Future Generation Computer Systems 34 (2014) 110–123

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

The user in the loop: Enabling user participation for self-adaptive applications

Christoph Evers [a,*], Romy Kniewel [b], Kurt Geihs [a], Ludger Schmidt [b]

[a] Distributed Systems Group, University of Kassel, Wilhelmshöher Allee 73, 34121 Kassel, Germany

[b] Human–Machine Systems Engineering, University of Kassel, Mönchebergstraße 7, 34125 Kassel, Germany

HIGHLIGHTS

- Interdisciplinary research results regarding human–computer interaction and self-adaptive software.
- We provide a concept of how the user can be integrated to the adaptive feedback loop of autonomous systems.
- We show how notifications on adaptations can be designed to be helpful and non-interrupting.
- We demonstrate extensions to an existing middleware solution that enable active user participation for adaptive applications.
- We conducted a substantial user study with 62 participants.

ARTICLE INFO

Article history:
Received 11 February 2013
Received in revised form
24 October 2013
Accepted 3 December 2013
Available online 16 December 2013

Keywords:
Self-adaptation
Usability
Autonomous computing
Interruption
User attention
Ubiquitous computing
Human–computer interaction

ABSTRACT

Future computing systems must adjust to the user's situations, habits, and intentions. Self-adaptive applications autonomously adapt to changing contexts without asking the user. However, the self-adaptive behaviour lacks of success if it does not correspond to the user's personal interaction habits and intentions, particularly for complex scenarios with a high degree of user interaction. Concerning the interaction design, such adaptations can be irritating and distracting for the user if they do not match the current situation. In this article we provide a solution how to integrate the user in the self-adaptation feedback loop. The user will be able to influence the adaptation behaviour at run-time and in the long term by setting individual preferences. Consequently, we achieve a harmonisation between full application autonomy and user control. We implemented our generic concepts by extending an existing self-adaptation middleware with capabilities to respect the user's application focus and interaction behaviour. A notification-based solution for user participation has been evaluated in a substantial user study with 62 participants. Although participants perceived much better control with our solution, the study made clear that notification design is specific for each adaptation type.

1. Introduction

Context-aware and self-adaptive software is a key element of future computing systems. Self-adaptive applications are able to consider their environmental context through information provided by sensors or other data sources and dynamically adapt to changes that may occur in highly volatile and heterogeneous environments.

Unlike traditional applications, adaptive applications are able to change their state and behaviour at run-time. Application adaptation is a result of autonomic computing research [1] to achieve the best desirable service for the user without involving the user in the machine's adaptation decision. There have been various approaches to incorporate adaptive behaviour in software. The *MAPE-K* feedback loop [2] is the foundation for many of them. It defines an autonomic manager that comprises the four basic elements *Monitoring*, *Analysing*, *Planning*, and *Executing* (Fig. 1). The *Knowledge* component collects data and allows further reasoning, e.g. creating a knowledge base or learning of behaviour. A MAPE-K feedback loop is a closed loop that explicitly aims at excluding user participation during run-time. All decisions and actions are made by the system. The design of the decision making, i.e. analysis and planning, is created by the software developer.

A research challenge is to integrate the user in this feedback loop when user participation is desired or required [3]. Experience with adaptive applications has shown that it is not always preferable to have applications operating completely autonomously. Users may further not trust such applications and conceal their actions [4]. Van der Heijden [5] argues that transferring control from

* Corresponding author. Tel.: +49 5618046279.
E-mail addresses: evers@vs.uni-kassel.de (C. Evers), r.kniewel@uni-kassel.de (R. Kniewel), geihs@uni-kassel.de (K. Geihs), l.schmidt@uni-kassel.de (L. Schmidt).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.12.010

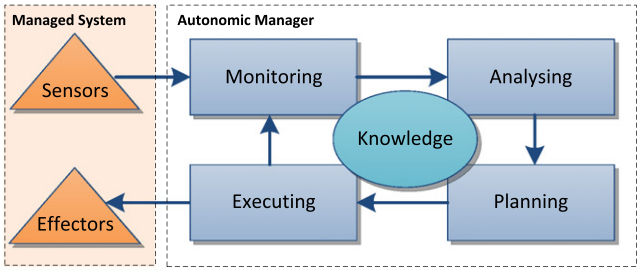

Fig. 1. MAPE-K feedback loop in autonomic computing [2].

the user to the system results in increased user anxiety. Also, users would like to customise the application to certain different degrees so it matches their personal preferences. Salehie et al. [6] state that human involvement in general is quite valuable for improving the manageability and trustworthiness of adaptive software. Barkhuus and Dey [7] revealed in a study on context-aware applications that users preferred context-aware features over personalisation, but in the same moment they experienced a lack of control. We do not question the concept of autonomic computing in general as it is fundamental for future computing scenarios, but rather we propose to open the feedback loop for user participation in applications with a high degree of user interaction.

Our contribution is as follows: we first define specific requirements from human–computer interaction for adaptive applications. We then provide a solution how to integrate the user in the decision loop of an adaptive application to respect his interaction practices and to influence the application's behaviour in order to increase the controllability of an application. An important element of our model-driven approach is the consideration of the user's perceptional focus on the application. We identify dimensions of component-based applications that the user is able to influence and hence achieve a higher user acceptance. Based on the dimensions of user participation we define several mechanisms for active user participation while still maintaining application autonomy.

Although the presented solution assumes an underlying middleware-based and component-based solution for architectural adaptation, the generic concept can be transferred to other adaptation domains. Further, we focus on user acceptance coming through usability. Also privacy and security issues need to be considered for user acceptance in general, but this is beyond the focus of this paper.

The remainder of this article is structured as follows: in Section 2 we first discuss the related work. Section 3 gives an overview on adaptive software and the corresponding usability requirements. Section 4 introduces the Meet-U case study and includes results from an initial usability evaluation. Based on the experience we present our approach to integrate the user in the adaptive feedback loop in Section 5. Section 6 describes our system architecture. We evaluate the notification-based reactive user adaptation mechanism in Section 7 and finish with a conclusion in Section 8.

2. Related work

A typical approach to achieve application-level adaptation is the use of an adaptation middleware which incorporates a MAPE-K feedback loop. Available solutions [8–10] show how architectural re-configuration is performed in the domain of ubiquitous and mobile applications. We expect a high degree of user interaction in those domains. However, none of the middleware approaches explicitly considers user participation. For example, in MUSIC [9], the application acts autonomously according to a variability model that is specified at design-time by the application developer. This would not be a problem as long as the user is not affected negatively, but as we will see later, this is unlikely.

Latest applications like Google Now [11] or Motorola Smartactions [12] adapt information or systems setting according to particular situations. However, these applications include very limited elements of adaptation and adaptation features are implemented in proprietary ways for specific use cases. For every application the developer has to rethink how to implement the adaptive behaviour. There is no systematic way of designing application-level adaptation with a specific degree of user participation.

User influence in adaptive applications has been considered as important ever since [3,6] but only few works actually address this topic. Henricksen et al. [13], as well as Fong et al. [14], present approaches to incorporate user preferences in the adaptation decision. Their work focuses on intelligibility of adaptation by modelling context and context information, revealing context information to the user, and by allowing the user to modify the adaptation behaviour based on preferences. The concept of user preferences can be generalised to personalisation. Gil et al. [15] employ personas and feature models to model service obtrusiveness based on the type of user. Feature models decompose the interaction in different adaptation aspects. Each feature describes a variant of the application interaction.

We claim that intelligibility and accountability of adaptive applications in combination with preference modelling are very important requirements. Revealing context information to the user might be one solution but the user does not necessarily want to know about technicalities of the adaptation, nor does he want to be bothered with concerns about which context information is actual necessary to realise a particular feature. Rather we propose to integrate the user in the adaptation feedback loop and thus enable him to influence the behaviour of the application explicitly and implicitly.

Another promising concept is machine learning on the user's device [16,17]. With active learning the user is able to rate a specific adaptation decision whether it was good or bad. The application is then able to incorporate the user's decision in its own adaptation reasoning which results in higher personalisation. Unfortunately, this approach requires a lot of additional interaction in the early phases of application use. This interaction is not related to the user's actual tasks and hence an additional burden. Here the challenge is to find a trade-off between long term profit and short term interruption. Another problem could be the increased computational demand on a mobile device.

Gil and Pelechano [18] use machine learning techniques to minimise mobile interaction obtrusiveness by adapting service notifications. However, their focus is on notifications, coming from different services the user has subscribed to. They do not consider adaptive software in the sense of autonomous software reconfiguration.

Since users are surrounded by information technology every day and everywhere, obtrusiveness and interruption became important research topics. Several studies analysed the interruption of messages and notifications on mobile devices [19,20], while others examined the interruptive character of an unmuted mobile device [21]. They all stress the fact that interruptions come at the cost of diminished user attention whether they are signalled by notifications or ringing phones. This is very similar to adaptive software as application reconfiguration at run-time can be at least as obtrusive as a notification.

3. Adaptation and usability

In this section we first describe properties of self-adaptive software and the basic terminology. We then go on by defining specific requirements coming from usability engineering and apply them to self-adaptive software.