A Pattern Approach to Dealing with NFRs in Ubiquitous Systems

Tomás Ruiz-López[1], José Luis Garrido[1], Sam Supakkul[2], and Lawrence Chung[3]

[1] Software Enginering Department, University of Granada
Periodista Daniel Saucedo Aranda s/n, 18.014 Granada, Spain
{tomruiz, jgarrido}@ugr.es
[2] Sabre Inc., USA
ssupakkul@ieee.org
[3] Department of Computer Science, University of Texas at Dallas, USA
chung@utdallas.edu

Abstract. Ubiquitous Computing has become increasingly important, thanks to its attractive aims to provide invisible computing everywhere and anytime. Dealing with Non-Functional Requirements (*NFRs*), such as usability or privacy, during requirements engineering for Ubiquitous Computing, however, is oftentimes a difficult task, given the unique features of this paradigm (e.g. context-awareness or technological unobtrusiveness). In this paper, we present an approach in which pieces of knowledge about recurring situations in ubiquitous systems are identified and clearly captured as patterns. The approach is illustrated through an example of an e-Learning ubiquitous system.

Keywords: requirements engineering, patterns, ubiquitous computing

1 Introduction

Dealing with Non-Functional Requirements (*NFRs*), such as usability or privacy, during requirements engineering for Ubiquitous Computing [12], however, is oftentimes a difficult task. Ubiquitous Computing requires a body of both broad and deep knowledge about such unique features as context-awareness and technological unobtrusiveness in this paradigm, and yet such knowledge by and large has been exercised only implicitly and informally, leading to miscommunications and lost opportunities for reuse.

Consider an e-Learning ubiquitous system, as shown in Figure 1. It depicts three situations, where some concerns have not been addressed properly: (a) phones start to ring during a class, when they are not supposed to; (b) devices expose sensitive data to some people that should not be aware of the data; and (c) the security mechanism to access the classroom/lab asks a person to enter a password to open the door. These three situations are contrary to some of the key principles of Ubiquitous Computing, such as context-aware adaptation or unobtrusiveness of the technology in the user's daily activities. These situations

have something in common - a bad assessment of different Non-Functional Requirements (NFRs), such as usability, privacy or security [1] in different context situations. Failure to systematically treat NFRs under contextual constraints may result in an overall loss of quality of the system, as pointed out in the previous scenario. Moreover, these situations can occur in some other domains as well (e.g. interrupting a movie at the cinema or sharing medical data outside the hospital). In other words, these problems can be recurring and may happen regardless of the type of the system.

Undoubtedly, clearly capturing knowledge about such problems and solutions to them for later explicit reuse would be highly desirable. In this paper, we present an approach in which pieces of knowledge about recurring situations in ubiquitous systems are identified and clearly captured as patterns, and later explicitly reused across different domains. The use of patterns during Requirements Engineering is not new. It has been explored in previous works [11], but there is little or no previous work regarding Patterns for NFRs in Ubiquitous Systems. Our work is complementary to the method to systematically deal with NFRs in Ubiquitous Systems [8].

The rest of this paper is organized as follows. Our pattern approach to dealing with NFRs for Ubiquitous systems is introduced in Section 2, where different types patterns are discussed, together with their benefits and drawbacks. The approach is illustrated through the application of patterns to an e-Learning Ubiquitous System in Section 3. Related works are described in Section 4. Finally, a summary and contributions of the paper, together with future work, are described in Section 5.

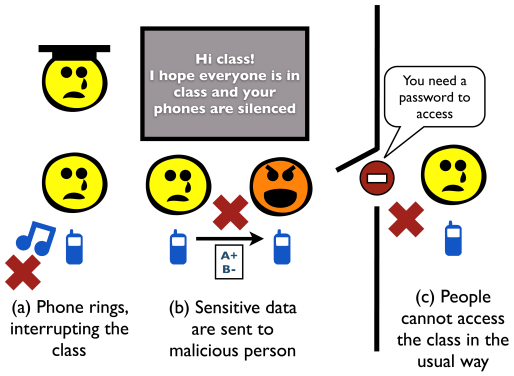

Fig. 1. Ubiquitous e-Learning Scenario. Different problems can occur, derived from a bad addressing context-based adaptations.

2 Non-Functional Requirements Patterns for Ubiquitous Systems

In this section, the core of this proposal is presented. It consists of a set of Non-Functional Requirements Patterns for Ubiquitous Systems that have been identified from the study of existing systems and the experience of the authors in the construction of ubiquitous applications. The patterns presented in this section are a small subset which has been chosen to illustrate their need. The patterns presented above are a subset from a larger set that have been identified. Moreover, there are different dimensions to organize them that can enable the appearance of new patterns by their generalization, specialization, composition or instantiation.

2.1 Context-sensitive I/O Pattern.

Ubiquitous systems are essentially interactive. Thus, the user and the system needs to exchange information with each other using natural interfaces that, at the same time, are not obtrusive to the user. The way the system outputs information to the user and the user provides input to the system should not be fixed. In fact, it depends on contextual circumstances. Depending on the environment that the user is, the system may decide to provide the information in a different way, or even do nothing, if that causes a disruption in the user's activities [5].

The *Context-sensitive I/O Pattern* provides a possible solution to model this situation, which is recurring in different ubiquitous systems. Consider that there is a goal, G, that aims to provide or receive information from the user. Many different operationalizations, $O_1 \dots O_n$, may contribute to the satisfaction of this objective, with different levels of contribution. Finally, some context situations, $C_1 \dots C_m$, are relevant to the system. The pattern proposes to relate the operationalizations to the goal, conditioned by the occurrence of some context situations. Formally speaking, O_i contributes to G under the context situation C_j (Figure 2, top).

2.2 Context-sensitive Data Sharing Pattern.

Preservation of sensitive data is a general challenge for software designers, which also holds true for ubiquitous systems [4]. Personalization and customization are key features of this kind of systems, and to be able to achieve them, they need to know big amounts of the user's sensitive data.

To achieve this task, either the user provides data to feed the adaptation mechanisms, or the system records events from the user's daily life and applies mining techniques to infer the user profile from user's behavior, and try to anticipate the customization. Dangerous scenarios can happen when all this sensitive data have to be exchanged with others. Adaptations in new environments or data exchange with other users, or between applications on their behalf, can lead to

the exposure of sensitive data that the user does not want to share. In order to overcome this situation, the *Context-sensitive Data Sharing Pattern* proposes to take into account the trustworthiness of the context situation to determine how much information is shared. To achieve this, user sensitive information is organized in a hierarchical structure, where the bottom level nodes provide few user information, and the top level nodes provide a lot of information. Formally, operationalization O_i (that enables the exchange of data D_i) contributes to the satisfaction of goal G under context situation C_j (Figure 2, middle).

When this structure has been created, each node represents the amount of information that may be shared. These nodes are related to the context situations where they have to be shared, in such a way that the lower the trustworthiness of the situation is, the less information is shared. Also, it enables to provide the right amount of information at each time, enhancing the accuracy of the information transmission.

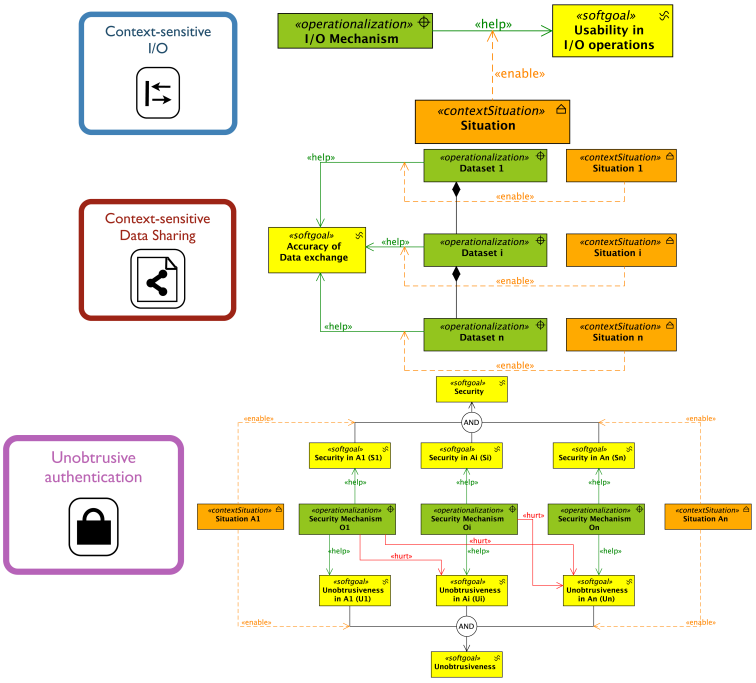

Fig. 2. Structure of the proposed patterns.

2.3 Unobtrusive Authentication Pattern

In many cases, users need to do security-sensitive activities, either because they deal with private information or because they access to certain devices which

use is restricted. Thus, the ubiquitous system has to determine the identity of the user to guarantee access to the desired data or operations. However, the paradigm claims that this authentication should be done in a transparent way, making the analyst to face a tradeoff between security and unobtrusiveness [10]. The *Unobtrusive identification Pattern* aims to provide support for a tradeoff between *Security* and *Unobtrusiveness* applying the following strategy (Figure 2, bottom):

1. Decompose softgoal *Security* into different sub-softgoals, $S_1 \dots S_n$, where the security level of S_i is higher than the security level of S_j if $i < j$. In order to perform the decomposition, take into account the different context situations, $A_1 \dots A_m$ where the security levels may change.
2. Decompose softgoal *Unobtrusiveness* into different sub-softgoals, $U_1 \dots U_n$, where the accepted level of unobtrusiveness of U_i is lower than the accepted level of unobtrusiveness of U_j if $i < j$. In order to perform the decomposition, proceed similarly to what was exposed in the previous step, taking into account context situations.
3. Assign operationalizations, $O_1 \dots O_n$, that help to *satisfice* the security requirements, in such a way that O_i helps to achieve both S_i and U_i, but hurts U_j, for all $j > i$.

3 Illustrative Example: e-Learning Ubiquitous System

The application of the proposed patterns to the running scenario is as follows:

- **Context-sensitive I/O:** When students are communicating with each other, the system may need to deliver messages (i.e., output some information). Depending on whether they are in class or not, the device should vibrate or ring, respectively, in order to avoid disturbances in the class (Figure 3, top).
- **Context-sensitive Data Sharing:** The e-Learning system supports communication between users through nearby devices discovery. In these cases, the user needs to provide some information about his/her identity and sensitive data. In order to provide accurate data, s/he may decide to share his/her student records to School administrative staff, and not to share any information to unknown users. Outside school, s/he may, for instance, share his/her medical data to hospital services staff (Figure 3, middle).
- **Unobtrusive Authentication:** Consider two possible situations with different security levels: accessing a lab and accessing to someone's private account. In this case, unobtrusiveness and security are decomposed taking this into account. For the former, using Near Field Communication (NFC) [3] can be a good option, since this technology enables the detection of a device carried by the user without any user action, while being secure. For the latter, the typical username and password authentication can be chosen, since more sensitive data are handled (Figure 3, bottom).

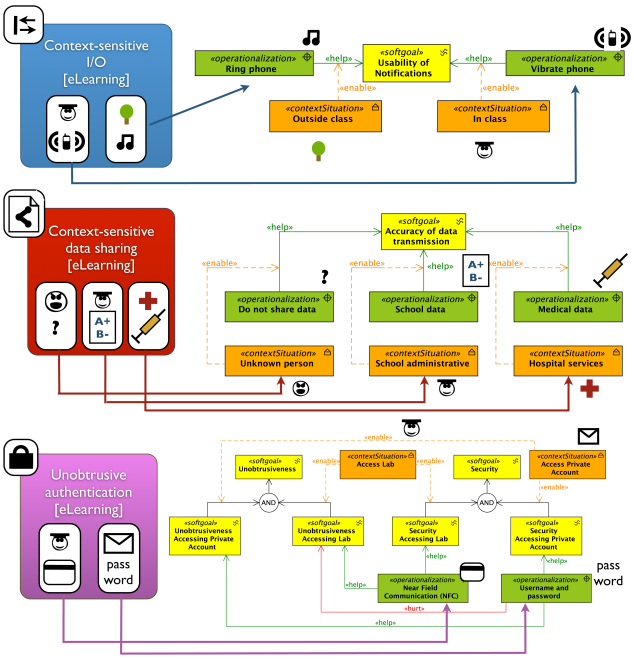

Fig. 3. Application of the Context-sensitive I/O, Context-sensitive Data Sharing and Unobtrusive Authentication Patterns, respectively.

Figure 4 depicts the resulting scenario after the application of the patterns. The problems that were presented at the beginning of this paper have been addressed through the application of the proposed patterns. Also, some NFRs have been enhanced. Since I/O is adapted to the context, the number of unnecessary interruptions decreases and the users are more likely to accept the system. The amount of data that are shared is more accurate to the target receiver, and it avoids sending private data to unknown users as well. Finally, users can perform normal activities, like accessing a class, in the usual way, but still being secured.

4 Discussion

The notion of design pattern in software engineering has been present for several years since the appearance of the book by the *Gang of Four* with a set of patterns for Object-oriented design [2]. Similar to this approach, some authors have identified recurring design patterns in the ubiquitous computing field [5]. Some other existing works try to discover and define potential patterns that mainly focus on one of the main features of ubiquitous systems, context-awareness [7].

The use of patterns is not limited to design; requirements patterns can also be defined and used, although systematic treatment of NFRs through the application of patterns has not been widely explored. For instance, Withall [13]

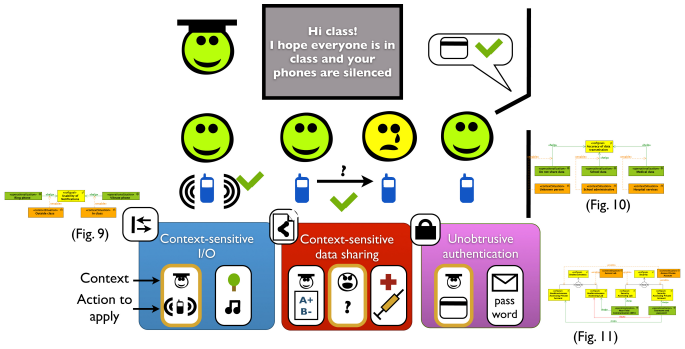

Fig. 4. Ubiquitous e-Learning Scenario after the application of patterns.

presents an extensive catalogue of software requirements patterns, grouped into different categories, where there are specific patterns regarding some NFRs, such as performance or usability. These patterns are broad and not specific of Ubiquitous Computing. Supakkul et al. [11] propose an NFR Pattern approach, also for general-purpose systems. The patterns in this proposal can be located in a meta-level, as proposed in this paper. To the best of our knowledge, there is no previous work regarding patterns for requirements engineering in ubiquitous systems.

The *Context-sensitive I/O* pattern can be defined from Mark Weiser's initial definition of Ubiquitous Computing [12], and has been defined as a design pattern in [5]. Several approaches can be found to provide *unobtrusive identification* [6], as well as to enhance user's privacy through the concept of *pseudonymity* [4], as in the *Context-sensitive Data Sharing* pattern. There is a wide use and application of the presented patterns in several related works, showing their usefulness.

5 Conclusions

Dealing with NFRs for Ubiquitous Systems has to address their unique features, which oftentimes requires the use of a rich body of knowledge. In this paper, we have presented a pattern-based approach to explicitly and clearly capturing, organizing and reusing such knowledge, with promising results in some of our developments [9]. In particular, we have presented three important types of patterns, among a more extensive set of patterns, from our analysis of, and experience in, a number of existing ubiquitous systems. In our experience, their use has been useful in a number of domains, as discussed earlier in this paper. We have also presented several ways to organize the patterns, hence making them amenable to extensions, through *generalization* and *specialization*, *composition*, and *instantiation*. Our systematic treatment of NFRs for Ubiquitous Computing is novel, since it has been explored in little or no previous work. As illustrated in the paper, our approach can help people for a wide variety of systems.

For future work, we are planning to build a repository of patterns, which can be searched and grow with new knowledge. Currently, we are investigating how to achieve a high quality design based on a pattern-based requirements analysis. Also, we are planning to study the implications of these patterns in the software design stage, as well as analyzing those techniques that enable model transformation to (semi-automatically) derive designs and implementations following a model-driven engineering approach.

Acknowledgements

This research work is funded by the Innovation Office from the Andalusian Government through project TIN-6600, the Spanish Ministry of Economy and Competitiveness through the project TIN2012-38600, and The Spanish Ministry of Education, Culture and Sports through the FPU Scholarship.

References

1. L. Chung, B. Nixon, E. Yu, J. Mylopoulos: *Non-Functional Requirements in Software Engineering*. Springer, 2000.
2. E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design patterns: Elements of reusable object-oriented design*. Addison-Wesley Reading, 1995.
3. E. Haselsteiner, K. Breitfuß: *Security in Near Field Communication (NFC)*. Workshop on RFID Security RFIDSec, 2006.
4. J.I. Hong, J.A. Landay: *An architecture for privacy-sensitive Ubiquitous Computing*. Proceedings of the 2nd Int. Conf. on Mobile Systems, Applications and Services, 2004. pp. 177 – 189.
5. J.A. Landay, G. Borriello: *Design patterns for Ubiquitous Computing*. IEEE Computer 2003. vol 36(8), pp. 93 – 95
6. G. Lenzini: *Design of architectures for proximity-aware services: Experiments in context-based authentication with subjective logic*. Electronic Notes in Theoretical Computer Science 2009. vol. 236, pp. 47 – 64.
7. G. Rossi, S. Gordillo, G. Lyardet: *Design patterns for context-aware adaptation*. The 2005 Symp. on Applications and the Internet Workshops, 2005. pp. 170 – 173.
8. T. Ruiz-López, M. Noguera, M.J. Rodríguez, J.L. Garrido, L. Chung: *REUBI: A requirements engineering method for ubiquitous systems*. Science of Computer Programming, 2012.
9. T. Ruiz-López, C. Rodríguez-Domínguez, M. Noguera, J.L. Garrido: *Towards a Reusable Design of a Positioning System for AAL Environments*. Evaluating AAL Systems Through Competitive Benchmarking, 2012. pp. 65 – 79.
10. A.P. Sabzevar, J.P. Sousa: *Authentication, authorisation and auditing for ubiquitous computing: a survey and vision*. International Journal of Space-Based and Situated Computing, 2011. pp. 59 – 67.
11. S. Supakkul, T. Hill, L. Chung, T.T. Tun, J. Leite: *An NFR pattern approach to dealing with NFRs*. Proceedings of the 18th IEEE international requirements engineering conference (RE) 2010. pp. 179 – 188.
12. M. Weiser: *The computer of the 21st century*. Scientific American, 1991. vol. 265(3) pp. 94 – 104.
13. S. Withall: *Software Requirement Patterns*. Microsoft Press, 2010.