The Journal of Systems and Software 105 (2015) 54–71

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Emotion-led modelling for people-oriented requirements engineering: The case study of emergency systems

Tim Miller[a,*], Sonja Pedell[b], Antonio A. Lopez-Lorca[c], Antonette Mendoza[d], Leon Sterling[e], Alen Keirnan[f]

[a] Department of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia
[b] Centre for Design Innovation, School of Design, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[c] Centre for Design Innovation, Faculty of Science, Engineering, and Technology, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[d] Department of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia
[e] Faculty of Science, Engineering, and Technology, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[f] Centre for Design Innovation, Swinburne University of Technology, Melbourne, VIC 3122, Australia

ARTICLE INFO

Article history:
Received 24 March 2014
Revised 23 December 2014
Accepted 17 March 2015
Available online 27 March 2015

Keywords:
Requirements engineering
Agent-oriented modelling
Emotions

ABSTRACT

In the field of design, it is accepted that users' perceptions of systems are influenced by emotion as much as cognition, and functionally-complete products will not be adopted if they do not appeal to emotions. While software engineering methodologies have matured to handle non-functional requirements such as usability, what has not been investigated fully is the emotional needs of people. That is, what do users want to feel, and how do they feel about a system? In this paper, we argue that these emotional desires should be treated as first-class citizens in software engineering methodology, and present preliminary work on including emotions in requirements models using *emotional goals*. We evaluate these models both with a controlled user study, and on a case study of emergency systems for older people. The results of the controlled user study indicate that people are comfortable interpreting and modifying our models, and view the inclusion of emotions as first-class entities as a positive step in software engineering. The results of our case study indicate that current emergency systems fail to address the emotional needs their users, leading to low adoption and low usage. We conceptualised, designed, and prototyped an improved emergency system, and placed it into the homes of nine older people over a period of approximately two weeks each, showing improved user satisfaction over existing systems.

1. Introduction

> "*... even if a design is elegant and functional, it will not have a place in our lives unless it can appeal at a deeper level, to our emotions.*"
> — Hartmut Esslinger (Sweet, 1999, p. 9).

Evidence suggests that inadequate consideration of requirements is a major cause of software project failure (El Emam and Koru, 2008). In the context of technology adoption, users reject a technology or use it in limited ways when their needs and experiences with that technology are not addressed (Mendoza et al., 2010a; 2010b). As the famous quote from Esslinger above conveys, this is especially true in the case of social objectives such as the *emotional* needs of users. The consideration of emotion in addition to cognition has become more prevalent in design in recent years Norman (2007), including human–computer interaction design, but such considerations have not transferred successfully to software engineering, despite evidence showing that a user's acceptance of product is typically based on emotion rather than cognitive (Norman, 2007). This is especially important in domestic or social systems, in which workflows are loose and people do not generally have the well-defined roles and responsibilities found in organisational settings.

Software engineers are trained to build systems with desired functionality and non-functional properties. However, software systems are often designed[1] poorly, detracting from the user experience. Cooper (1999) refers to this as "*the inmates running the asylum*": software engineers elicit functional and non-functional requirements from users, then design a product to fulfil these requirements as they themselves would like it to be, resulting in software that fails to fulfil the desires of its intended users. This problem is made worse by a

* Corresponding author: Tel.: +61383441318.
E-mail address: tmiller@unimelb.edu.au (T. Miller).

[1] By "design" here, we refer to the design of the product, not of the software architecture or detailed designs.

http://dx.doi.org/10.1016/j.jss.2015.03.044
0164-1212/

common misconception that problems with the interaction design can be addressed after the development by simply fixing up the user interface.

From the perspective of software engineering, a first important step of addressing user experience is eliciting the emotional desires of stakeholders. A growing appreciation can be found in literature that existing software engineering methods are limited by not considering social objectives (Baxter and Sommerville, 2010; Rahwan et al., 2006; Walenstein, 2003), a view expressed well by Baxter and Sommerville (2010, p. 14) in their comprehensive review of design methods for socio-technical systems: "*Modelling and abstraction is fundamental to software engineering, with models of different types being used by engineers to communicate. The practical use of socio-technical approaches has to acknowledge this by providing a means of modelling, and by integrating with existing approaches. [. . .] The abstractions currently used in technical system modelling (e.g., use-cases, objects, etc.) do not seem to us to be sufficient to represent socio-technical considerations.*"

In previous work (Miller et al., 2012; Pedell et al., 2014b), some of the authors presented a systematic and repeatable process and method for understanding the roles and goals within a social domain for the purpose of informing technology design. At the heart of the method were agent-oriented models (Sterling and Taveter, 2009). Ethnographic data were collected using a variety of means, and analysed using a grounded analysis. We used agent-oriented models to record the ground theory that resulted from that analysis. An important aim of the work was to provide a simple yet flexible modelling notation that could be used to create *boundary objects*, which, as Paay et al. (2009) demonstrate, can be used as shared artefacts between stakeholders from different disciplines. We designed and implemented technology probes (Hutchinson et al., 2003), and modelled the data collected from these probes as agent-oriented models. These models allowed us to represent human activities as well as software system behaviour. One novel outcome was the use of *quality goals* to represent socially-oriented requirements such as "having fun" and "being playful".

In this paper, we improve our previous work by adding the concept of *emotional goals* (Marshall, 2012) to the notation and method, which capture the desired feelings of stakeholders in a socio-technical system, and how these relate to the system and each other. We call these models *people-oriented software engineering* (POSE) models, because of their focus on the people within the system, as well as the software.

In this paper, we present a two-part evaluation of our models. First, we present a user study in which we compare our notation for capturing user needs against the well-known social modelling notation *i*∗ (Yu, 2009). We asked a set of participants, some technical and some non-technical, to answer a series of questions about an *i*∗ model and a POSE model, and measured the time and accuracy of their responses. We then asked a set of qualitative questions around their preferences between the models. The results show that participants understand POSE models better and more quickly, prefer POSE models as boundary objects for modelling socio-technical systems, and prefer the use of explicit emotional goals in models over our previous approach of using quality goals to represent these social aspects.

Second, we evaluate the concept of emotional goals via a case study on emergency alarm systems. These systems allow a person to raise an alarm in the case of an emergency, and also to "check in" (wellbeing check) each day to convey that they are well by pressing a button. We interviewed 12 participants about emergency systems and their feelings toward technology in general. Using the ethnographic data collected, we modelled the emotional, functional, and quality goals of the key stakeholders using our models. Based on the findings of the case study, we discovered that many users of emergency systems, as well as their families, were not happy with the way the system operates for them. While the technological systems themselves were well-engineered, reliable products that fulfilled the functionality of alarms and wellbeing checks, the emotional needs of users were not met, leading to failure of the overall goals of the system. That is, older people did not receive assistance when needed because they were not carrying their devices, and did not feel that the wellbeing check supported their wellbeing. In the process, we learnt lessons about our models, and how to improve them. From the resulting models, we designed and built a new prototype of an emergency system. Evaluation of this prototype demonstrated an improved user experience over existing emergency alarms.

We next present our argument as to why emotions should be embedded as first-class citizens in software engineering, and in Section 3, we present relevant background for the paper. In Section 4, we present our modification to our previous modelling notation to include emotional goals. In Sections 5 and 6 discuss our evaluations of this using a user study and the case study of emergency alarm systems.

2. Emotions as first-class citizens in software engineering

In this section, we outline our argument for why emotions should be considered as first-class citizens in software engineering methodology.

The consideration of emotions in requirements engineering is not new, but has received insufficient attention outside of the games community, with only a handful of papers addressing the issue of how to address emotions in software engineering (Bentley et al., 2002; Colomo-Palacios et al., 2011; 2010; Ramos and Berry, 2005; Thew and Sutcliffe, 2008). Further, as far as the authors are aware, integrating emotions fully within requirements engineering has not been explored, nor has carrying emotions through the software engineering lifecycle. As part of our larger research program, we aim to model emotions as first-class entities in software engineering, carrying these goals through the software engineering process, including requirements engineering, product design, software design, implementation, testing, and validation.

2.1. The case for emotions

The idea of eliciting emotional desires for product design is not new, and there has been a large body of work over the previous two decades; see Desmet and Hekkert's editorial for the special issue *Design & Emotion* in the International Journal of Design (Desmet and Hekkert, 2009) for an excellent overview of this work.

Norman's book on emotional design (Norman, 2007) is one of the most seminal pieces of work on this topic. He argues that designers must elicit desired user emotions and explicitly address them as part of the design process. Norman describes how three levels of the human brain affect emotion, and what this means for designers:

(1) *Visceral* processing is the automatic, pre-conscious processing that makes fast judgements. Visceral processing is programmed in humans, meaning that its effect is fairly consistent across different people. With regards to design, a person's emotional state is affected by visceral processing based on the appearance or "look & feel" of a product, such as colours and style.
(2) *Behavioural* processing is sub-conscious, and is the part that controls "everyday" behaviour. With regards to design, behavioural processing is about the use and experience with a product, with the experience related to "function, performance, and usability" (Norman, 2007). Like the visceral level, emotional responses to the same event at this level are quite consistent across different people.
(3) *Reflective* processing is conscious, and is the contemplative part of processing. It is only at this level that "the highest levels of feeling, emotions, and cognition reside" (Norman, 2007). With regards to design, it is about the meaning of a product and its