic nature of the world, which includes users trends and experience, competitive technologies, and context of use, users' judgement on the system is sometimes dynamic. Thus, a traditional one step design-time system validation and customization might lead to judgements which are correct but only temporarily. For example, users' who currently like interaction via touch screens, might dislike it in the future when a better interaction technology is devised. Users' evaluation of a system alternative is not static and what is shown to be valid at certain stage of validation may become eventually invalid. Thus, the repetitive nature of social adaptation allows to accommodate the volatile nature of users judgement on a system which would not be easy to accommodate in traditional usability testing sessions.

Requirements are the natural subject of social feedback and, thus, social adaptation should be in the first place a requirements-driven adaptation. Upon the execution of a system alternative, users will be primarily concerned whether their requirements are met correctly. Users would also care about the degree of excellence of an execution against certain quality attributes. Consequently, social feedback concerns the validity and the quality of a system alternative as a way to meet users' requirement. This judgement is affectable by context which, thus, has to be monitored. The same system alternative could be judged differently when operating in different contexts. When a context *C* occurs, social adaptation analyses the feedback provided for each alternative when it was executed in *C* and choose the alternative which is collectively judged to suit *C*. However, as we proposed in (Ali et al., 2010), some contexts have clear influence on the applicability and quality of systems behaviour regardless the users' feedback.

In this paper, we propose social adaptation which relies on the users' collective judgement on the alternative behaviours of a system as a first-class driver for adaptation. We discuss the foundations and motivations of social adaptation. We provide a conceptualization of the main artefacts needed in a requirements model for systems adopting social adaptation. We develop analysis techniques to process social feedback and choose upon the system alternative which is collectively shown to be the most appropriate to meet a requirement. We evaluate our approach by applying it on a messenger system looking to convey messages in a way collectively shown to fit a certain context.

The paper is structured as follows. In Section 2 we define and motivate and discuss main principles of social adaptation. In Section 3 we present a set of core modelling artefacts for social adaptation from a requirements engineering perspective. In Section 4 we develop analysis algorithms to process social feedback and adapt the system at runtime. In Section 5 we evaluate our approach in practice. In Section 6 we discuss related work and in Section 7 we present our conclusions and future work.

2 SOCIAL ADAPTATION

Social adaptation is a specific kind of adaptation which responds to the social collective judgement on the correctness and efficiency of a system behaviour in meeting users' requirements. Social adaptation treats social feedback as a primary driver for adaptation. Social feedback allows a runtime continuous evaluation of each of the alternative behaviours of a system. The alternative which is socially proved to be correct and more efficient in a certain context will be the one to apply when that context occurs. That is, social adaptation allows for inferring the applicability and the quality of each system behaviour over a space of different contexts. In Figure 1, we outline the loop of social adaptation where the system analyses social feedback and decide upon which alternative behaviour to apply, applies it, and finally gets and stores the feedback of the users of that operation.

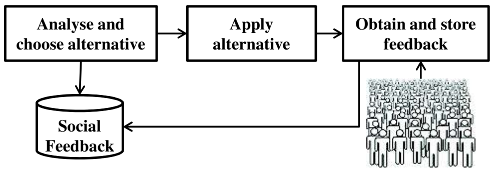

Figure 1: Social Adaptation Loop.

The key characteristic of social adaptation is the incorporation of users' perception and judgement as a part of the system monitor. In self-adaptation, the system has to monitor changes in its internal and operational environment autonomously. Monitored events and states could indicate security breaches triggering a self-protecting action, or new settings of the available resources triggering a self-optimizing action, or faults triggering a self-healing action. In social adaptation, the system has to monitor the social judgement about its role in meeting requirements. Such judgements concern humans opinions and conclusions rather than events and states in the system environment. In self-adaptation, the system enjoys an autonomous ability to monitor all information which is necessary for taking adaptation decisions which means that the monitor is a fully automated component. Social adaptation, on the other hand, requires capturing its own kind of drivers, the users' judgement, which is un-monitorable by relying on solely automated means as we discussed in (Ali et al., 2011c). Social adaptation requires a socio-technical monitor which involves users' perception as an integral part of the system monitor.

Social adaptation includes users in the activity of shaping adaptation and evolving it continuously during the lifetime of a software. Designers should plan adaptation at design time and leave certain adaptation decisions to be taken collaboratively by the community of users at runtime. Users are enabled to provide their feedback and the system will analyse users feedback and adapt accordingly. Self-adaptive systems follow an evolution rationale, which is planned at design time, until the system apparently fails in taking correct adaptation decisions where a maintenance evolution has to take place. Social adaptation enriches this evolution paradigm by allowing users-driven runtime evolution. A socially-adaptive system responds to the users feedback which is itself likely to evolve over time, and thus the adaptation rationale itself evolves without the need to re-design the system.

As a running example to illustrate the rest of this paper, we consider the case of a gallery-guide system designed to support the visitors of an art gallery. The system has two alternatives to convey information about art pieces to the visitors. *PDA-based system alternative*: by which the system will use the visitor's PDA (personal digital assistant) to explain, possibly interactively, about an art piece. *Staff-supported system alternative*: by which the system will try to find a mentor to meet with the visitor and explain the art piece and also facilitate the meeting. Each of the two alternatives fits a set of different contexts. For certain contexts, designers can not be fully certain of which alternative (PDA-based, Staff-supported) is more appropriate. As a solution, they leave this decision to be taken by the users collectively. The system will try these two alternatives in the different context where designers are uncertain of their quality and validity and get user feedback and infer the fitness between each system alternative and a context variation. This process should be iterative to cope with the aforementioned fact that users judgement is not static and likely to change over time.

Users can judge and give feedback whether a system alternative is a valid means to reach their requirements and its quality. We classify social feedback into two kinds:

- *System validity* which concerns whether a certain applied system alternative succeeds in meeting a certain requirement in a certain context. Users are able to give feedback using requirements-related terms rather than software-related ones. For example, upon executing the PDA-based system alternative, a gallery visitor who is not really familiar with the use of PDAs would give a feedback as a Boolean answer saying "I could not ask for and get information". The visitors cannot generally explain how the complexity or simplicity of the HCI design prevented or facilitated the role of the system in the process of asking for and getting information.
- *System quality* which concerns the degree of excellence of a system alternative. For example, a visitor who is familiar with the use of PDAs but had to wait for a long time to meet a mentor, would admit the delivery of information upon the execution of the PDA-based system alternative, i.e. the requirement is reached, but would probably give a negative assessment of some quality attributes like "comfort" and "quick response".

3 MODELLING REQUIREMENTS FOR SOCIAL ADAPTATION

In the last section we discussed the definition and motivations of social adaptation and its main characteristics. It relies on users to give feedback about the validity and the quality of each of the different system alternatives. This feedback could be influenced by certain contextual attributes. The ultimate goal of social adaptation is to identify the best alternative to adopt in each settings of contextual influencers. In this section, we relay on our discussion in the last section and propose a set of fundamental artefacts a requirements models should include when designing systems to enact social adaptation. Figure 2 shows the metamodel of these artefacts and Figure 3 shows an instance of it. It is worth pointing out that this model is meant to operate on the top of established requirements models which capture the relation between requirements, quality attributes and system alternatives. It is meant to extend such requirements models by the social feedback about this relation and the context influence on it. We will show an example of the application of our metamodel on a main-stream requirements model, the goal model, in Section 5.

The central horizontal part of the metamodel of Figure 2 captures the relation between the requirements and quality attributes on one side, and the space of system alternatives one the other. Requirement indicates an intention to reach a certain state of the world. System Alternative is a synthesis between human and automated activities designed to meet a requirement. A requirement could be reached via multiple system alternatives. Quality Attribute is a distin-

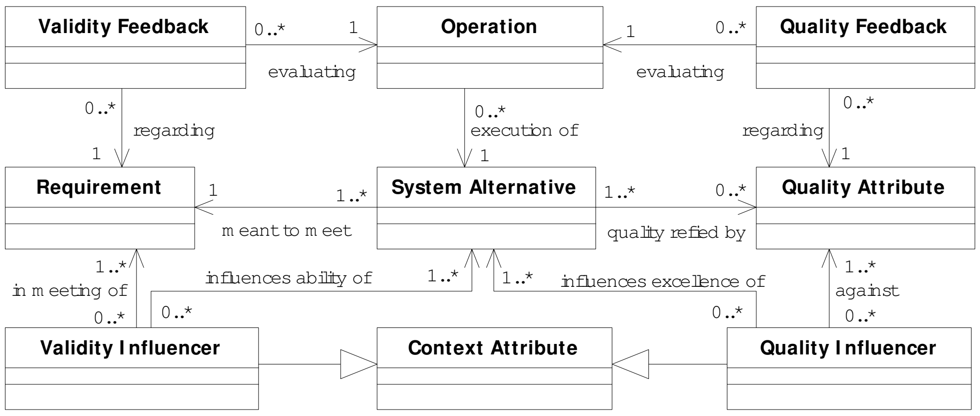

Figure 2: Requirements modelling for Social Adaptation

Instance of the Model	
Requirement	R: visitor can ask for and get information about an art piece
System Alternatives	S_1: PDA-based (input handling, interactive presentation, video, etc.)
	S_2: Staff-supported (estimating time to meet, notifying staff, etc.)
Quality Attributes	Q_1: visitor is well-informed
	Q_2: visitor comfort
Validity Influencers	C_1: visitor' age, C_2: visitor' technology expertise level, influence the ability of S_1 to meet R
	C_3: estimated time for a staff to meet visitor, C_4: visitor has companions? Influence the ability of S_2 to meet R
Quality Influencers	C_1, C_2, and C_5: complexity of art piece information, influence the quality of S_1 with regards to Q_1
	C_1, C_2, and C_6: user movement status (moving, standing, sitting), influence the quality of S_1 with regards to Q_2
	C_7: staff expertise and C_8: staff ability to speak the visitor's native language, influence the quality of S_2 with regards to Q_1
	C_3 and C_9: existence of a free seat closed to the visitor's location, influence the quality of S_2 with regards to Q_2
Runtime Feedback Example	
Operations	Operation_1: execution of S_1. The values of its relevant context attributes are C_1= >65 years, C_2=low, C_5: low, C_6= standing.
	Operation_2: execution of S2. The values of its relevant context attributes are C3 = <5 min, C_4= alone, C_7= medium, C_8= fair, C_9= no
Validity Feedback	Operation_1.Validity_feedback= False (R is not reached)
	Operation_2.Validity _feedback= True (R is reached)
Quality Feedback	Operation_1. Quality_feedback is irrelevant (R was judged unreached)
	Operation_2.Quality_feedback(Q_1)= medium Operation_2.Quality_feedback(Q_2)= high

Figure 3: An instance of the model of requirements artifacts for social adaptation shown in Figure 2

guished characteristic of the degree of excellence of a system alternative. In our framework, these three artefacts and the relations between them are specified by the designers at design time and are static and, thus, are not subject of monitoring at runtime.

The lower part of the metamodel of Figure 2 stands for the context influence on the relation between the system alternatives on one side and the requirements and quality attributes on the other. Context Attribute is a distinguished characteristic of the environment within which the system operates. Validity Influencer is a context attribute that influences the ability of a system alternative to meet a requirement. Quality Influencer is a context attribute that influences the degree of excellence of a system alternative with respect to a quality attribute. The context attributes, of both categories, are specified by designers at design time and monitored at runtime, i.e. the real values are obtained and stored at runtime.

The upper part of the metamodel of Figure 2 stands for the social feedback that expresses users' judgements about each operation of a system alternative. Operation is a single execution of a system alternative. Validity Feedback is a Boolean judgement given by a user concerning his evaluation whether an operation has led to meet his requirement. Quality Feedback is an assessment, reified by a numeric value, given by a user concerning his evaluation of an operation against a quality attribute. The social feedback, of both categories, is specified at design time by designers. The value of the feedback relevant to a specific system alternative is obtained from users and stored at runtime after an operation of that alternative is executed.

In this work, and to enable the analysis which we propose in Section 4, we restrict the values of each context attribute to belong to an enumeration specified by the analyst. For example, the visitor age could be specified to be in {"<18", "between 18 and 25", "between 25 and 65", ">65"}. The validity feedback is already restricted to be a Boolean value. We also restrict the quality feedback value to be within a range of integers [0..n] where 0 stands for "the alternative has very bad quality against the quality attribute q" and n stands for "the alternative is excellent against the quality attribute q".

4 SOCIAL ADAPTATION ANALYSIS

The main goal of obtaining social feedback is to support the system decision about the best alternative to apply for reaching users' requirements and overcome the designers' uncertainty about this decision and cope with the changing trends of users and the world over time. When the system has to meet a requirement, it has to choose an alternative to apply. The system has to assess the collective judgement of each alternative of being a valid and a good-quality means to meet requirements. We propose to take into consideration basic factors that together help for inferring the collective judgement on the validity and quality of a system alternative. We emphasize here that this set is not restricted and more research would lead to discover yet other factors. In what follows, we discuss our proposed factors taking examples from Figure 3.

- *Feedback value*. This factor stands for the values given by the users when evaluating the validity and quality of each operation of a system alternative. Users provide the validity feedback by giving a Boolean answer reflecting their judgement whether the operation led to reach their requirements. Users evaluate the quality of each operation of a system alternative against each quality attribute by giving a value within a designated rank [0..n] where 0 means the lowest quality and n means the highest. These values are the basic factor in assessing the validity and the quality of a system alternative.
- *Feedback relevance*. This factor stands for the meaningfulness of each of the users' validity and quality feedback when assessing a system alternative. This relevance will be interpreted as a weight for the feedback value which reifies the user' judgement of the validity and quality of a system alternative. We consider two sub-factors which influence the relevance:
  - *Feedback context*. This factor stands for the match between the context of a particular operation of a system alternative for which the feedback was given, and the current context where a decision about that alternative has to be taken. The validity of a system alternative and its quality against each quality attribute are affected by a set of context influencers as we explained earlier. The more the match between the values of these context influencers when the feedback was given and their values at the assessment time, the more relevant the feedback is. For example, suppose the system is assessing the validity of the PDA-based system alternative and that the current values for its validity influencers are (C_1: visitor' age) = "> 65", (C_2: visitor's technology expertise level) = "low". Suppose we have two validity feedback F_1= "valid" and F_2= "invalid" and the values of contexts

for F_1 and F_2 are C_1= ">65", C_2= "medium", and C_1= ">65", C_2= "low", respectively. Then the relevance of F_2 is higher than the relevance of F_1 because more context influencers match in F_2 than in F_1. Thus, and according to the feedback context factor, the alternative will be judged closer to "invalid" than "valid".

  – *Feedback freshness*. This factor stands for the recentness of the feedback. Feedback relevance is proportional to its freshness. That is, the more recent the feedback is, the more meaningful. There could be several ways to compute feedback freshness. One design decision could compute it by dividing its sequential number by the overall number of feedback of its kind. For example, suppose that PDA-based system alternative got two validity feedback, the earlier (with a sequential number 1) F_1= "invalid" and the later (with a sequential number 2) F_2= "valid". Thus, and according to the feedback freshness factor, the alternative will be judged closer to "valid" than "invalid".

- *User's preferences*. This factor stands for the preferences of a user while assessing the overall quality of a system alternative. The analysis of the social feedback results in giving an overall assessment of each system alternative against each quality attribute. However, the assessment of the overall quality, i.e., the aggregated quality, of a system alternative may consider how the user appreciates each of these quality attributes. Similarly to the work in (Hui et al., 2003), we allow users to express their preferences by ranking the degree of importance of each of the quality attributes. For example, suppose that by analysing the historical quality feedback, the PDA-based alternative quality against Q_1= "visitor is well-informed" was assessed to 3 and against Q_2 = "visitor's comfort" was assessed to 2. Suppose that the Staff-supported alternative quality against Q_1 was assessed to 2 and against Q_2 to 3. If a user appreciates Q_1 more than Q_2 then PDA-based alternative overall quality will be assessed higher than that of the Staff-supported system alternative, and vice versa.

The algorithm Assessing Validity shown in Figure 4 computes the collective validity judgement of a system alternative based on the validity feedback users have provided in the past. It takes as input a system alternative s, the set of validity influencers C which affect the ability of s to meet the requirement it is designed to reach and the actual values of these influencers at the time of assessment $C.Values$. It gives as output an assessment of the validity of the statement *"s is a valid means to meet the requirement it is designed for"*. First, the algorithm identifies the operations OP of the system alternative s which got validity feedback from users (Line 1). If the system alternative has no validity influencers then the *context match* factor is irrelevant and the algorithm returns simply the proportion of valid operations over the overall number of operations $\mid OP \mid$ multiplied (weighted) by the average freshness of the operations set OP (Lines 2-3).

When the system alternative has validity influencers, the algorithm iterates for each possible partial or complete match of the context validity influencers at the feedback time and the assessment time (Lines 7-17). For each combination of validity influencers C_i with a cardinality i, the algorithm identifies the set of operations OP_C_i whose validity influencers values ($o.C_i.Values$) match with the validity influencers values at the assessment time ($C.Values$) (Lines 9-11). The algorithm then computes the validity probability concerning the context matches of the cardinality i by dividing the number of valid operation of $Op.C_i$ by $\mid Op.C_i \mid$ (Line 12). The relevance of this probability is decided by both the cardinality of context match (i/$\mid C \mid$) and the value of the freshness factor (Avg_Freshness($Op.C_i$)), computed as we explained earlier (Line 13). The algorithm then multiplies the relevance with the computed validity to get the relevant (i.e., the weighted) validity (Line14). The algorithm then (Lines 15-16) accumulates the relevance and the relevant validity into the variables *relevant_validty_sum* and *relevance_sum* (initiated at Lines 5-6). After the iteration goes through all partial and complete context matches, the algorithm gives the overall assessment by dividing the *relevant_validty_sum* by the *relevance_sum* (Line 18).

Similarly to this algorithm, we have developed the algorithm Assessing Quality to calculate the collective judgement of the quality of a system alternative against a quality attribute. The main difference with regards to Assessing Validity algorithm is the consideration of quality feedback of only the operations with positive validity feedback as negative validity feedback makes any quality feedback irrelevant. Moreover, Assessing Quality deals with the average value of the quality feedback provided for an alternative against a quality attribute. Assessing the overall quality of an alternative considers also the users' preferences expressed via ranking the importance of each quality attributes. For the limitation of space, we have not included neither this algorithm nor the examples which explains both algorithms. For details, please see our technical report (Ali et al., 2011b).

```
Algorithm: Assessing Validity
Input:      s: System alternative
            C: {c, c is a validity influencer of s}
            C.Values: {(c,v), c in C and v = c's monitored value}
Output:     assessed validity of (s,r)
1.    OP:= {o ∈ s.Operations, o got a validity feedback}
2.    If |C| = 0 then
3.        RETURN Avg_Freshness (OP) * (|{o in OP, o.validity_feedback= 'valid'}| / |OP|)
4.    Else
5.        relevant_validity_sum:= 0
6.        relevance_sum:= 0
7.        For i = 1 to |C| Do
8.          OP_Ci := Ø
9.          For each Ci ∈ $2^C$ and |Ci|= i
10.           OP_Ci = OP_Ci U {o in Op; ExactMatch(o.Ci.Values, C.Values)}
11.         EndFor
12.         validity_Ci:= |{o in OP_Ci; o.validity_feedback= 'valid'}| / | OP.Ci|
13.         relevance_Ci  := (i / |C| + Avg_Freshness(OP.Ci))/2
14.         relevant_validity_Ci := relevance_Ci   * validity_Ci
15.         relevant_validty_sum:= relevant_validity_sum + relevant_validity_Ci
16.         relevance_sum:= relevance_sum + relevance_Ci
17.       EndFor
18.       RETURN relevant_validity_sum/relevance_sum
19.   EndIf
```

Figure 4: Assessing Validity Algorithm

5 EVALUATION

To evaluate our framework, we have organized a lab session and invited 5 researchers specialized in requirements engineering and their research is in the area of requirements-driven adaptive systems engineering. We have explained our design principles of modelling requirements for socially-adaptive systems. We then explained the scenario of an adaptive messenger system which is able to deliver messages in different ways and asked the participants to draw a goal model, namely Tropos goal model (Bresciani et al., 2004), presenting its requirements together with the validity influencers (on the decompositions and means-end) and the quality influencers on the contribution links between any goal/task and a set of softgoals which evaluates all alternatives (please refer to our report in (Ali et al., 2011b) for more details). The left part of Figure 5 shows a goal model built during the session; in the other part we extract one alternative and show its validity and quality influencers.

The main issues which were raised by the participants concerned the following. *Context monitorability*: besides the validity and quality judgement, the values of certain context attributes might not be monitorable by relying solely on automated means and may require users to act as monitors. *Quality and context attributes identification*: users should be given the chance to define extra quality and context attributes which were not considered by the designers. *Context influence (un)certainty*: the design should also consider that some context influences are already known and our approach should accommodate both certain and uncertain context influences. *Runtime evolution of the requirements model*: the analysis should be enriched to also decide parts of the model which should be removed when collectively proved to be invalid or having very low quality and also help the analyst by indicating loci in the model where an evolution should take place. *Feedback relevance*: we need mechanisms to exclude feedback which are not significant or inconsistent taking into consideration several factors like the user' history and pattern of use. We need to handle these limitations of our approach to maximize its applicability and operations and broaden and optimize its automated analysis.

To evaluate the social adaptation analysis proposed in Section 4, we have asked 15 users to provide validity and quality feedback about 6 alternatives of a prototype messenger in 3 different contexts. For the quality feedback we have considered 2 quality attributes (Q_1: less distraction, Q_2: less effort). Then we have run the validity and the quality assessment analysis for the different alternatives in the different contexts taking as input the set of obtained users' feedback. Then, we have surveyed a set of 3 other users (testing users) and compared their feedback to the collective judgements obtained by running the validity and quality analysis algorithms.

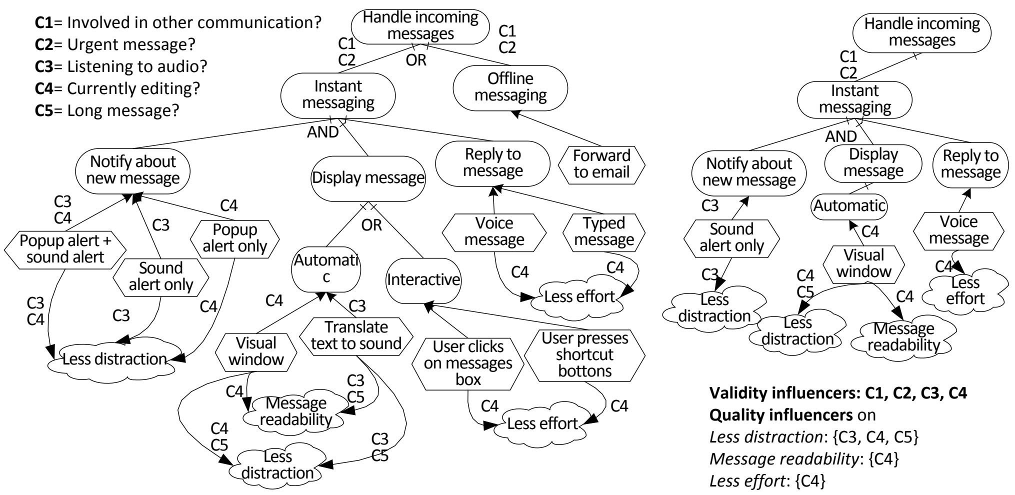

Figure 5: A goal model (left) and an alternative of it with quality and validity influencers (right)

To evaluate the validity assessment analysis, we have considered 3 alternatives and 3 different contexts where the validity assessment algorithm has given high probability for validity (>85 %). We have asked the 3 users to provide their validity feedback (thus 27 feedback were provided in total) and out of which 23 feedback had the value "valid" and 4 had the value "invalid". This shows a good match between the collective validity judgement of the 15 initial users, computed by the validity assessment algorithm, and the judgement of each of the testing users. It is worth pointing out that the consensus of users about the validity is more likely to be achieved than the quality due to the nature of the decision about validity which has a clear-cut criteria to be judged upon.

To evaluate the quality assessment analysis, we have considered 3 other alternatives and 3 different contexts and asked the 3 testing users to provide the quality feedback of each alternative in each of the contexts against Q_1 and Q_2. Table 1, show the ranking of the automated analysis assessment and the testing users assessment of the 3 different messenger system alternatives (A_1, A_2, A_3) against the quality attribute Q_1: "Less Distraction" (LD), in 3 different contexts (C_1, C_2, C_3). The acronym "Sys" stands for the assessment given by the algorithm Assess Quality and U_i stand for each testing user. For example, and taking the first data column, the automated analysis of the users' feedback to obtain the quality assessment of A_1, A_2, and A_3 alternatives against the quality attribute "Less Distraction" (LD) within the context C_1 indicated that A_2 has the best quality, and A_1 has the second and A_3 has the lowest quality. The ranking the automated analysis gave in this case, matched the ranking made based on the quality assessment each testing user gave. In the context C_2, the user U_2 gave a ranking different from the one given by the automated analysis and we highlight the mismatching results. The same for U_2 and U_3 for the context C_3. As shown in the table, the matching between the collective quality judgement computed by the quality assessment algorithm and the testing users feedback was good enough (21 out of 27). For the quality attribute Q_2= "Less Effort", the number of matches between the automated collective judgement and testing users was also good (18 matches out of 27 comparisons).

There are several threats to validity concerning our evaluation. The first threat concerns the small size scenario which we used (the messenger system) and the relatively small number of participants who gave feedback. Larger scale experiment would maximize the credibility of our results. The second is the kind of practitioners who modelled the scenario who already had a good expertise in requirements modelling and self-adaptive systems. Communicating our principles and guidelines to novice practitioners might raise other concerns related to the practitioners understandability and acceptability of our framework. The third is the relatively short period of time for getting feedback which makes it hard to evaluate certain things such as the evolution of social trends. The fourth is that our users committed to provide feedback which might not be the case with real users. Thus, we still need to study users' acceptance of our approach.

6 RELATED WORK

There are several research areas highly related to Social Adaptation. We are mainly concerned about addressing challenges in the research in software-

Q_1 :	C_1				C_2				C_3			
LD	Sys	U_1	U_2	U_3	Sys	U_1	U_2	U_3	Sys	U_1	U_2	U_3
A_1	2	2	2	2	2	2	**1**	2	2	2	2	2
A_2	1	1	1	1	1	1	**2**	1	1	1	**3**	**3**
A_3	3	3	3	3	3	3	3	3	3	3	**1**	**1**

Table 1: For each context, the rank of the different alternatives (mismatches are in bold font)

engineering for adaptive systems and, particularly, requirements-driven adaptation. Our proposed approach enriches requirement-driven adaptation research by a systematic approach to incorporate users' collective judgement of systems behaviours, perceived as means to meet requirements, as a primary adaptation driver. This helps for a more holistic adaptation which overcomes the limitation of automated means to judge if requirements (functional and non-functional) are being met (Ali et al., 2011c). It also allows for reflecting the collective voice of users' rather than relying on designers' judgements which could be, or eventually become, invalid (Ali et al., 2011a). This systematic approach is inline with an increasing trend to involve users in the adaptation loop (Cheng et al., 2008). We concretely position users as feedback providers and specify their feedback structure and provide mechanisms to process it and assess the collective judgement of system behaviours.

The seminal work of (Fickas and Feather, 1995) highlights the importance of requirements monitoring at runtime as a basic and essential step for planning and enacting adaptation. This is a fundamental principle of our approach. Cheng et al. note that in requirement models uncertainty have not been explicitly addressed in traditional requirements engineering (Cheng et al., 2008). We address uncertainty by involving users in evaluating the system alternatives against their capability to meet requirements so that certainty is achieved based on the perception of users regarding the actual operation. In (Silva Souza et al., 2011) the authors note that the (partial) un-fulfilment of requirements triggers adaptation. They introduce awareness requirements to refer to success, failure, performance and other properties of software requirements (i.e. meta-requirements) and propose that the system should monitor changes in these properties and decide upon when and what adaptation should take place. We argued that certain information can not be monitored by the system and require an explicit users' intervention via feedback and provided a systematic way to realize that.

The work in (Baresi et al., 2010) proposes FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) for requirements-driven adaptation at runtime. FLAGS extends goal models mainly with adaptive goals which incorporate countermeasures for adaptation. When goals are not achieved by the current course of execution, adaptation countermeasures are triggered. This approach can be potentially integrated with ours so that when social collective judgement indicates some failures, a certain adaptation goals should be activated. The work in (Qureshi and Perini, 2010) emphasizes on flexibility of requirements refinement and provide a method that supports the runtime refinement of requirements artifacts as a repetitive activity performed collaboratively between the users and the application itself. We plan to benefit from this approach to incorporate users in the modelling process at runtime and not only the judgement of the system different behaviours.

7 CONCLUSIONS

Adaptation is a demand to maintain the validity and quality of software over time. Adaptation is driven by certain categories of changes in the system internal state and its operational environment (security breaches, faults and errors, available resources, context, etc.). We advocated another driver for adaptation; the collective judgement of users on the alternative behaviours of a system. We proposed social adaptation to refer to the repetitive process of analysing it to choose upon the behaviour which is collectively judged to best fit a certain context. The main ingredient to get the collective judgement is the feedback of individual users. Feedback reflects users' main interest which is the validity and quality of a behaviour in meeting requirements. Feedback is influenced by certain contextual attributes which should be monitored when obtaining the users feedback. We have proposed and tested analysis mechanisms to infer the collective judgement on the validity and quality of different alternatives of a messenger system and discussed the results. We have also reported several observations on modelling requirements for social adaptation and presented them as research challenges to face.

Social adaptation is meant to enrich self-adaptation by accommodating users perception as a part of the system computation and their collective judgement as an adaptation driver. Our inspiring principle is the wisdom-of-crowds (Surowiecki, 2005) which keeps the door open for decisions formulated collectively by users. The power of the crowd is not easy to manage and social adaptation would work when users are wiling to collaborate with the system and moreover when the system already has implemented different alternatives and is running them. Social adaptation helps the system to tune the distribution of its alternative behaviours over the space of the different contexts of use. Social adaptation does not replace personalization; it only helps to know the collective judgement of users which is specifically useful for novice users as a recommendation system. It also helps developers to know what improvement and maintenance to do during the software lifetime.

As a future work, we are going to address the research challenges which we listed in Section 5. Mainly, we plan to enrich our modelling framework to capture constructs supporting a better judgement of the relevance of a feedback like the pattern of use and the learning history of a user. We also need to devise techniques to involve users in collectively taking harder decisions at runtime such as altering the requirements model itself by adding (removing) requirements, context and quality attributes. However, our main challenge is to find ways for balancing between users effort and computing transparency which is the essence of adaptive systems. We also need to work on incentives such as rewarding active users as a part of the social adaptation engineering. We need to further validate our approach on more complex systems and settings and develop a CASE tool to better facilitate social adaptation modelling and analysis.

ACKNOWLEDGEMENTS

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie) and by the European Union Seventh Framework Programme (FP7/2007-2013) under grants no 258109 (FastFix).

REFERENCES

Ali, R., Dalpiaz, F., and Giorgini, P. (2010). A goal-based framework for contextual requirements modeling and analysis. *Requir. Eng.*, 15:439–458.

Ali, R., Dalpiaz, F., Giorgini, P., and Souza, V. E. S. (2011a). Requirements evolution: from assumptions to reality. *In the 16th International Conference on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 11).*

Ali, R., Solis, C., Omoronyia, I., Salehie, M., and Nuseibeh, B. (2011b). Social adaptation: When software gives users a voice. Technical Report Lero-TR-2011-05, Lero. University of Limerick. Ireland.

Ali, R., Solis, C., Salehie, M., Omoronyia, I., Nuseibeh, B., and Maalej, W. (2011c). Social sensing: when users become monitors. In *Proceedings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, ESEC/FSE '11, pages 476–479. ACM.

Baresi, L., Pasquale, L., and Spoletini, P. (2010). Fuzzy goals for requirements-driven adaptation. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 125–134.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F., and Mylopoulos, J. (2004). Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236.

Cheng, B. H. C., Giese, H., Inverardi, P., Magee, J., and de Lemos, R. (2008). Software engineering for self-adaptive systems: A research road map. In *Software Engineering for Self-Adaptive Systems*, pages 1–26.

Dumas, J. S. and Redish, J. C. (1999). *A Practical Guide to Usability Testing*. Intellect Books, Exeter, UK, UK, 1st edition.

Fickas, S. and Feather, M. S. (1995). Requirements monitoring in dynamic environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, RE'95.

Hui, B., Liaskos, S., and Mylopoulos, J. (2003). Requirements analysis for customizable software goals-skills-preferences framework. In *Proceedings of the 11th IEEE International Conference on Requirements Engineering*, pages 117–126.

Laddaga, R. (1997). Self-adaptive software. Technical Report 98-12, DARPA BAA.

Murch, R. (2004). *Autonomic computing*. IBM Press.

Qureshi, N. A. and Perini, A. (2010). Requirements engineering for adaptive service based applications. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 108–111.

Salehie, M. and Tahvildari, L. (2009). Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4:14:1–14:42.

Silva Souza, V. E., Lapouchnian, A., Robinson, W. N., and Mylopoulos, J. (2011). Awareness requirements for adaptive systems. In *Proceeding of the 6th international symposium on Software engineering for adaptive and self-managing systems*, SEAMS'11, pages 60–69. ACM.

Surowiecki, J. (2005). *The Wisdom of Crowds*. Anchor.

Vredenberg, K., Isensee, S., and Righi, C. (2001). *User-Centered Design: An Integrated Approach*. Prentice Hall PTR.