A Reward Design Pattern in BCSS

Tuomas Alahäivälä, Michael Oduor and Harri Oinas-Kukkonen

[1]University of Oulu, Department of Information Processing Science
P.O BOX 3000 FI-90014 Oulu, Finland
{tuomas.alahaivala,michael.oduor,harri.oinas-kukkonen}@oulu.fi

Abstract. Although constructs have been developed for designing the features of Behavior Change Support Systems (BCSSs), detailed descriptions and guidelines for their software level implementation are lacking. Through developing software design patterns one is able to examine BCSSs at a more intricate technical level instead of merely a black-box approach to them. In this paper, we present a software design pattern for rewarding users as a way of enhancing persuasive human-computer dialogue in BCSS. The resulting pattern contributes to both research on software design of persuasive system features, and for assisting the practical development of such systems.

Keywords: behavior change supports systems, persuasive technology, persuasive systems design, human-computer dialogue, software design patterns

1 Introduction

Behavior change support systems (BCSSs) have been defined as information systems that form, alter, or reinforce attitudes, behaviors, or acts of complying without using deception or coercion [1]. They can provide solutions for problem areas such as improving health and sustainability. The research into BCSSs focuses on the approaches, methodologies, processes, and tools for their design, as well as their potential effects [1]. There are many features that add to the persuasiveness of a system, such as those contributing to support user's primary task, human-computer dialogue, system credibility or social influence [2]. In this paper we focus on conceptualizing a software design pattern for specifically implementing rewards as a feature of *persuasive human-computer dialogue in BCSSs*. Our study uses the design science research methodology, which includes an iterative process of designing and evaluating a functional IT artifact to produce a solution to the research problem [3].

Although a prominent research area, BCSSs have in prior studies been described at an undetailed technical level [cf. 4–5]. The persuasion context, containing the case-by-case use, user, and technology contexts should be fully regarded when describing a BCSS [2]. Describing systems without knowledge of its internals, or so-called "black-box approach", makes it difficult to argue generalizable results related to systems design [1]. By utilizing more software engineering oriented approaches and tools such as software architectures and software design patterns, BCSS research can be enhanced from proof-of-concepts to concrete software development guidelines. There has been prior research on design pat-

terns for persuasive systems, such as discovering persuasive patterns in social networks and introducing a set of general patterns for influencing user behavior through design [cf. 6–7]. While covering many aspects of persuasive design issues, these patterns have been mostly presented at a generally high conceptual level. We are aiming to reach a more detailed technical view into persuasive systems design by inspecting our patterns also from the object-oriented modeling and code implementation level. This will also make our results presentable to both researchers studying persuasive systems design and practitioners implementing future BCSSs. In this paper we will, based on the background literature on Persuasive Systems Design and software design patterns, present a Reward design pattern for BCSSs.

2 Background

2.1 Rewards in Persuasive Systems Design

Oinas-Kukkonen and Harjumaa's Persuasive Systems Design (PSD) model states that the development of persuasive systems (including BCSSs) requires three steps: understanding the key design issues related to persuasive systems, analyzing the persuasion context, and designing the system qualities [2]. For understanding persuasion in a system, its use, user, and technology contexts should be recognized. The use context covers the characteristics of the problem domain in question, the user context includes the differences between the individuals, and the technology context contains the technical specifications of a system [2]. A lack of precision in describing the technological context has been common in prior studies on BCSSs, making it difficult to understand the persuasiveness of these systems as a whole [1].

Concerning the design of the *software features* of persuasive systems, Oinas-Kukkonen and Harjumaa have proposed four categories of design principles: primary task, dialogue, system credibility, and social support [2]. These design principles may function as guidelines for determining software requirements, as well as an evaluation method for persuasive systems. The dialogue support category contains design principles for system features that concern the dialogue between a system and its users. These features include praise, rewards, reminders, suggestion, similarity, liking and social role. By providing virtual *rewards* a system works as a motivational tool [2]. In this paper we will focus on the rewards feature of persuasive systems.

2.2 Software Design Patterns

Patterns are reusable solutions that can be applied to commonly occurring problems in software design and enable building of systems with good object-oriented design qualities [8]. They do not provide the code, but rather provide solutions to general design problems, which are to be applied to specific applications – a solution to a problem in context. They serve as templates that can be used in different ways for solving problems. Most patterns allow some part of a system to vary independently of all other parts and these varying parts are often encapsulated.

Use of patterns provides a shared language that maximizes the value of communication amongst developers and reduces the time spent on making design decisions related to feature changes and enables reuse of solutions that have previously been effective. Furthermore, they aid in avoiding design alternatives that compromise reusability and they can improve documentation and maintenance of existing systems by providing an explicit specification of class and object interactions and their underlying intent [8–9].

Patterns depict the static and dynamic structures and collaborations of successful solutions to problems–discerning of non-functional features for example–that arise when developing applications within a particular context. Patterns (or their solutions) should be applicable in many different situations without the need to make extensive changes as they provide ways to arrange and categorize relatively mundane solutions in technology-related development projects. According to Gamma et al. [8], patterns have four essential characteristics:

1. The **pattern name** – a common term that eases the communication amongst stakeholders and enables design at a higher abstraction level while simplifying thoughts on designs and communicating these and their trade-off to others.
2. The **problem** describes when a pattern should be applied and its context.
3. The **solution** provides an abstract description of a design problem and how a general arrangement of elements (classes and objects) solves it.
4. The **consequences** are the results and tradeoffs of applying the pattern

According to Buschmann et al. [10], there currently is a common set of well-known generic software patterns but when looking toward future developments, patterns could be more domain-specific and tailored to particular focus areas. Many domain areas such as behavior change support systems are yet to be properly covered by the pattern languages.

3 Reward Design Pattern for BCSSs

Rewards and virtual achievements are powerful motivational tools. A reward system can make the process more enjoyable and help users get pleasure from their tasks [cf. 11–12]. Rewards have an effect on intrinsic motivation, although depending on the way they are delivered [13]. We now present a design pattern for implementing reward features in behavior change support systems. For issuing virtual rewards in a web-based BCSS, the performance of its users must be monitored. This can be efficiently done utilizing the well-known object-oriented *Observer* design pattern [8], which defines and maintains a one-to-many dependency between objects such that a change in one object leads to all its dependents being notified and being updated automatically.

Building on Model-View-Controller (MVC) [14] and Representational state transfer (REST) [15] approaches, we presume that the application's resources are implemented as their corresponding Models, Views, and Controllers with Create, Read, Update, and Delete (CRUD) actions. There are at least two generalizable resource entities that are necessary for a BCSS: the User resource and the Entry resource. The User resource depicts the users of the systems, containing their account information and possible behavioral profiles. The Entry resource is an abstraction of the data that the user submits to the system to monitor her behavioral habits – for example weight measures in a weight monitoring application, or individual exercise activities in exercise a tracking application.

Hence, for providing the rewards functionality in BCSS, *User*, *Entry*, *EntryObserver*, *Reward*, and *Accomplishment* components are needed as seen in the class diagram (see Figure 1). In the diagram, theUser model contains the profile information of a certain user, User has a one-to-many relationship to the Entry–being an actualization of a pursued behavioral habit–model: an Entry depicts an action that the user submits to the system,, The Reward model contains the description of the reward in question. The Accomplishment model depicts the many-to-many relationship between the User and the Reward and is used for maintaining the record of the rewards users have gained. The EntryObserver class then contains the logic that observes upon creation of Entries, if they account for a reward and if so, creates the corresponding Accomplishment. See Table 1 for summarization.

Table 1. Reward pattern

Pattern name	*Reward*
Problem	The system should give virtual rewards to users to further motivate them to stay involved.
Components	User, Entry, EntryObserver, Reward, Accomplishment
Solution	The resources in the system should be modeled to implement the User, Entry, EntryObserver, Reward and Accomplishment components. When the User submits an Entry to the system, the EntryObserver component observers whether the action is eligible for issuing a reward to the user.
Consequences	Rewarding users for performing after their target behavior motivates them and assists their goal setting. But it should be minded that all users might not find rewards as desirable.

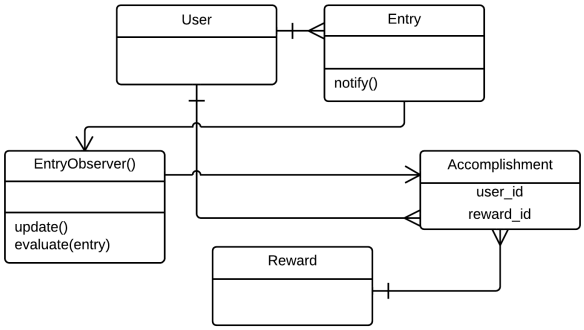

Figure 1. Class diagram of the Reward pattern

4 Conclusion and discussion

In this study we have proposed the Reward software design pattern to facilitate enhanced computer-human dialogue in behavior change support systems, based on the PSD model. The paper's implications for research are in providing an intricate implementation level view of the software development aspects of BCSSs. We hope that the object-oriented design and code level findings presented will result in breaking out from the black-box thinking approach in persuasive systems design, allowing researchers to inspect the internals of software components needed to produce persuasive applications. In the future, a full set of design patterns for BCSSs could be accomplished. Practitioner-wise, using the pattern will assist in creating conventions to bootstrap future BCSSs development. The pattern can be used to add rewarding features to existing behavior change support systems, thus potentially increasing their persuasiveness. This study is limited by the fact that the verification of the implied pattern was conducted only as describing the development of a demonstrative system prototype. For further proof, more complex applications, which apply the pattern, should be developed. The application of the pattern in different programming environments, languages, and frameworks should also be studied. For example, whether the pattern applies in the development of a native mobile application as well as of a web-based BCSS could be studied. The presented pattern solely concerns the rewarding features in a system, and there still remain many other persuasive system features that should be covered when studying persuasive software design patterns. Thus, the future work will include further definition and verification of the presented pattern and developing new ones focusing on both human-computer dialogue and the other aspects of persuasive systems design, such as social influence.

Acknowledgements. This research is part of OASIS research group of Martti Ahtisaari Institute, University of Oulu.

References

1. Oinas-Kukkonen H. (2013). A foundation for the study of behavior change support systems. *Personal and ubiquitous computing*, Vol. 17, No. 6, August 2013, pp. 1223-1235.
2. Oinas-Kukkonen, H., Harjumaa, M. (2009). Persuasive systems design: Key issues,process model, and system features. In: Communications of the Association forInformation Systems, 1, vol. 24, pp. 485-500.
3. Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. MIS Quarterly: Management Information Systems, 28(1), 75-105.
4. Bennett, G.G., Glasgow, R.E. (2009). The Delivery of Public Health Interventions Via the Internet: Actualizing their Potential. Annual Review of Public Health, vol. 30, 273–292.
5. Lehto T., Oinas-Kukkonen H. (2011). Persuasive Features in Web-Based Alcohol and Smoking Interventions: A Systematic Review of the Literature. In: Journal of Medical Internet Research, 3, vol. 13, e46.
6. Weiksner, G., Fogg, B., Liu, X. (2008). Six patterns for persuasion in online social networks. *Persuasive Technology*.
7. Lockton, D., Harrison, D., & Stanton, N. (2010). The Design with Intent Method: a design tool for influencing user behaviour. *Applied ergonomics*, *41*(3), 382–92. doi:10.1016/j.apergo.2009.09.001
8. Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley.
9. Zhu, Z. (2009). Study and application of patterns in software reuse. Control, Automation and Systems Engineering, 2009. CASE 2009. IITA International Conference on, 550-553.
10. Buschmann, F., Henney, K., & Schmidt, D. C. (2007). Past, present, and future trends in software patterns. *Software, IEEE, 24*(4), 31-37.
11. Ritterband, L., Thorndike, F., Cox, D., Kovatchev, B., & Gonder-Frederick, L. (2009). A behavior change model for internet interventions. *Annals of Behavioral Medicine, 38*(1), 18-27. doi:10.1007/s12160-009-9133-4
12. Sohn, M., & Lee, J. (2007). UP health: Ubiquitously persuasive health promotion with an instant messaging system. *CHI '07 Extended Abstracts on Human Factors in Computing Systems,* San Jose, CA, USA. 2663-2668.

13. Deci, E. L. (1971). Effects of externally mediated rewards on intrinsic motivation. *Journal of Personality and Social Psychology*, 18(1), 105-115.
14. Krasner, G., Pope, S. (1988) A Description of the {Model-View-Controller} User Interface Paradigm in the Smalltalk-80 System. In: Journal of Object Oriented Programming, 3, vol. 1, pp. 26-49.
15. Fielding, R.T., Taylor, R.N. (2002) Principled design of the modern Web architecture. In: ACM Transactions on Internet Technology, 2, vol. 2, pp. 115-150.