Social Adaptation
When Software Gives Users a Voice

Raian Ali
Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Carlos Solis
Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Inah Omoronyia
Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Mazeiar Salehie
Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

Bashar Nuseibeh
Lero – The Irish Software Engineering Research Centre
University of Limerick, Ireland

November 2011

Contact

Address	Lero International Science Centre University of Limerick Ireland
Phone	+353 61 233799
Fax	+353 61 213036
E-Mail	info@lero.ie
Website	http://www.lero.ie/

This work is partially supported by Science Foundation Ireland

Lero Technical Report Lero-TR-2011-05

Technical Report Lero-TR-2011-05, The Irish Software Engineering Research Centre
University of Limerick, Ireland. November 2011

Social Adaptation

When Software Gives Users a Voice

Raian Ali, Carlos Solis, Inah Omoronyia, Mazeiar Salehie, Bashar Nuseibeh
Lero, University of Limerick, Ireland

Abstract Adaptation requires a system to monitor its environment so that when changes occur, a suitable adaptation action is planned and taken at runtime. The ultimate goal of adaptation is that users get their dynamic requirements met efficiently and correctly. Users are the primary decision makers about the correctness and the quality of adaptation and their feedback is primitive to continuously guide adaptation along the lifetime of a system. In this paper, we propose *social adaptation* which gives users a voice in planing adaptation. To this end, the system has to be designed to obtain and analyze users feedback repetitively and select upon the behavior which is socially judged to be the best for meeting requirements. Social adaptation treats users as system collaborators rather than pure consumers of system's functionalities so that users perception becomes an integral part of the computation of a system. We propose a requirements engineering modeling and analysis framework for systems built to enact social adaptation. Finally, we evaluate our framework on a case study of socially-adaptive messenger system 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 own state [1]. Each change can trigger a different response to satisfy or maintain the satisfaction of certain design objectives [2]. 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 [3].

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.). Arguably, there are drivers which are unmonitorable by relying on solely automated means as proposed by Ali et al in [4]. The judgment of users of the successfulness and quality of each of the system alternatives as a way to meet requirements, is an example of that. Such judgment is a primary driver for adaptation to decide upon what alternative to enact and maximize users' satisfaction. Hence, we advocate that social feedback is essential to give users a voice in planing adaptation. We define *Social Adaptation* as the system ability to obtain and analyze users' feedback and choose upon a system alternative which is socially judged to be the best to meet requirements in a specific context.

Social feedback can support a feasible and correct systems adaptation. This is particularly true for high-variability 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 [5,6]) 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 analyze such feedback and validate upon each system alternative. Thus, social adaptation helps to eval-

uate and adapt upon high-variability systems rapidly and effectively.

Social adaptation advocates the repetitive obtainment and analysis of social feedback to keep adaptation up-to-date. Due to the volatile nature of the world, users' judgment of the system could change over time. Thus, a one step design-time system validation and customization might lead to decisions which are correct but only temporarily. For example, users' who currently like interacting with the system 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 proved to be valid during the design-time validation may become eventually invalid.

Requirements are the main subject of social feedback and thus social adaptation is in the first place a requirements-driven adaptation. Upon the execution of a system alterative, 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. In other words, social feedback concerns the validity and the quality of a system alternative as a way to meet users' requirement. This judgment 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 certain context occurs, social adaptation will analyze the social feedback provided for each alternative when executed in that context and execute upon the one which is socially judged to suit that context.

In this paper, we propose social adaptation which enables users to be primary drivers for adaptation. It allows users to continually express their judgments about the the role of a system in meeting their requirements and allows the system to enact the users' choices. We discuss the foundations and motivation of social adaptation. We provide a conceptualization of the main artefacts needed in a requirements model for enabling social adaptation and discuss the use of goal models [7–10] to identify these artefacts. We develop analysis techniques to process social feedback at runtime and choose upon the system alternative which is socially shown to be appropriate to meet requirements. We evaluate our approach on a socially-adaptive messenger system.

The paper is structured as follows. In Section 2 we discuss the key differences between social adaptation and self-adaptation. In Section 3 we present an example scenario to be used along the paper. In Section 4 we discuss the theoretical foundations of social adaptation. In Section 5 we present the modelling artefacts of social adaptation and the use of goal models to extract these artefacts. In Section 6 we develop analysis algorithms to process social feedback and adapt the system at runtime. In Section 7 we evaluate our framework and discuss the results. In Section 8, we discuss our developed CASE tool. In Section 9 we discuss the related work and in Section 10 we present our conclusions.

2 Social Adaptation Vs. Self-Adaptation

Social adaptation is a distinguished kind of adaptation which responds to the social judgment about the correctness and efficiency of a system in meeting users' requirements. Social adaptation treats social feedback as a primitive driver for adaptation. Social feedback allows a runtime continuous evaluation of each of the alternative behaviours of a system and, thus, enables ranking the system alternatives. The alternative which is socially proved to be correct and more efficient will be the one to apply. In Figure 1 we outline the social adaptation loop 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.

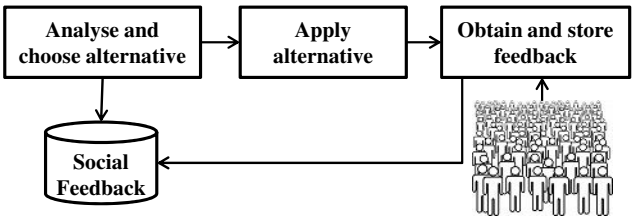

Fig. 1 Social Adaptation Loop

In what following we discuss the key differences between our proposed social adaptation and traditional self-adaptation which is surveyed in [2]:

- *The Monitored* which represents information which has to be monitored in order to decide upon a suitable adaptation action. In self-adaptation, a system has to monitor changes in its operational environment and its internal state. Monitored events and states could indicate security breaches which trigger a self-protecting action, or new settings of the available resources which trigger a self-optimizing action, or faults which trigger a self-healing action. In social adaptation, a system has to monitor the social judgment about its role in meeting users' requirements. Such judgments concern human's opinions and conclusions rather than events and states in the system internal and external environment.
- *The Monitor* which represents the entity that is responsible and capable of capturing the drivers of adaptation. 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 [11]. Social adaptation, on the other hand, requires capturing its own kind of drivers, the users' judgment, which is un-monitorable by relying on solely automated means [4]. Social adaptation requires a socio-technical monitor which involves users' perception as an integral part of the system computation.

- *The adaptation target* which stands for the subject and the goal of adaptation. Self-adaptation provides a dynamic machinery to ensure that certain system properties are maintained when changes happen. That is, the subject of adaptation is the system itself and the goal is to guarantee certain properties by preventing and acting against possible security breaches, faults, low performance, etc. On the other hand, social adaptation provides machinery to ensure that the system responds to the social judgement of each of its alternative behaviors. Thus, the subject of social adaptation is the users' judgement about the system and the goal is to maximize the possibility of having positive judgement about the role of the system in meeting their requirements.
- *The decision maker* which stands for the entity that takes adaptation decisions. In self-adaptation, decisions are planned by designers at design time so that adaptation is deterministic [12]. The system at runtime will monitor changes and apply a rationale developed by the designers and take an appropriate decision. In social adaptation, the decision makers are multiple. Designers plan adaptation at design time and leave the adaptation decision to be taken collaboratively by both users and systems. Users will interact with the system to provide their feedback and the system will analyse such feedback and adapt according to the rationale provided by the designers. Thus, adaptation is not determined by only designers' but it is also subject to users' decisions.
- *Evolution* which refers to the eventual changes in the adaptation rationale to keep adaptation up-to-date. Self-adaptive systems follow an evolution rationale, which is planned at design time, until the system apparently fails in taking correct adaptation decisions. When this happens, an evolution has to take place. To this end, the designers need to alter system and make an evolved version able to cope with the new situation. Besides the possibility of being subject to design time evolution, social adaptation incorporates runtime evolution driven by the users. A socially-adaptive system responds to the users' feedback which is itself likely to evolve over time, and thus the adaptation rationale itself evolves. In other words, the evolution of the adaptation rationale is driven by the evolution of users judgement about each system alternative with respect to meeting their requirements.

Table 1 Self-Adaptation Vs. Social Adaptation

Criteria	Self-Adaptation	Social-Adaptation
Monitored	States and events	Users' judgments
Monitor	System	System and users
Target	System properties	Users' satisfaction
Decision Maker	Designers	Designers and users
Evolution	Designers	Designers and users

3 Running Example

We consider the case of a gallery-guide system designed to support the visitors of an art gallery. The system is supposed to interact with visitors using PDAs given to them by the receptionist upon registration. The purpose of the system is to help visitors to ask and get information about the art pieces in the gallery. To this end, the system has to enable visitors of specifying an art piece (through RFID reader, say). Upon that, the system will convey information to visitors choosing autonomously between two alternatives:

1. *PDA-based system alternative*. The system will use the visitor's PDA to explain, possibly interactively, about the specified art piece.
2. *Staff-supported system alternative*. The system will try to find a mentor to interact with the visitor and explain the art piece. The system will also notify the visitor about the approximate time for a mentor to come to his place. It will also interact with the mentor and direct him to the visitor's location.

The designers are not fully certain of which alternative (PDA-based, Staff-supported) is more appropriate than the other and in which context. As a solution, the designers intend to build the system to obtain and benefit from the visitors feedback for taking such decision. This feedback will be obtained and analysed at runtime to choose the alternative that better fits each execution context. As a result, when a new visitor comes, the system will process the feedback obtained in the past from other visitors and select a system behavior that would fit the new user in the current context.

4 Social Adaptation: Foundations

In this section, we discuss the theoretical foundations of our proposed kind of adaptation, the social adaptation (sketched in Figure 2).

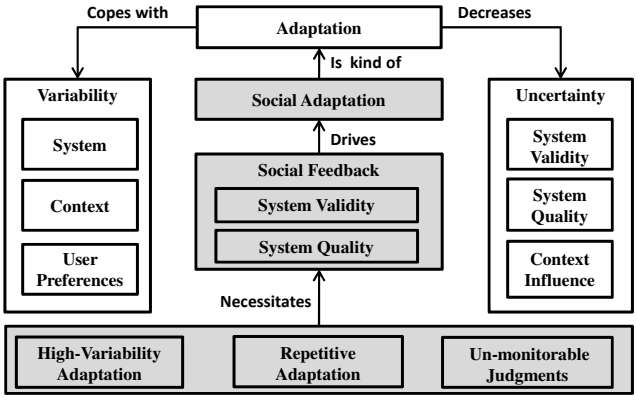

Fig. 2 Social Adaptation Foundations

4.1 Variability

The existence of variations is an essential characteristic that enables and also justifies adaptation. In our approach, we deal with variability concerning the state of the system environment (context [13]), the preferences the users of the system might have, and the alternative behaviors the system is provided with:

- *Context variability.* The variability of context necessitates adaptation. If context is static and its value is a priori known, the system can be also designed to act uniformly without planning any adaptation to its environment. For example, if the gallery-guide system is designed to work within a single gallery maintaining the same settings all the time and to interact with visitors having the same characteristics then the system can be adjusted, at design time, to fit to those static settings.
- *User preferences.* Users of the system might have different preferences which are another driver of adaptation. The system has to ensure compliance between its behavior and user's preferences. For example, a visitor might prefer "more comfortable" than "more informative" system alternative. Thus, the system could choose an alternative which sacrifices information richness for visitor's comfort.
- *System variability.* Adaptation to variable context and preferences requires providing the system with multiple alternatives. Adaptation means the ability of the system to select the alternative that fits each monitored context and enacted user's preference. For example, if the gallery-guide is designed to support only the PDA-based system alternative, then it may not be able to adapt to a context like "visitor has no technology expertise" or the preference of "visitor prefers minimum interaction with computing devices".

4.2 Uncertainty

The need for adaptation comes essentially from the inability to take certainly-true and/or infinitely-true design decisions. Designers might make assumptions at design time regarding the relation between the requirements, the system, and the world. These assumptions might be, or eventually become, invalid when the system operates in practice [14]. Adaptation is needed to cope with the validity/invalidity of assumptions at runtime. The uncertainty concerns mainly the decisions about:

- *System validity.* It concerns the uncertainty in predicting if a system alternative is indeed able to reach a specific requirement. For example, as a part of the Staff-supported system alternative, sending a voice message to a visitor telling the approximate time to meet a mentor is just a potential way to make the visitor aware of the time he has to wait. However, creating such awareness by executing such functionality is never certain as visitors may miss the message or misunderstand it. Validity could also evolve over time. For example, even if currently the voice message alternative is certainly valid, this does not mean it will be infinitely valid. When, in the future, the gallery-guide system issues more notifications to meet new requirements, the degree of visitor's concentration on each specific voice message might become lower and thus the alternative will become practically invalid.
- *System quality.* It concerns the decision about the degree of excellence of a system alternative. Quality could consist of different attributes and the evaluation of an alternative against each attribute can be uncertain. For example, taking user's comfort as a quality attribute, it is unpredictable if the PDA-based system alternative is more comfortable than the Staff-supported one. Moreover, this might evolve over time as people might become more comfortable with PDAs after a period of time or vice versa.
- *Context influence.* Context influences both the system validity and quality. While a design decision about this influence could be certain in some cases, it could be also uncertain in others. For example,

the design could presume that the validity of PDA-based system alternative certainly requires a context like "visitor has a good visual acuity" to hold. The design could also presume that a context like "PDA screen is big enough" is certainly supportive to "visitor comfort" perceived as a dimension for the quality of the PDA-based system alternative. On the other hand, some other influences of context could be uncertain. For example, the designer can only assume that "visitor's age" is relevant for both the validity and the quality attribute "visitor comfort" of the PDA-based system alternative but without being certain of the age ranges supportive to this validity and quality attribute.

4.3 Social Feedback

Users can judge and give feedback whether a system alternative leads to reach their requirements and its quality. We classify social feedback into 2 kinds:

- *System validity.* It concerns whether a certain applied system alternative succeeds in meeting a certain requirement. 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 would only give 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.* It concerns the degree of excellence of a system alternative. For example, a visitor who had to wait for long time to meet a mentor, would admit the delivery of information, i.e. the requirement is reached, but would probably say "it was not comfortable" (comfort is a quality attribute).

User-supplied feedback is a primitive driver for adaptation which is irreplaceable for the following reasons:

- *Un-monitorability of judgments.* A user's judgment of validity and quality is a human's opinion which is not always obtainable by relying solely on automated means [4]. For example, while the system can monitor if a requirement like "visitor is out of the gallery area when the gallery is to be closed" is met by sensing the visitor' location and movement, certain requirements cannot be monitored unless the system asks the users. The validity of the PDA-based system alternative with respect to meeting the requirement "visitor can ask for and get information" requires monitoring the visitor's judgement which is only possible if the user disclose it.
- *Repetitive adaptation.* As discussed before, certainty is not always achievable when designing a system. Moreover, certainty is not a static property. Thus, a one-stage validation of a system against its requirements may lead to certainty at only the time of validation. When time passes, what was valid at the validation time may become invalid and vice versa. Consequently, we need a repetitive validation and adaptation to ensure up-to-date fitness between the system and the requirements it is meant to meet. Moreover, planned iterative design-time validation would be inadequate as changes that influence the validity and quality of the system are inherently unpredictable. The solution is to enable this process as a repetitive automated activity at runtime. Users can give their feedback and adaptation can take place while the system is operating. For example, at an early stage of the system life, the users' feedback could indicate that PDA-based system alternative is not valid and that it has a poor quality. When time passes and visitors become more familiar with the use of PDAs, it might become a valid and good quality alternative. In the future and when a new communication technology is devised, this system alternative may turn back to be inappropriate.
- *High-variability adaptation.* Variability is a cornerstone for adaptivity as we discussed earlier in this section. Adapting highly-variable systems, which incorporate a large number of alternatives, necessitates asking the feedback of a large number of users. In traditional usability testing, user experience, and user-centric design, capturing users' feedback is usually performed at design time and involves a limited number of users. Applying these techniques on high-variability systems would not be feasible due to the number of users and cases and the amount of time needed to cover all possible alternatives and context variations. As a solution, social adaptation crowdsources users at runtime to give feedback about validity and quality of each executed system alternative. For example, each of the alternatives of the gallery-guide system (PDA-based and Staff-supported) is just a high level description of a system alternative. Each of these two alternatives incorporates a large number of finer-grained functionalities and variations and context influences as well. To enable a continuous and open-to-the-crowd feedback, visitors should be asked for and enabled to give their feedback at runtime. Analyzing and responding to feedback autonomously at runtime supports a rapid and efficient system adaptation.

5 Modelling Requirements for Social Adaptation

In this section, we explain the fundamental artefacts for the requirements models when designing systems to enact social adaptation. Figure 3 shows a model of these artefacts and Table 2 gives an example of an instance of this model. 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 and 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 apply that on a main-stream requirements model, goal model, in Section 5.1.

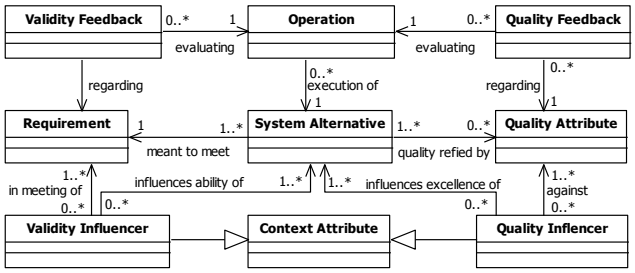

Fig. 3 Requirements modelling for Social Adaptation

The central part of the model captures the relation between the requirements and quality attributes from one side, and the designed system from the other. Requirement is a reification of a human desire to reach a 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 distinguished 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.

Example 1. The statement "visitor can ask for and get information about any art piece" is a requirement statement that indicates a visitor's desire to reach a certain level of awareness and knowledge about an art piece in the gallery. To meet this requirement, different system alternatives can be designed (PDA-based, Staff-supported). The Staff-supported alternative has technical and social counterparts. The technical part includes enabling the visitor to specify a product via RFID technology, alerting the mentor via his PDA, guiding the mentor to meet visitor in certain meeting points. The social part includes that the mentor interacts with the visitor and explain to him about the art piece. "Visitor's comfort" while delivering information about an art piece is a quality attribute against it each of the system alternatives can be evaluated.

The lower part of the model stands for the context influence on the relation between the system alternatives from one side and the requirements and quality attributes from 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.

Example 2. The context attributes "visitor's age" and "visitor's technology expertise level" are validity influencers which possibly affect the ability of PDA-based system alternative to enable visitor of querying and getting information. In other words, these attributes could influence whether a visitor finds PDA-based system alternative a valid means to meet his requirement of getting information about an art piece. The context attributes "the estimated time for a mentor to meet visitor", "the existence of a free seat close to the visitor's location" are context attributes that possibly affect the excellence of Staff-supported system alternative with respect to the quality attribute "visitor comfort".

The upper part of the model stands for the social feedback that reflects the users' judgment about each operation of a system alternative. An Operation is a single execution of a system alternative. Validity feedback is a Boolean judgment 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 by the system at runtime after an operation of that alternative is executed.

Example 3. An operation could start when a user specifies an art piece via RFID reader plugged in his PDA. Then the system chooses autonomously (based on analysing the historical feedback as we explain in Section 6) to follow a PDA-based system alternative and applies it. The correct execution of the functionalities this alternative incorporates (i.e. bug-free, no failure in network, etc.) does not mean that the requirement is reached. The visitor may not complete the interaction via his PDA or feel unable to interact with it, so the

Table 2 Instance of the model presented in Figure 3 and feedback example

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 mentor, etc.)
Quality Attributes	Q_1: visitor is well-informed Q_2: visitor's comfort
Validity Influencers	C_1: visitor's age and C_2: visitor's technology expertise level, influence the ability S_1 to meet R C_3: estimated time for a mentor to meet visitor and C_4: visitor has companions? influence the ability S_2 to meet R
Quality Influencers	C_1, C_2, and C_5: complexity of art piece information, influence S_1 quality vs. Q_1 C_1, C_2, and C_6: user's movement status (moving, standing, sitting), influence S_1 quality vs. Q_2 C_7: mentor's expertise and C_8: mentor's ability to speak the visitor's native language, influence S_2 quality vs. Q_1 C_3 and C_9: existence of a free seat closed to the visitor's location, influence S_2 quality vs. 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 S_2. The values of its relevant context attributes are C_3 = <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

visitor will give a negative validity feedback. Now, let us suppose that the system followed a Staff-supported system alternative. After completing the operation, a visitor could state that he got the information (the validity feedback is positive) but it was a bit uncomfortable to wait for a mentor and thus the quality attribute "visitor comfort" could be evaluated to "medium".

In this work, and to enable the forthcoming analysis, we presume that the values of both context attributes and the social feedback fall into a designated enumeration. The analyst should specify for each context attribute a set of values it may be assigned to. For example, the visitor age values set could be specified to "<18", "between 18 and 25", "between 25 and 65", ">65". The validity feedback is already set to a Boolean value {"requirement is reached", "requirement is not reached"}. The quality feedback value has to be taken from an ordered set of values. The designer may specify an integer range of values [0..n] where 0 reifies the user feedback "the system alternative has a very low quality with respect to the quality attribute q" and n stands for "the system alternative is excellent with respect to the quality attribute q". Alternatively, the range could be more expressive such as [low, medium, high] and [bad, acceptable, good, very good, excellent] to helps users to give feedback in their own terms.

5.1 Goal-driven social adaptation

In this section we discuss the use of goal model, a mainstream requirements model, for capturing the main artefacts of social adaptation (presented in Figure 3). Goal models (e.g., i* [7], Tropos [8,9], and KAOS [10]) represent an intentional ontology used at the early requirements analysis phase to explain the why of a software system. Goal modelling is a powerful technique to represent the rationale of both humans and software systems providing constructs to analyze their goals achieve a space of alternatives to satisfy these goals [15]. Such features are essential for the analysis and the design of systems adaptive to changes [16,17].

In Figure 4, we show an example of Tropos goal model [8,9]. It represents a part of our gallery-assistant running example explained in Section 3. Tropos goal analysis projects a system as a set of interdependent actors, each having its own strategic interests (goals). Actors may depend on each other for goals to achieve, tasks to execute, or resources to obtain. Goals represent requirements at the intentional level and are analysed iteratively, in a top-down way, to identify the more specific sub-goals needed for satisfying the higher-level goals. Goals can be ultimately satisfied by means of executable processes (tasks).

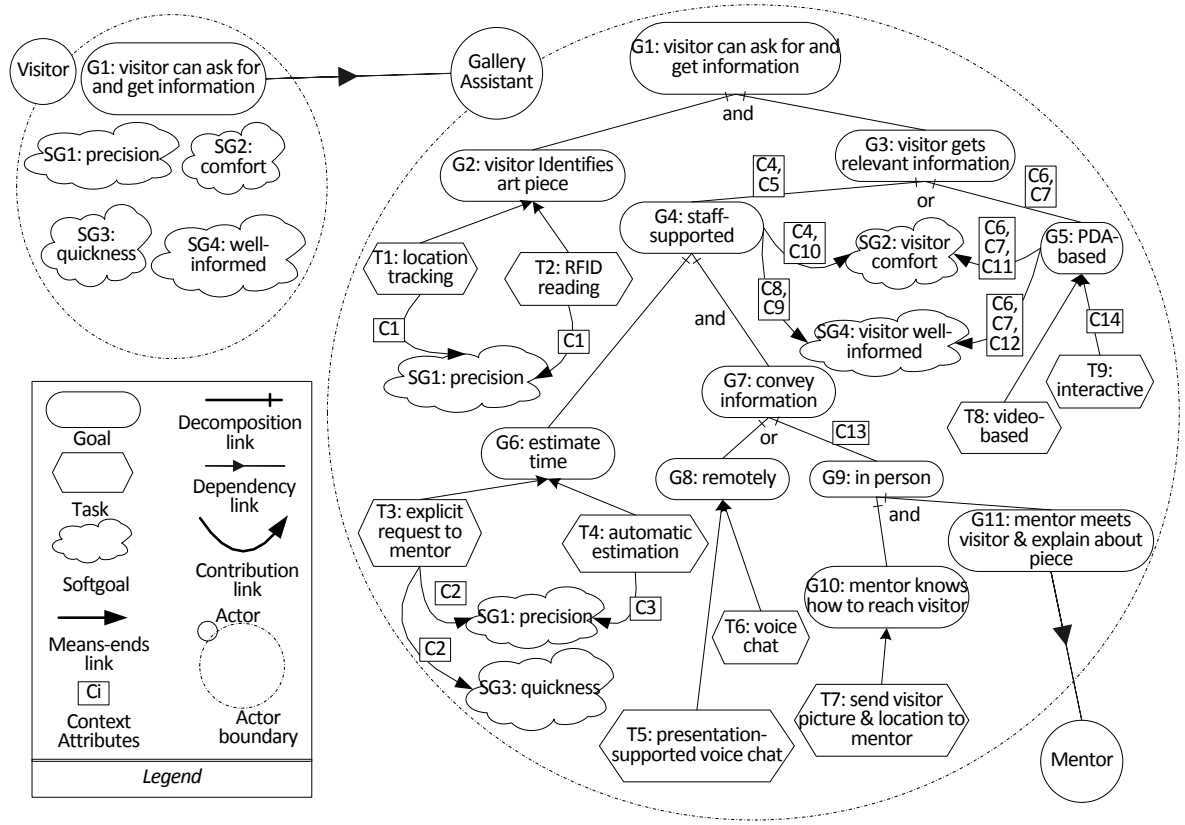

Fig. 4 Example of using Goal Model for Social Adaptation

The actor standing for the gallery-assistant system (Gallery Assistant) has the top-level goal G_1= "visitor can ask for and get information". Goals are iteratively decomposed into subgoals by AND-Decomposition (top goal is achieved when all sub-goals are achieved) and by OR-Decomposition (top goal is achieved when at least one sub-goal is achieved). The goal G_1 is AND-Decomposed into G_2= "visitor identifies the art piece" and G_3= "visitor gets relevant information"; the goal G_3= is OR-Decomposed into G_4= "Staff-supported" and G_5= "PDA-based". Goals are finally satisfied by means of executable tasks; the goal G_2= "visitor identifies the art piece" can be reached by one of the tasks T_1= "location tracking" and T_2= "RFID reading". A dependency indicates that an actor (depender) depends on another actor (dependee) to attain a goal or to execute a task or to get a resource; the actor "Visitor" depends on the actor "Gallery Assistant" for achieving the goal G_1. The goal model represents multiple alternatives that the system may adopt to reach the visitors' main goal G_1. Soft-goals are qualitative objectives for whose satisfaction there is no clear cut criteria (e.g., visitor comfort). Softgoals can represent quality attributes upon which the system alternatives for reaching goals (requirements) are evaluated.

We add to the model context annotations ($C_1 \ldots C_{14}$) which are described in Table 3. As discussed earlier, context potentially influence the validity and the quality of system alternatives. For example, C_{13}= "mentor familiarity with the gallery" possibly influence the validity of G_9="in person" alternative to reach G_7. A mentor, who is still new, might not be able to reach the location of visitor on time. C_1= "distance between art pieces" might influence the SG_1="precision" quality attribute of the RFID reading. That is, if the art pieces are very close and the visitors are not familiar with RFID, then a wrong art piece might be identified.

In the following we list two main guidelines for using goal models to identify social adaptation artefacts taking examples from Figure 4:

1. *Requirements and quality attributes.* Requirements are perceived as goals in goal model. Users of the system may have different goals they expect the system to reach (G_1). Each of these goals corresponds to a set of quality attributes perceived as softgoals (G_1 corresponds to SG_1, SG_2, SG_3, SG_4). Each of the goals delegated to the system actor (Gallery Assistant) has to be iteratively analysed to reach tasks (executable processes) which the system can execute (T_1, T_2, ..., T_9), or goals and tasks the sys-

Table 3 Contexts in Tropos model of Figure 4

C1	distance between art pieces
C2	amount the pending activities the mentor has to do
C3	mentor movement status
C4	estimated time for a mentor to meet visitor
C5	visitor has companions?
C6	visitor's age
C7	visitor's technology expertise level
C8	mentor's expertise
C9	mentor's ability to speak the visitor's native language
C10	existence of a free seat closed to the visitor's location
C11	user's movement status
C12	complexity of art piece information
C13	mentor's familiarity with gallery
C14	PDA has touch screen?

tem actor can delegate to other actors involved in the socio-technical system (G_{11}). The result of this analysis incorporates a space of alternatives to reach the analysed goals. For example $\{T_1, T_3, T_7\}$ is a system alternative to reach G_1. The main difference between traditional modelling of requirements and modelling requirements for social adaptive systems is the consideration of *uncertainty* discussed earlier in Section 4. In traditional goal modelling, contributions to softgoals, i.e., the assessments of the quality of system alternatives, are a priori known at design time and, moreover, each alternative is presumed by designers a valid means to reach the analysed goals. However, this is not always achievable and designers' could be uncertain regarding the judgment of validity and quality of each alternative to reach goals. To deal with such uncertainty, social adaptation relies on the users' judgments provided as feedback after using each alternative in practice. That is, users become collaborators in the engineering of the requirements model and certain decisions are left for users instead of being fully taken by designers.

2. *Context influence.* As discussed in Section 4, context influences the validity of each system alternative and its quality assessment against each quality attribute. Goal model allows for a hierarchical refinement of goals to ultimately identify a space of system alternatives (e.g., $\{T_1, T_8\}$, $\{T_1, T_4, T_6\}$). This makes it easier to discover what context attributes influence the validity and the quality of those system alternatives in an iterative way as well. After each refinement in the goal model, the designers can identify the context attributes which influence the validity of the alternatives emerging at that refinement stage (C_4, C_5, and C_6, C_7, and C_{13}, and C_{14}) and their quality against the softgoals the user is concerned with (C_4, C_{10}, and C_8, C_9, and C_6, C_7, etc.). The designers have then to specify an enumeration specifying the values each context attribute might take. The difference between specifying context in social adaptation and specifying it in contextual goal models [18] is again the *uncertainty* of the relevance and the influence of context. For example C_3= "mentor movement" values could fall in sitting, standing, walking. The relevance of this context attribute and how each of its values influences the quality of all system alternatives containing T_4 against the softgoal SG_1 will be decided by the users at runtime. The reason is that the uncertainty of designers about the relevance of this attribute and about the influence each of its values has on the quality attribute "precision".

In Figure 5, we present examples of system alternatives taken from Figure 4 and show their validity and quality influencers.

Requirements={G_1}
Quality Attributes= {SG_1,SG_2,SG_3,SG_4}
Users= {visitor}

System Alternatives examples:

Example 1. S_1= {T_1, T_3, T_7}
Validity Influencers= {C_4, C_5, C_13}
Quality Influencers of S_1
    against SG_1={C_1, C_2},
    against SG_2={C_4, C_10},
    gainst SG_3={C_2},
    against SG_4={C_8, C_9}.

Example 2. S_2= {T_2, T_9}
Validity Influencers= {C_6, C_7, C_14}
Quality Influencers of S_2
    against SG_1={C_1},
    against SG_2={C_6, C_7, C_11},
    against SG_3=$\emptyset$,
    against SG_4={C_6, C_7, C_12}.

Fig. 5 Examples of system alternatives taken from Figure 4

6 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 to overcome the designers' uncertainty about this decision. When the system has to meet a requirement, it has to choose an alternative to apply. The system has to evaluate the probability of each alternative of being a valid and a good-quality means to meet that requirement. We propose to take into consideration different factors that together help for a holistic assessment of the validity and quality of a system alternative. In what following, we discuss these factors taking examples of Table 2:

- *Feedback value.* This factor stands for the values the users gave when evaluating the validity and the quality of each operation of a system alternative. Users provide the validity feedback upon each operation of a system alternative by giving a Boolean answer reflecting their judgment 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' judgment of the validity and quality of a system alternative. We consider two sub-factors which influence the relevance of a feedback:
  - *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", 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 the 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 the 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. As we mentioned earlier, we take into consideration the feedback value and relevance (context match and freshness factors). 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 [19], we allow users to express their preferences by ranking the degree of importance of each of the quality attributes. For example, suppose that by analyzing 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 the PDA-based alternative overall quality will be assessed higher than the overall quality of the Staff-supported system alternative, and vice versa.

The algorithm Assessing Validity (Figure 6) computes the validity probability 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 the probability of the statement "s is a valid means to meet the requirement it is designed for". The algorithm identifies first 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

```
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 := ∅
 9.   For each Ci | 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_validity_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
```

Fig. 6 Assessing Validity Algorithm

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 $|\ Op.C_i\ |$ (Line 12). The relevance of this probability is decided by both the cardinality of context match (i/$|\ C\ |$) 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).

The algorithm Assessing Quality (Figure 7) computes the quality of a system alternative against a quality attribute based on the quality feedback provided by users who used that system alternative. It takes as input a system alternative s and a quality attribute q, the set of quality influencers C which affect the quality of s with respect to q and the values of these influencers at the time of assessment C.values. The rationale of this algorithm is similar to the algorithm Assessing Validity. The main difference is that the algorithm considers only the operations where the validity feedback was positive as negative validity feedback makes any quality feedback irrelevant (Lines 1). Moreover, the algorithm deals with the average value of the quality feedback provided for the alternative s against the quality attribute q (Line 3, Line 12). As we mentioned earlier, the quality feedback is represented via a numeric value in an integer range [0..n] specified by designers where the value is proportional to the satisfaction of users with respect to that quality attribute.

In Figure 8, we present examples of the user feedback concerning the example shown in Table 2 and then show how both algorithms (Assessing Validity and Assessing Quality) process feedback to evaluate the validity and quality of a system alternative.

Users generally have more than one quality attribute to evaluate the degree of excellence of a system alternative. When this happens, the system needs to assess the overall quality of a system alternative. Our approach is based on allowing users to express their judgment of the importance of each quality attribute. This way of capturing preferences is proposed in [19]. The degree of importance is an integer in a range [0..n] where 0 represents the user statement "It is not important at all", and n represents "it is highly important". The overall quality of a system alternative s is thus com-

```
Input: s: System alternative
       q: Quality attribute of s
       C: {c, c is a quality influencer of (s,q)}
       C.Values: {(c,v), c in C and v = c's monitored value}
Output:  assessed quality of (s,q)

 1. OP:= {o ∈ s.Operations| o.validity_feedback= ''valid'' and o got quality feedback for q}
 2. If |C| = 0 then
 3.  RETURN Avg_Freshness(OP)*Avg({quality_feedback(o,q); o in Op})
 4. Else
 5.  relevant_quality_sum:= 0
 6.  relevance_sum:=0
 7.  For i = 1 to |C| Do
 8.   OP_Ci := ∅
 9.   For each Ci | Ci ∈ 2^C and |Ci|= i
10.    OP_Ci = OP_Ci U {o in Op; ExactMatch(o.Ci.Values, C.Values)}
11.   EndFor
12.   quality_Ci:= Avg{quality_feedback(o,q); o in OP.Ci}
13.   relevance_Ci := (i / |C| + Avg_Freshness(OP.Ci))/2
14.   relevant_quality_Ci := relevance_Ci * quality_Ci
15.   relevant_quality_sum:= relevant_quality_sum + relevant_quality_Ci
16.   relevance_sum:= relevance_sum + relevance_Ci
17.  EndFor
18.  RETURN relevant_quality_sum/relevance_sum
19. EndIf
```

Fig. 7 Assessing Quality Algorithm

puted by the following equation, where Q is the set of quality attributes relevant to the system alternative s, Assessed_Quality(s,q) is the result of applying the algorithm Assessing Quality to evaluate the quality of s against a quality attribute q, and Importance (q) is the degree of importance of q to user:

$$\frac{\sum_{q \in Q} Importance(q) * AssesedQuality(s,q)}{\sum_{q \in Q} Importance(q)}$$

Social adaptation relies on the collective judgment of a community of users. This requires having enough feedback for each system alternative in order to take correct social adaptation decisions. To this end, and when the system is to be initiated, the system designers might adopt strategies like (i) benefiting from instances of the same system which operated in similar environments and analysing their feedback, or (ii) allowing a trial period in which the system aims to apply all its alternatives to reach a certain threshold of feedback. Another question relates to the decision about the alternative to adopt, after assessing the validity and the overall quality of each alternative. One design option could divide validity percentage into ranges and consider all alternatives whose validities fall in the same range equally valid. Then the differentiation between the equally valid alternatives will be based on the overall quality assessment of each alternative. Another design option could follow a different rationale by deriving an overall assessment for each alternative reflecting both validity and quality. Such option could give a certain weigh to validity and another for quality reflecting their importance when calculating the overall assessment.

7 Evaluation

In this section we evaluate our approach via a socially-adaptive messenger system. The system is adaptive in terms of delivering the message to the receivers using the best way in each different context. To this end, the system will gather feedback from messages receivers and reflect their judgment of the validity and the quality of each alternative for delivering messages over time. The evaluation concerns mainly two aspects:

1. *Modelling requirements for social adaptation.* In which we report on the observations made upon modelling the requirements of the messenger system. We have adopted goal models as a requirement model and the modelling activity followed the guidelines we have discussed in Section 5.1.
2. *Social adaptation analysis.* In which we evaluate the analysis explained in Section 6. We evaluate whether the analysis supports the system to choose alternatives which maximize the validity and quality judgments of messages receivers.

S_2	Staff-supported alternative
S_2**.Op**	operations of S_2
VF	Validity Feedback
VI	Validity Influencers
QF	Quality Feedback (scale [0,1,2,3,4])
QI	Quality Influencers
Q1	visitor is well informed
C_3	estimated time for a mentor to meet visitor in minutes { $\leq$5, >5}
C_4	Visitor has companions? {yes, no}
C_7	mentor's expertise {low, medium, high}
C_8	mentor's ability to speak visitor's language {low, medium, high}

S2.Op	**VF**	**VI**	**QF(S2,Q1)**	**QI (S1,Q1)**
1	False	C_3: $\leq$5, C_4: no	N/A	N/A
2	True	C_3: >5, C_4= no	2	C_7= medium, C_8= high
3	True	C_3: $\leq$5, C_4= no	4	C_7= high, C_8= high
4	False	C_3: >5, C_4= no	N/A	N/A
5	True	C_3: $\leq$5, C_4= yes	3	C_7= high , C_8= medium
6	True	C_3: $\leq$5, C_4= no	1	C_7= low, C_8= high
7	True	C_3: >5, C_4= no	0	C_7= low, C_8= medium

Assessing Validity (S_2,{C_3,C_4}, {C_3: $\leq$5, C_4:no})

i=1

OP_C_i={2,4,5,7}, Valid(OP_C_i)={2,5,7}, validity_C_i=3/4=0.75
relevance_C_i= ((1/2)+ ((2+4+5+7)/4)/7)/2=0.57
relevant_validity_C_i= 0.57*0.75=0.43
relevant_validty_sum=0.43, relevance_sum= 0.57

i=2

OP_C_i={1,3,6}, Valid(OP_C_i)={3,6}, validity_C_i=2/3=0.67
relevance_C_i= ((2/2)+ ((1+3+6)/3)/7)/2=0.74
relevant_validity_C_i= 0.74*0.67=0.5
relevant_validty_sum=0.43+0.5=0.93, relevance_sum= 0.57+0.74=1.31

RETURN 2.01/1.47 =1.37 (quality scale is [0..4], the quality is 1.47/4 =36.75 %)

Assessing Quality(S_2,Q_1,{C_7,C_8}, {C_7:low, C_8:high})

i=1

OP_C_i={2,3,7}, avg_quality_C_i=(2+4+0)/3=2
relevance_C_i= ((1/2)+((2+3+7)/3)/7)/2=0.54
relevant_quality_C_i= 0.54*2=1.08
relevant_quality_sum=1.08, relevance_sum= 0.54

i=2

OP_C_i={6}, avg_quality_C_i=(1)/1=1
relevance_C_i= ((2/2)+((6)/1)/7)/2=0.93
relevant_quality_C_i= 0.93*1=0.93
relevant_quality_sum=1.08+0.93=2.01, relevance_sum=0.54+0.93

RETURN 2.01/1.47 =1.37 (quality scale is [0..4], the quality is 1.47/4 =36.75 %))

Fig. 8 Example of Assessing Validity and Quality based on Table 2

The messenger system should be provided with alternatives to deliver messages to users (messages receivers) to enable adaptation. Adaptation is perceived as the selection between alternatives so that the messages receivers are satisfied. When designing the system, the designers are not certain about the most valid and the best quality alternative to apply. Thus, the designers will leave the decision to users themselves. To this end, users will be asked to provide feedback and the system will analyze this social feedback and choose upon the best alternative to apply. Mainly, the system has two strategies to deliver the incoming messages; instant messaging and offline messaging. By instant messaging we mean that the system, upon receiving a new message, will notify the user and show the message and enable the user to reply to it. The ways to notify the user and display the message and reply to it are various and each variation could be subject to a different judgment of validity and quality. By offline messaging we mean that the system will store the message so that the user can view it and replay to it in the time he wants.

7.1 Modelling requirements for social adaptation

To evaluate the modelling part of our framework, we have organized a lab session and invited 5 researchers with good expertise in requirements modelling and goal models. We have explained our design principles of modelling requirements for socially-adaptive systems and

guidelines of using goal models, namely Tropos goal model [8], for such purpose. We then explained the scenario of the messenger system and asked the participants to draw a goal model presenting its requirements. The purpose of the session is to find out relevant concerns and limitations of our proposed framework at the modeling stage. The session was interactive and we documented our observations as well as the concerns the participants have explicitly raised. Figure 9 shows one of the goal models built during the lab session. We here summarize the main observations made upon this experiment:

- *Context monitorability.* Besides the fact that the judgement made by users about the validity and the quality of system alternatives is generally un-monitorable by automated means, some contexts could be un-monitorable as well. As a solution, we may need to enable users of sensing context attributes which are un-monitorable by the use of only technological devices. For example, C_1 which stands for the context "user is involved in other communications" is not fully monitorable by automated means. The user could be having in-person chat with somebody around or using a communication device different from the one on which the messenger is installed. Other context could concern also judgements, not only environment status and events, which are un-monitorable as well, e.g., "the user is very interested about the received message". However, involving users heavily in monitoring context could compromise computers transparency [20] which is the main goal of adaptation. We still need approaches to guarantee precision when users have to supply context values and, in the same time, to minimize the users' interaction with the system so that the computing transparency is not highly compromised.
- *Quality attributes identification.* The designers' decisions on which quality attributes are relevant and whether the set of specified attributes are sufficient to capture all relevant dimensions of system quality could be uncertain. Moreover, and due to the fuzzy nature of quality attributes semantic, some designers might find one attribute synonym to, or partial of, another. For example, there could be more attributes than "less distraction", "less effort", "readability" relevant for the quality of each system alternative in the messenger system. Moreover, an attribute like "easy to use" and "comfortable" might be seen as synonyms for one designer while "comfortable" might be seen as part of "easy to use" for another. As a solution for the inherent difficulty of reaching a certain decision and consensus about quality attributes relevance and semantics, we plan to leave this decision to users themselves. That is, designers might define an initial set of quality attributes and users might alter as well. The attributes which survive are the ones users judge often and consider relevant. In such a way, users are the main decision makers even in the identification of quality attributes not only the judgment of the quality of system alternatives.
- *Context influence (un)certainty.* The influence of context attributes values on the validity and the quality of each system alternative is not always uncertain as we studied in this paper. We presumed that designers define context attributes which have such influence without being certain about the exact influence of their values and leave this decision for the analysis done by the system at runtime as we explained in Section 6. However, designers could be certain of the influence of some context values and we do not need to wait user feedback in order to infer that influence. For example, for interactive communication with messages receivers who are using mobile devices, a context attribute like "touch screen?" influences the quality of this alternative against the quality attribute "fast". Designer might say that having a touch screen is certainly supportive to this quality attribute while not having it has a negative influence and there will be no need to wait for social feedback to be sure of that. In [18], Ali et al. study modelling and analysing context influences on the validity and quality under certainty. We still need to integrate that work and our current work for a more holistic treatment of context influence on the validity and quality of requirements-driven adaptation.
- *Evolving the requirements model.* In our framework, designers are involved in taking decisions under uncertainty and often on subjective basis. Adaptation has to evolve the model to cope with reality and overcome uncertainty and subjectivity [14]. The refinement in the requirements model is an example of uncertain design decisions. When we refine the requirements either by decomposition or specialization (corresponding to AND-Decomposition and OR-Decomposition in the goal models respectively), we are just assuming implication relation stating that reaching all (one of) sub-requirements will lead to reaching the refined requirement. Social adaptation is the way we presented to validate such assumptions based on social feedback, i.e., to evolve the validity and the quality of system alternatives over time. We still need to develop more automated analysis to evolve the model for different other reasons. For example, we need the system

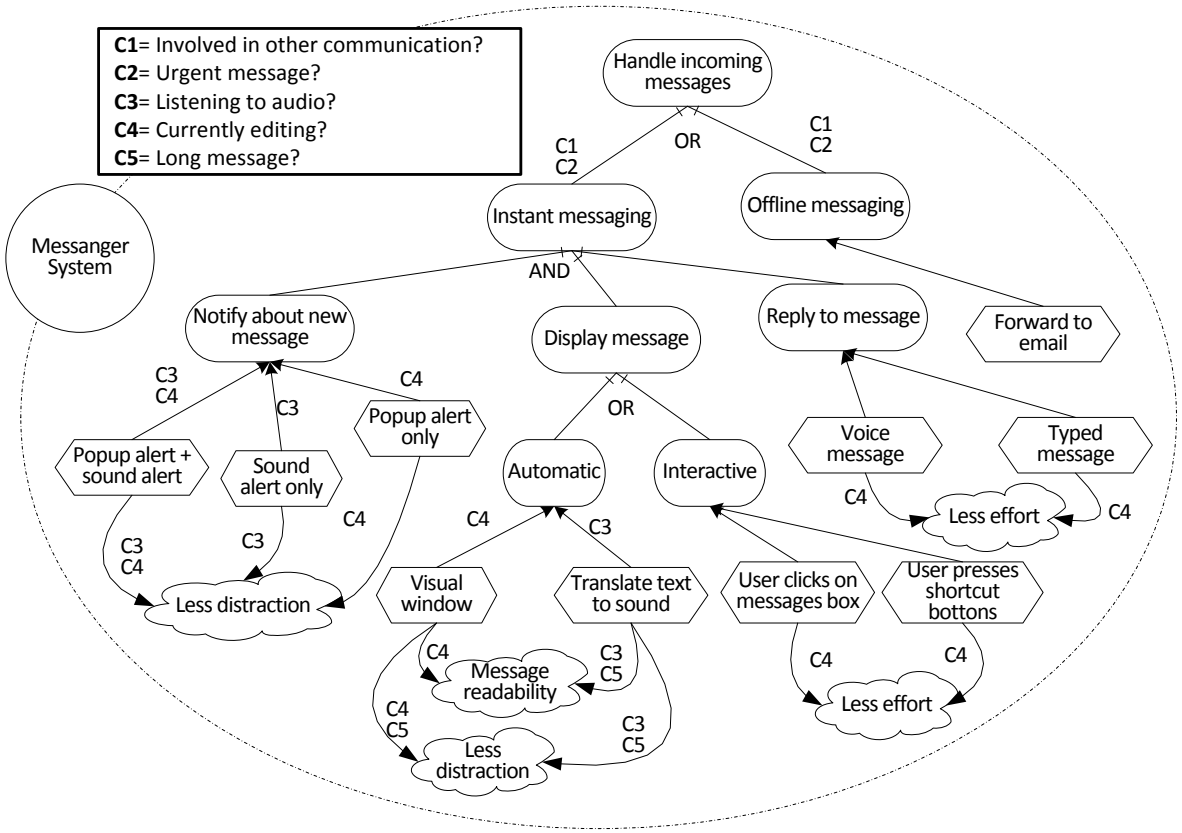

Fig. 9 A Goal Model developed for the Socially-adaptive Messenger

to identify removable alternatives which were shown extremely invalid (low quality) by users and, moreover, to identify loci where the model lacks reasonably valid and/or acceptable quality alternatives. On the other hand, specifying how the system will behave at the initial period of operation to get users' feedback for all system alternatives is another decision to be taken under uncertainty in our current approach. To help for less involvement of designers and more flexible decision, we need to devise automated techniques that take this decision at runtime aiming to balance between the need for fast adaptation and the need for having feedback for all alternatives.

– *Feedback relevance.* In our framework, we do still do not preprocess users feedback and qualify which feedback are significant and trustworthy. Several factors play a role in this decision such as the pattern of use of the user and the consistency of users feedback. This, will avoid us noise in the information obtained via users feedback and elect the ones which really refelect users' judgment of systems alternatives.
– *Automated support for modelling.* When requirements model gets bigger, error-free modelling becomes harder to achieve without an automated support. Even in the small-size model shown in Figure 9, it was not easy to track context definitions, e.g. a context was defined twice for the validity/quality of the same alternative. The automated support is also needed to simulate the system behaviour and deriving alternatives so the designers can comprehend the model better and potentially enhance its quality.

Overall, we still need to enrich our modelling framework in order to minimize the subjective nature of design decisions, help for consensus between designers when conflict arises, and increase the automated support for modelling requirements and evolving the requirements model. Moreover, we need to devise techniques to involve users in taking more design decisions on voluntary basis and treat users as collaborators in the engineering of software rather than consumers of its functionalities. However, these techniques need to balance between the extra effort required from users and the desired computers transparency which is the essence of adaptive systems.

7.2 Validating social adaptation analysis

To evaluate the social adaptation analysis proposed in Section 6, 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 different alternatives in different contexts taking as input the set of obtained feedback. To evaluate the correctness of the automated analysis, we have surveyed a set of 3 other users (testing users) and compared their feedback to the results the automated analysis has reported.

To evaluate the validity assessment analysis, we have considered 3 alternatives and 3 different contexts where the validity assessment 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 judgment 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. As a base for comparison, we have ranked the quality assessment of automated analysis and the feedback of each user in 2 ways (i) the rank of the different alternatives for each context, and (ii) the rank of the same alternative in the different contexts.

Table 4, follows the first way of 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", 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" 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 judgment computed by the quality assessment algorithm and the testing users feedback was good enough (21 out of 27).

Table 5 follows the second way of ranking. For example, taking the first row of data, the automated analysis gave the alternative A_1 quality the best rank in the context C_3, a lower rank in the context C_2, and the lowest rank in the context C_1. The users U_1 and U_3 matched the system ranking while the user U_2 quality feedback was different as U_2 considered A_1 quality the best when operating in C_2, and the medium when operating in C_3, and the lowest when operating in C_1. Again, and according to this matching, the collective quality judgment computed by the algorithm we explained in Section 6 matched to a good extent the testing users feedback (21 out of 27). For the other quality attribute Q_2= "less effort", the number os matches between the automated analysis and the testing users was also good. It was 18 matches out of 27 comparison following the first way of ranking and 20 matches out of 27 comparison following the second way of ranking.

7.3 Threats to validity

The first threat to validity is the small size scenario we have used (the messenger system). The size is small in terms of the number of alternatives to reach the main requirement which is the handling of incoming messages (the root goal in goal model of Figure 9) , and the number of context influencers and quality attributes as well. This small size scenario helped us to discover several limitations and concerns at the modelling level and also to evaluate our social adaptation analysis. However, more complex systems would even reveal further concerns related to requirements engineering such as elicitation of requirements, viewpoints and conflicts, model consistency and so on. These challenges are potentially maximized when dealing with requirements for social adaptation which incorporates new elements such as the design uncertainty and the runtime analysis and the role of user in decision making at runtime. Moreover, analysing complex models might raise other challenges related to the computation complexity of the automated analysis (the space of alternatives grows exponentially with hierarchies like goal models). Other problem relates to the feasibility of trying all system alternatives in all context variations and obtaining feedback within a reasonable time. Our claim

Table 4 For each context, the rank of the different alternatives (mismatches are in bold font)

Q_1	C1				C2				C3			
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 5 For each alternative, the rank in the different contexts (mismatches are in bold font)

Q_1	C1				C2				C3			
LD	Sys	U_1	U_2	U_3	Sys	U_1	U_2	U_3	Sys	U_1	U_2	U_3
A_1	3	3	3	3	2	2	**1**	2	1	1	**2**	1
A_2	3	3	**2**	3	2	2	**3**	**1**	1	1	1	**2**
A_3	3	3	3	3	2	2	2	2	1	1	1	1

that the-openness-to-crowd will help to accelerate this coverage but our experiment size did not allow us to validate this claim on large-scale system.

Another threat to validity is the kind of participants in the lab session made for modelling requirements for social adaptation. The participants are academic researchers who have already good expertise in requirements engineering and particularly the use of goal models as a way to identify requirements. That is to say, we did not need to train the participants and our framework was easily understood and applied. However, communicating the principles and guidelines to novice practitioners might raise other concerns and limitations related to the understandability and the acceptability of our framework.

The users who used the prototype messenger system have committed to give a feedback for each operation. Moreover, our system generated messages and handled them via each system alternative and then asked the users to give their feedback (the exploratory phase of the experiment). That is, the users were aware of the purpose of the experiment and committed to collaborate. However, in real world settings, users might refuse to give feedback or may give incomplete and even inconsistent feedback which severely influence the applicability and the quality of our social analysis.

The experiment period was short and this fact has several implications. First, within a short period of time, the freshness factor does not have a real impact on assessing the relevance of feedback. We cannot expect users' to change their judgement of a system alternative radically in a week time period especially for the kind of system we used during the experiment (the messenger system). Moreover, the short experiment period does not allow us to analyse the system behaviour in a phase where the trend of users changes, i.e., when the collective feedback changes largely. When this happens, the system adaptation might be slower than expected so that the whole applicability of the system is compromised.

8 Automated Support Tool

We have implemented a tool for specifying goal models which conform with our metamodel of Fig.3. The goal of our tool is to support designers in modelling requirements for social adaptation proposed in Sec. 5 and performing the analysis proposed in Sec. 6.

The metamodel was defined using the Eclipse Modeling Framework (EMF)[1], which permits to define metamodels using the Ecore modeling language. EMF is a modelling language that is a subset of the Meta-Object Facility (MOF). MOF is a reflective meta-modeling framework which has three layers and it is a standard of OMG. The top layer is called M3, in M3 is located the MOF metamodel, which provides a domain specific language (concepts and relationships) for defining other metamodels. MOF is a reflective metamodel, i.e., MOF is defined using MOF, therefore MOF is an instance of MOF. All the other metamodels are defined using a common language which is MOF, and they are instances of it. These metamodels are located in the M2 layer, and the instances of the M2 metamodels are located in M1.

The Ecore concepts and relationships are based on those found in MOF. In the case of EMF, the top layer of the reflective tower M3 is located the Ecore metamodel. The user defined metamodels are instances of the Ecore metamodel, these models are located in M2, and the instances of the user metamodel are located in M1. The user defined metamodels in M2 and their instance can be defined are defined using generic tree metamodel and model editors which are part of the framework. The EMF.Edit framework includes generic reusable classes for building editors for EMF models.

[1] http://eclipse.org/modeling/emf/

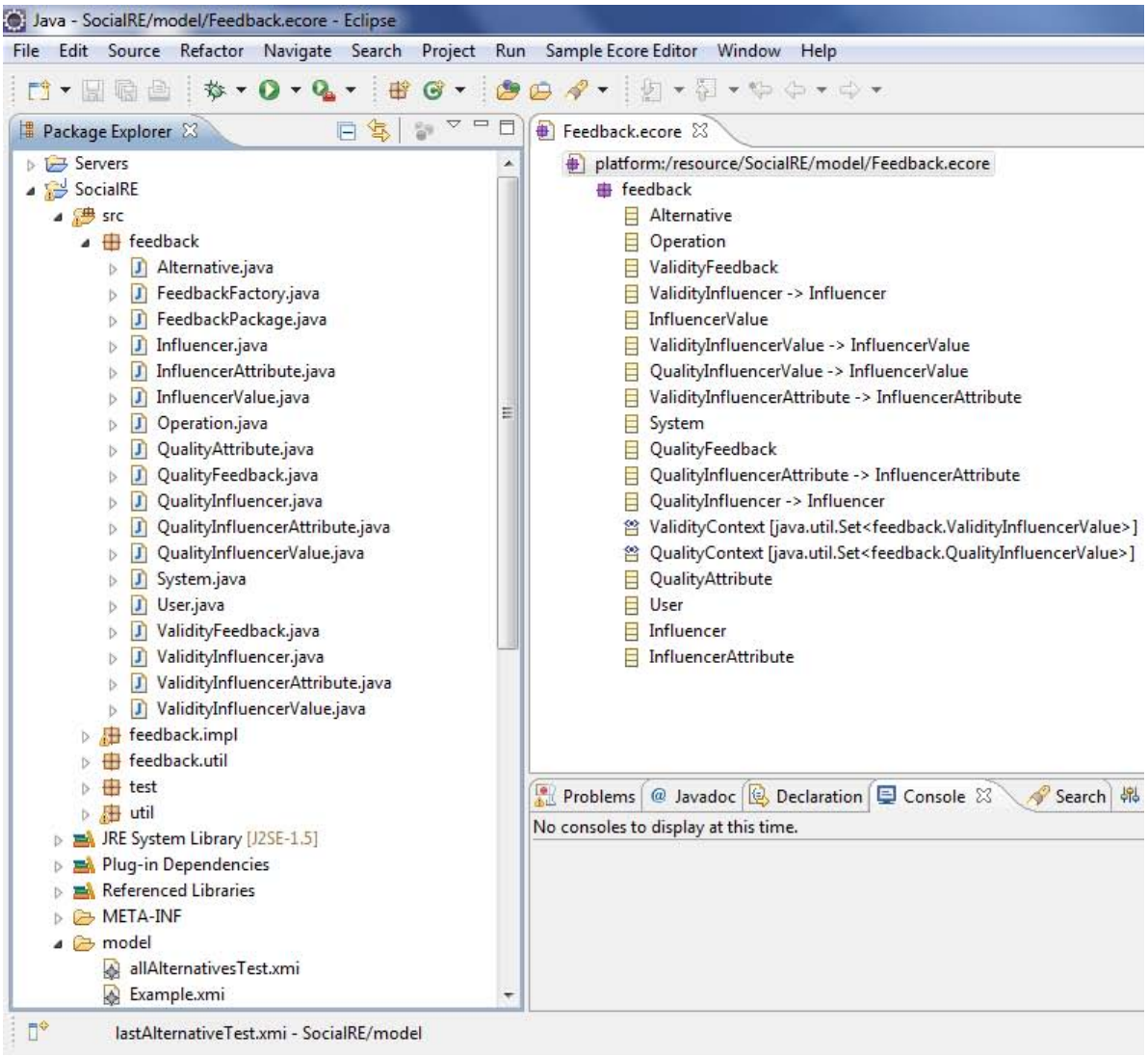

Fig. 10 A metamodel editor in EMF

Figure 10, presents an example of a metamodel editor in EMF.

EMF also generates java source code for representing, persisting and manipulating those models. The EMF.Codegen code generation framework can generate Java interfaces and implementation classes for all the classes in the model, and a factory implementation class. In addition, the generated metamodel classes are instances of the Ecore implementation classes, therefore, the source code is also reflective.

Using the EMF model editor, we can define a goal model metamodel which is able to represent goal models with any number of alternatives, quality attributes, quality influencers, and validity influencers. The goal model metamodel is an instance of Ecore, therefore it is located in M2 metamodel.

Figure 11 presents a snapshot of the goal model metamodel definition editor. In addition, we can observe some of the concepts that are part of our metamodel, such as alternative, validity influencer, operation, etc. Our metamodel representation also includes feedback classes; therefore a model can contain the feedback provided by users of the system.

Using EMF, we generated the model editor of the goal model metamodel, the model editor allows us to define specific goal models. In the model editor we can specify which validity and quality influencers affect a given alternative. Figure 12 presents an example of a specific instance of our generic goal model metamodel. The instances of our goal model metamodel are located in the M3 layer of EMF.

Using the EMF generated source code, we have implemented our proposed algorithm. This implementa-

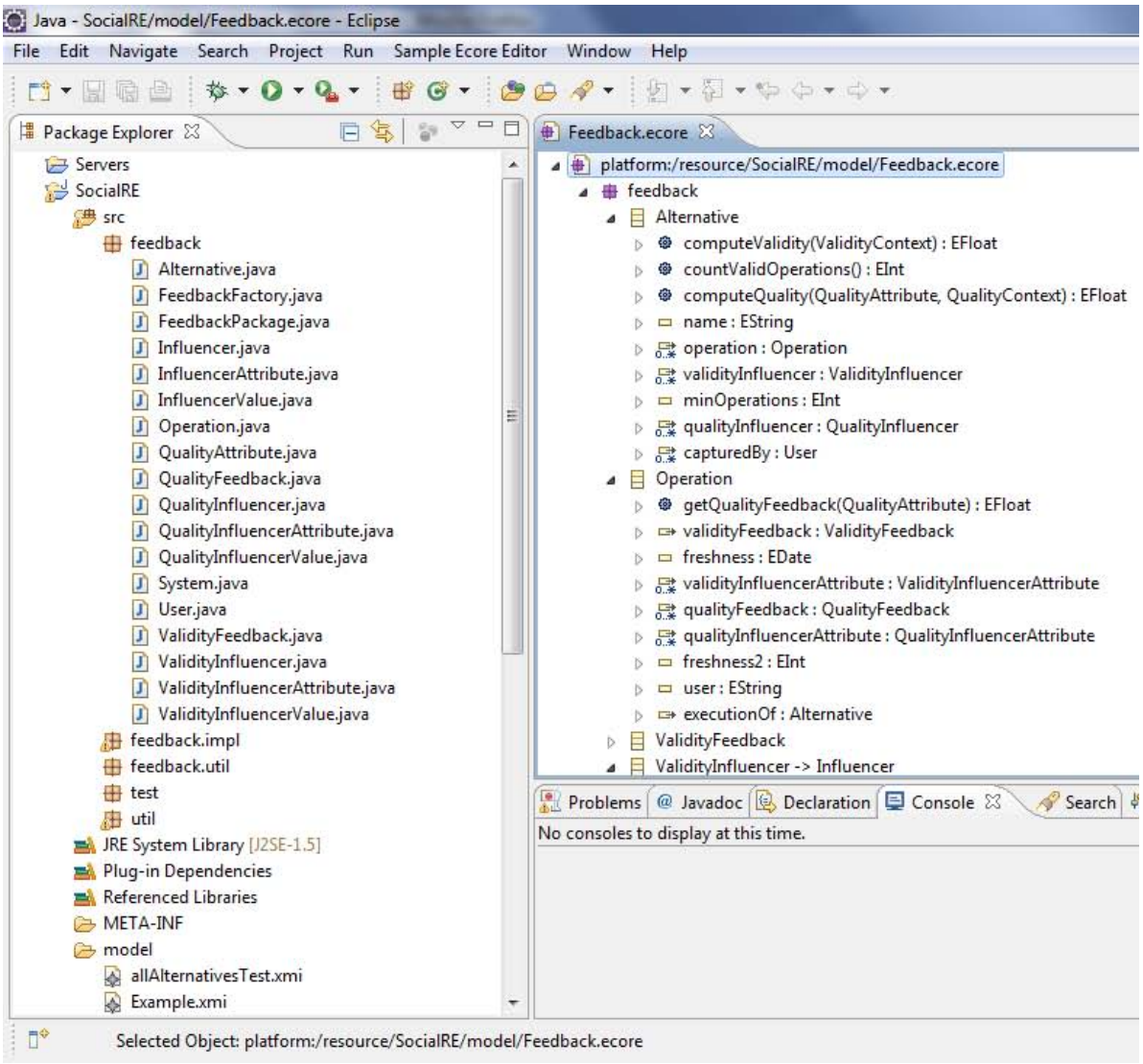

Fig. 11 A goal model metamodel definition editor

tion can take a goal model and can compute the validity and quality of the variants according to the provided user feedback. We have also used our tool for computing the quality and validity of the models used in the evaluation of the algorithm.

9 Related Work

Requirements engineering for adaptive systems raises several challenges [21, 22]. These challenges cover a wide spectrum of topics from coping with uncertainty and flexibility, the runtime monitoring of requirements and planning responses to changes, and the traceability from requirements to architecture and source code. Our proposed approach complements requirement-driven software adaptation by the collaboration of users so that the adaptation is not purely automatic but rather driven by social feedback. 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 and moreover allow for reflecting social judgment of software behavior rather than relying on designers' judgments which could be, or eventually become, invalid.

Cheng et al. note that in requirement models uncertainty have not been explicitly addressed in traditional requirements engineering [21]. Coping with uncertainty is an essential feature for adaptive systems so that the system can switch between different alternatives based on the operation of the system in practice. Qureshi and Perini [23] emphasize on flexibility of requirements refinement and provide a method that supports the runtime refinement of requirements arti-

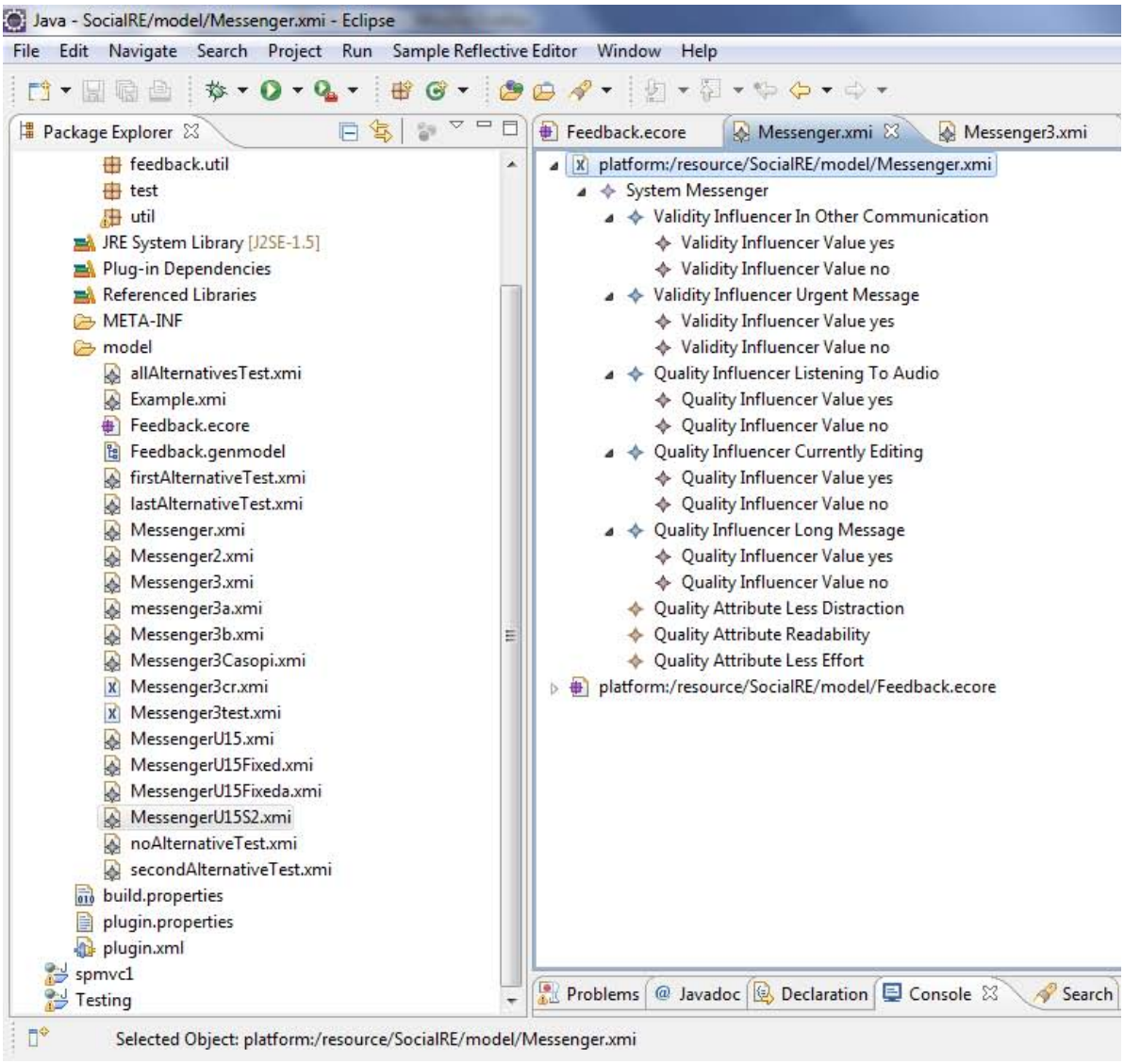

Fig. 12 An instance of our goal model metamodel shown in Fig.11

facts as a repetitive activity performed collaboratively between the users and the application itself.

In their seminal work, Fickas and Feather [16] highlight the importance of requirements monitoring at runtime as a basic and essential step for planning and enacting evolution. Souza et al. [24] note that the (partial) un-fulfillment 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 to monitor changes in these properties and decide when adaptation should take place. Baresi et al. [25] propose FLAGS (Fuzzy Live Adaptive Goals for Self-adaptive systems) for requirements-driven adaptation at runtime. FLAGS extend KAOS [10] mainly with adaptive goals which incorporate countermeasures for adaptation. When goals are not achieved by the current course of execution, adaptation countermeasures are triggered. The ultimate target is to alter the goal model at runtime and enforce adaptation directives on the running system.

Bencomo et al. [26] advocate that adaptation is planned either in a pre-defined way at design time or via an evolvable and reflexive response to some monitored parameters at runtime. The gap between goals and the system has to be bridged so that the system adaptation is guided by goals and the adaptation correctness is judge by the fulfillment of goals (requirements reflection). In another work related to requirements reflection, Sawyer et al. [12] discuss that runtime representation of requirements model, synchronizing the model with the architecture, dealing with uncertainty, multiple objective decision making, and self-explanation are

areas need to be considered in realizing a requirements-aware system.

In self-adaptation [1,2] and autonomic computing [27], software can monitor and analyse changes in its internal and operational environment and plan and execute an appropriate response. For example, Self-protection [28] refers to the ability of monitoring and analysing changes indicating security breaches and intrusions and planning a set of actions to defend the system or recover from their effects. Besides the research on purely-automated adaptation, there is an increasing trend to involve users in the adaptation loop [21] as we advocate in this paper. As we mentioned earlier, we present social adaptation to overcome self-adaptation limitations in monitoring and analysing another driver of adaptation which is the users' evaluation of the role of the system as a means for reaching the requirements. Moreover, social adaptation relies on the collective feedback provided by an open community of users so that adaptation is accelerated and its feasibility is increased.

Social adaptation is complementary to personalization and customizing software to individuals [19,29]. Personalization deals with various aspects of system design such as the user interfaces [30,31], information content [32,33], etc. Social adaptation tailors a system to the collective judgement of its users' community while personalization customizes a system to the characteristics of individuals. That is, social adaptation is about socializing a system instead of personalizing it. While socialization clearly does not replace personalization, it is essential when the system is highly variable and the individual users use the system for relatively limited number of times and the system validity and quality is subject to frequent changes. In such settings, customizing software would better aggregate the judgments made by users who used the system in the past and benefit of that to maximize the satisfactions of the current users.

Recent research has explored the role of context on requirements (e.g., [34], [35], [18]) and the elicitation of contextual requirements [36]. In these works, the relationship between context and requirements are not evolvable, i.e., there is a certain degree of certainty when analysing the context influence on requirements. Such assumption could be valid when dealing with well-known system scenarios where system relationship with its environment is predictable at design time. Social adaptation serves when specifying this relation is uncertain. It makes it subject to validation and evolution driven by the users feedback about the quality and the validity of system different behaviours in different contexts.

10 Conclusions and Future Work

Adaptation is increasingly becoming a critical demand so that validity and quality of software is maintained over time. It avoids us altering the system manually in order to cope with changes in its internal state and its operational environment. We advocated social feedback as a main and primitive driver for adaptation. Social feedback mainly concerns the validity and quality of each system alternative with respect to meeting users' requirements. Social feedback is provided by users who are considered first-class evaluators of the systems. Such feedback is essential as it cannot be replaced by or inferred from information monitorable by purely automated means. Social adaptation is the ability of software to obtain and analyze social feedback in order to plan adaptation. Adaptation is understood as the selection between different system alternatives to reflect users' feedback. Thus, we enrich software engineering for adaptive systems with a systematic approach for capturing another adaptation driver (users' feedback) and involving users as collaborators in planning and guiding adaptation.

The crux of social adaptation is the use of the collective feedback which stands on the other side of personalizing software to each user separately based on his interaction history and feedback. Our inspiring principle is the wisdom-of-crowds [37] which is particularly important when designing systems for an open community of users (e.g., tour-guide for tourists in a city, assistant-system for new students in a university, online auction systems for sellers and buyers over the world, etc.) rather than pre-determined community of users (e.g., payroll system or a meeting scheduled for already known or rarely changing set of users in a single organization). However, some systems are hybrid and incorporate both closed and open communities (e.g., a library system with predefined managers and officers and an open community of students). Social adaptation allows for accelerating adaptation by relying on peer users' previous judgment of the validity and the quality of each system alternatives.

This paper aimed to present foundations and a set of techniques for planning and enacting social adaptation. However, our research in this area is open to multiple directions such as:

- *Equilibrating multiple roles of users.* We studied social adaptation which considers only one kind of users. A system alternative might interact with different users each playing different role in the system. Users of one role might provide validity and quality feedback of a system alternative differently from users of another role. Consequently, the adaptation

has to equilibrate diversity of roles being played in the system. For example, the messenger system can also take the feedback from users who are sending the message and adapt by choosing the way to deliver messages which also considers the sender feedback.

– *Increasing openness-to-crowd.* We studied the involvement of users in judging the validity of systems alternatives and their quality against a pre-defined set of quality attributes. However, designers might not be even certain of what quality attributes are relevant to users and if the defined set is complete. As a solution, users might be given the possibility to manage collaboratively the definition and the assessment of their own quality attributes. For example, in a gallery-assistant system a user might put an attribute such as "less noise", and other users might contribute to judge each system alternative against it. That is to say, social adaptation can be established in a way more open to users so that the role of designers is minimized and users are given more liberty in planning and enacting adaptation.
– *Multi-criteria decision making.* We studied the adaptation that takes only one adaptation driver which is the social feedback. However, multiple other criteria play role in the selection between system alternatives and, thus, adaptation. For example, the context in which the system operates, the law enacted in the system environment, the resources available in it are examples of criteria which restrict the space of applicable alternatives of the system and influence their qualities. We still need to merge between these different criteria and social feedback in order to achieve a more holistic system adaptation.

Acknowledgement

This work has been partially funded by the EU Commission through the FastFix project, and by Science Foundation Ireland grant 10/CE/I1855. We also thank Haruhiko Kaiya for discussions that enriched the ideas of this paper.

References

1. R. Laddaga. Self-adaptive software. Technical Report 98-12, DARPA BAA, 1997.
2. Mazeiar Salehie and Ladan Tahvildari. Self-adaptive software: Landscape and research challenges. *ACM Transactions on Autonomous and Adaptive Systems*, 4:14:1–14:42, May 2009.
3. R. Murch. *Autonomic computing.* IBM Press, 2004.
4. Raian Ali, Carlos Solis, Mazeiar Salehie, Inah Omoronyia, Bashar Nuseibeh, and Walid Maalej. Social sensing: When users become monitors. *In the proceedings of the New Ideas Track of the joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE'11).*, 2011.
5. Joseph S. Dumas and Janice C. Redish. *A Practical Guide to Usability Testing.* Intellect Books, Exeter, UK, UK, 1st edition, 1999.
6. Karel Vredenberg, Scott Isensee, and Carol Righi. *User-Centered Design: An Integrated Approach with Cdrom.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 2001.
7. E. Yu. *Modelling Strategic Relationships for Process Reengineering.* PhD thesis, Department of Computer Science, University of Toronto, Toronto, Canada, 1995.
8. Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
9. Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information Systems*, 27(6):365–389, 2002.
10. Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.
11. Walid Maalej, Hans-Jörg Happel, and Asarnusch Rashid. When users become collaborators: towards continuous and context-aware user input. In *Proceeding of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, OOPSLA '09, pages 981–990. ACM, 2009.
12. Pete Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, and Anthony Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 95–103, Washington, DC, USA, 2010. IEEE Computer Society.
13. Anthony Finkelstein Andrea and Andrea Savigni. A framework for requirements engineering for context-aware services. In *Proceedings of the 1st International Workshop From Software Requirements to Architectures*, pages 200–1, 2001.
14. Raian Ali, Fabiano Dalpiaz, Paolo Giorgini, and Vitor E. Silva Souza. Requirements evolution: from assumptions to reality. 2011.
15. John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42:31–37, January 1999.
16. S. Fickas and M. S. Feather. Requirements monitoring in dynamic environments. In *Proceedings of the Second IEEE International Symposium on Requirements Engineering*, RE '95, Washington, DC, USA, 1995. IEEE Computer Society.
17. Daniel Sykes, William Heaven, Jeff Magee, and Jeff Kramer. From goals to components: a combined approach to self-management. In *Proceedings of the 2008 international workshop on Software engineering for adaptive and self-managing systems*, SEAMS '08, pages 1–8, New York, NY, USA, 2008. ACM.
18. Raian Ali, Fabiano Dalpiaz, and Paolo Giorgini. A goal-based framework for contextual requirements modeling and analysis. *Requir. Eng.*, 15:439–458, November 2010.
19. Bowen Hui, Sotirios Liaskos, and John Mylopoulos. Requirements analysis for customizable software goals-skills-preferences framework. In *Proceedings of the 11th IEEE International Conference on Requirements Engineering.* IEEE Computer Society, 2003.
20. Mark Weiser. The computer for the 21st century. *SIGMOBILE Mob. Comput. Commun. Rev.*, 3:3–11, July 1999.

21. Betty H. C. Cheng, Holger Giese, Paola Inverardi, Jeff Magee, and Rogério de Lemos. Software engineering for self-adaptive systems: A research road map. In *Software Engineering for Self-Adaptive Systems*, pages 1–26, 2008.
22. Betty H. C. Cheng and Joanne M. Atlee. Research directions in requirements engineering. In *2007 Future of Software Engineering*, FOSE '07, pages 285–303, Washington, DC, USA, 2007. IEEE Computer Society.
23. Nauman A. Qureshi and Anna Perini. Requirements engineering for adaptive service based applications. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 108–111, Washington, DC, USA, 2010. IEEE Computer Society.
24. Vítor E. Silva Souza, Alexei Lapouchnian, William N. Robinson, and John Mylopoulos. 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, New York, NY, USA, 2011. ACM.
25. Luciano Baresi, Liliana Pasquale, and Paola Spoletini. Fuzzy goals for requirements-driven adaptation. In *Proceedings of the 2010 18th IEEE International Requirements Engineering Conference*, RE '10, pages 125–134, Washington, DC, USA, 2010. IEEE Computer Society.
26. Nelly Bencomo, Jon Whittle, Pete Sawyer, Anthony Finkelstein, and Emmanuel Letier. Requirements reflection: requirements as runtime entities. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pages 199–202, New York, NY, USA, 2010. ACM.
27. Simon Dobson, Spyros Denazis, Antonio Fernández, Dominique Gaïti, Erol Gelenbe, Fabio Massacci, Paddy Nixon, Fabrice Saffre, Nikita Schmidt, and Franco Zambonelli. A survey of autonomic communications. *ACM Trans. Auton. Adapt. Syst.*, 1:223–259, December 2006.
28. Peyman Kabiri and Ali A. Ghorbani. Research on intrusion detection and response: A survey. *International Journal of Network Security*, 1:84–102, 2005.
29. Odd-Wiking Rahlff, Rolf Kenneth Rolfsen, and Jo Herstad. Using personal traces in context space: Towards context trace technology. *Personal Ubiquitous Comput.*, 5:50–53, January 2001.
30. Daniel S. Weld, Corin Anderson, Pedro Domingos, Oren Etzioni, Krzysztof Gajos, Tessa Lau, and Steve Wolf. Automatically personalizing user interfaces. In *International Joint Conference on Artificial Intelligence*, pages 1613–1619, 2003.
31. Silvia Schiaffino and Analía Amandi. User - interface agent interaction: personalization issues. *Int. J. Hum.-Comput. Stud.*, 60:129–148, January 2004.
32. Jaime Teevan, Susan T. Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 449–456, New York, NY, USA, 2005. ACM.
33. Paul Alexandru Chirita, Wolfgang Nejdl, Raluca Paiu, and Christian Kohlschütter. Using odp metadata to personalize search. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 178–185, New York, NY, USA, 2005. ACM.
34. Mohammed Salifu, Yijun Yu, and Bashar Nuseibeh. Specifying monitoring and switching problems in context. *Requirements Engineering, IEEE International Conference on*, 0:211–220, 2007.
35. Herman Hartmann and Tim Trew. Using feature diagrams with context variability to model multiple product lines for software supply chains. In *Proceedings of the 2008 12th International Software Product Line Conference*, pages 12–21, Washington, DC, USA, 2008. IEEE Computer Society.
36. Norbert Seyff, Florian Graf, Neil A. M. Maiden, and Paul Grünbacher. Scenarios in the wild: Experiences with a contextual requirements discovery method. In *Requirements Engineering: Foundation for Software Quality*, volume 5512 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2009.
37. James Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.