Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 771–775

Procedia Computer Science

International Symposium on Frontiers in Ambient and Mobile Systems (FAMS)

Towards Pattern-based Generic Interaction Scenarios

Bettina Biel[a,*], Stefan Seitz[b], Volker Gruhn[a]

[a] *University Duisburg-Essen, paluno – The Ruhr Institute for Software Technology, Essen, Germany*
[b] *University of Munich, Media Informatics Group, Munich, Germany*

Abstract

Usability is crucial for the user acceptance of software products. Some usability requirements influence software architecture and therefore should be considered early in the software development process. In this short paper, we propose the design of generic interaction scenarios that are mined from best practices (patterns). Our scenarios describe a user or system interaction as stimulus, the response of the system, and its assessment. We classified them in order to provide them in a knowledge base, as they are used for scenario-based analyses that assess usability properties of mobile applications.

Keywords: Usability evaluation, Interaction scenarios, Patterns, Knowledge base

1. Introduction

The usability of mobile applications determines their commercial success considerably. Usability is the extent to which a product can be used by specified users to achieve specified goals in a specified context of use [1] in an effective, efficient, safe, useful, easy-to-learn and memorable way [2]. It is a quality attribute of the usage of a software product, not of a quality attribute of the software itself [3]. Interaction design describes the information exchange between a user and a system with input and output channels as the user interface. Interface and interaction design must be supported by the software architecture design [4, 5, 6]. If not, usability related modifications of the architecture late in the software development process are time-consuming and costly. Thus, such requirements must be identified and considered early in the software development process.

Scenarios are an established means to design and analyze software systems. For example, software architecture design, interaction design and user-evaluations are based on usability requirements that can be expressed as scenarios.

We propose the design of generic interaction scenarios that are mined from best practices (interaction design patterns) and to be provided in a knowledge base. As we apply these scenarios in the scenario-based software architecture analysis SATURN [7], our interaction scenarios are designed to be easy to understand by stakeholders and software architects without usability expertise or software technical background and thus provide an appropriate degree of abstraction and the exclusion of technical aspects. Scenarios cover user or system initiated interactions with software in general, including mobile applications.

*Corresponding author

Email addresses: bettina.biel@paluno.uni-due.de (Bettina Biel), seitz@cip.ifi.lmu.de (Stefan Seitz), volker.gruhn@paluno.uni-due.de (Volker Gruhn)

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas.
doi:10.1016/j.procs.2011.07.103

This short paper presents the proposed design of our scenarios, by discussing related work in Section 2, defining necessary elements in Section 3. We give an example how scenarios are mined from patterns. We conclude the paper in Section 4.

2. Related Work

The *general usability scenarios* of Bass et al. [8] describe interactions that can occur in applications. Rafla et al. [9] show that the short examples in the scenarios of Bass et al. [8] do not sufficiently explain themselves. Golden et al. [6] extend these scenarios by a list of general responsibilities and an implementation example. We believe it is necessary to extend these scenarios further to provide context information, because usability depends on the usage context. Also, scenarios should be described from a user perspective and not using quality attributes of the usage of a system. Scenarios should refer directly to the patterns they are derived from in order to support understanding.

Juristo et al. [10] propose *usability elicitation patterns* for gathering functional usability requirements for the architecture from usability non-experts. The researchers analyzed usability patterns of different authors (e.g., Tidwell [11], van Welie [12]) for underlying usability mechanisms. An elicitation pattern lists questions that help to specify more detailed requirements for one mechanism. It serves as checklist during the gathering of usability requirements. However, non-experts need to learn and apply usability concepts, and stakeholders should focus on their own know-how. We believe it is necessary to renounce learning usability concepts or discussing requirements from an expert perspective, and rather provide usability knowledge "undercover" in the form of interactions from a user perspective.

Schmettow and Niebuhr [13] propose a *pattern-based usability inspection method* directed at software developers who are often usability non-experts. The work is primarily based on existing usability patterns. The reuse of knowledge from usability patterns, a categorization based on the user perspective, checklist functionalities, and a constructive contribution to the design of interactions are important aspects. We extend the approach by using the language and the perspective of the user by describing immediate interactions, instead of providing patterns whose contents must be understood first; they present ideas for interactions and serve as a template for concrete requirements.

The *problem frames* designed by Jackson [14] use frame diagrams to describe and structure user requirements and to derive instances of well-known and simple problem frames. In order to support problem solving, they can be assigned to software architecture patterns [15]. Specker and Wentzlaff [16] propose a variant, HCI Frames, representing descriptions of problem categories supplemented by additional usability functionality. The extended information is based on usability principles derived from usability patterns. Problem frame diagrams are a formal language with a precise problem description that allows a more accurate understanding of parts of the problem area. However, they appear unsuitable for a discussion of requirements with the kind of stakeholders that we regard in our research approach: They would have to learn a formal language and need to understand and adopt the technical and abstract way of thinking, which does not represent an intuitive approach. Where stakeholders are not concerned with software development on a regular bases, their command of the formal language is useless to them. Specker and Wentzlaff [16] refer to usability patterns for more details about interaction design, an idea that we follow by referring to usability patterns in the generic scenarios.

3. Generic Interaction Scenarios

Elements of a Scenario A generic interaction scenario captures a usability requirement. Scenarios describe stimuli and in what particular way the system should respond to them. The system is seen as a black box, a closed entity without considering its inner design. This abstraction is necessary to maintain focus to refrain from discussing details too early. Our scenarios are based on Bass et al. [5], however our scenarios do not state a usability attribute like *learnability* as a goal, and we believe that brainstorming scenarios from such abstract usability attributes is very difficult without expert knowledge or training. We propose scenarios which describe user- or system-initiated interactions with the system and its reactions to them. Stimuli of our scenarios are goals such as "a user *wants to know what the system is doing*". Generic scenarios are tailored for an application during software architecture analysis SATURN in phase two [7]. Table 1 lists and explains elements of our generic interaction scenarios.

Name	Description
Number, name	Number of the scenario and a short name for quick reference
Source	Interaction can be user-initiated or system-initiated
Stimulus	Short and precise descriptions of concrete interactions or goals
Artifact	Software system under evaluation as a black box
Environment	Environments a system is used in and technical conditions (e.g., runtime, high server load)
Response	Expected reaction of the artifact to the stimulus
Response Measure(s)	Measurable quality to analyze response, either by expert evaluation/SA analysis, or user test;
Pattern(s)	References to patterns the scenario was derived from
Interaction category/ies	13 Interaction classes (e.g., Acquiring and Processing User Input; Error Handling and Help; Executing, Repeating and Revoking Commands; Adaptation to by/to Users, to Tasks)
Usability-Goal(s)	Characterization of the scenario's impact on usage such as effectiveness, efficiency, safety, utility, learnability and/or memorability. User experience goals are excluded from the list due to the causal dependence of user experience from usability among others.
Potential Architecture Sensitivity	Potential impact on software architecture if scenario is unsupported
Potential Tradeoffs	Potential tradeoffs to other quality goals such as performance, security, modifiability

Table 1: Elements of a generic interaction scenario

Sources of scenarios Our scenarios[1] are mined based on an approach by Zhu et al. [17], who extracted and systematically documented scenarios from software architecture patterns to support the software architecture evaluation process. Architecture patterns offer proven solutions for architecture design problems in a certain context [18].

In order to be valid, a scenario must be based on at least one pattern that is proven to discuss problems and solutions relevant in practice. That means, this process demands to search for or to write a new pattern and evaluate it before a new scenario can be added (e.g., a new pattern describing a new context-aware interaction and a new context-aware architecture). Guidance on how to write patterns is available from the pattern community (for example, [21]). And a pattern life cycle for using and maintaining mobile interaction design patterns in a pattern library is provided by [22].

Regarding our sources, we focus on research results in the field of the relationship between software architecture and usability and added interaction design patterns in order to prepare analyzing their architectural impact in the future:

- *Interaction design patterns* by Tidwell [11] cover a wide range of usability issues, (interactions only, excluded principles, aesthetics: 75 of 94 usability patterns);
- *Architecture-sensitive usability patterns*, *Bridging patterns* by Folmer et al. [4, 19] cover four extensive and 15 brief usability patterns;
- *General usability scenarios* and *architecture sensitive usability patterns* by Bass et al. [8], and Golden et al. [6] cover 26 usability issues with recorded impact on software architecture;
- *Interface design patterns* by Little Springs Design [20] present 32 usability patterns or outlines particularly focusing on mobile applications.

Mining scenarios The mining of scenarios is accomplished by (1) highlighting and categorizing text passages: From the problem part of a pattern, stimulus, source of stimulus, and environment can be derived, while response and artifact can be found in the solution part. (2) Text passages are extracted and combined. The wording is reviewed and updated to improve readability and understandability. Afterwards, (3) the main idea of the solution of each pattern on

[1] Currently, there are 50 generic interaction scenarios available via http://www.saturn-lab.com/scenarios. [24]

which a scenario is based on is summarized and the pattern is referenced. Finally, (4) scenarios are categorized based on the content of these patterns.

Table 2 exemplifies the development of the scenario *Cancel*, which is based on the usability patterns *Cancelability* [11], *Cancel* [4] and *Cancelling Commands* [8]. As the highlighted text passages reveal, *users* (source) *cancel an operation* (stimulus) *because they do not want its further execution* due to an opinion change (stimulus). The command should be *cancelled immediately* (response). The system *effects* and *communicates the discontinuation* of the execution *within a limited time frame* (response measures). Scenario number 4 of interaction category *Executing, Repeating and Revoking Commands* supports usability goals *Effectiveness, Efficiency, Safety, Learnability*. If Cancel is unsupported by an application its architectural impact is *high based on literature (architectural patterns)*. Potential tradeoffs are security, trust, fault tolerance, reliability.

Pattern Part from [11]	**Extraction**
"'What"'	"'Provide a way to *instantly cancel*[3,4] a time-consuming *operation*[2], with *no side effects*[4]"'
"'Why"'	"'*Users*[1] *change their minds*[2]. Once a time-consuming *operation starts*[2], a user may *want to stop it*[2], especially if a Progress Indicator tells her that it'll take a while. Or the *user*[1] may *have started it*[2] by accident in the first place. Cancelability certainly helps with error prevention and recovery – a user can cancel out of something she knows will fail, like loading a page from a web server she realizes is down. (...)"'
"'How"'	"'(...) Put a *cancel button*[3] directly on the interface, next to the Progress Indicator (...) or wherever the results of the operation appear. (...) When the *user*[1] *clicks or presses the Cancel button*[2], *cancel the operation*[3] *immmediately*[4]. If you wait too long, for more than a second or two the user may doubt, that the cancel actually worked (or you may dissuade him from using it, since he might as well wait for the operation to finish). *Tell the user that the cancel worked*[4] – halt the progress indicator, and show a status message on the interface, for instance. (...)"'

Table 2: Extraction of the description of the scenario Cancel (1 – source, 2 – stimulus, 3 – response, 4 – response measure)

Mobility All interaction scenarios are reviewed regarding their applicability for interacting with mobile applications that run on mobile devices and are used in a mobile environment [23]. We found that all currently available scenarios are applicable in mobile as well as in stationary systems. For example, scenario *Standard Softkey Behavior* describes a typical interaction with mobile devices, but it is closely related to scenario *Short Cuts*; scenario *Long Page Scrolling Information*, at first glance, might not be useful for a small screen device, but it can be considered for an application providing a long text such as an electronic book.

The context of use of a particular mobile application is especially important. In order to be applicable to different mobile applications, the generic interaction scenarios provide only general information on usage context (user, task, environment, artifact). It is thus very important that a generic scenario is instantiated for a particular mobile application and its particular usage context.

Accordingly, our scenario-based software architecture analysis SATURN [7] applies a usage context analysis in the first phase. Participants of the analysis specify parameters of the usage context factors user, environment, device, task, application. These are provided as a checklist based on a literature study and a SATURN case study; they can be modified and extended. In phase two of the analysis, the specified usage context is employed to select interaction scenarios from the scenario catalog (e.g., single interactions as part of tasks) and to tailor a scenario selected for analysis by adding user types, environment types, device types, other information useful for analysis (such as reasons for selecting a scenario for analysis), and by adapting the wording of the scenario itself.

4. Conclusion

We proposed the design of generic interaction scenarios for the use during design and analysis of mobile applications. Scenarios are based on [5], however, we adapted the schema of the scenarios in several ways. Most importantly, our scenarios describe user- and system initiated interactions as stimulus. For example, stimuli of our scenarios are goals such as "a user *wants to know what the system is doing*" instead of goals such as "learnability".

In our work in progress, we mine more scenarios and evaluate how architects use them during our software architecture analysis SATURN [7] which addresses mobile applications.

References

[1] International Standard Organization, ISO 9141-11: Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidelines on usability, International Organization for Standardization, 1998.
[2] H. Sharp, Y. Rogers, J. Preece, Interaction Design: Beyond Human-Computer Interaction, 2nd Edition, Addison-Wesley, 2007.
[3] N. Bevan, International standards for HCI and usability, International Journal of Human Computer Studies 55 (4) (2001) 533–552.
[4] E. Folmer, J. V. Gurp, J. Bosch, A framework for capturing the relationship between usability and software architecture, Software Process: Improvement and Practice (2003) 67–87.
[5] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, 2nd Ed., SEI Series in Software Engineering, Addison-Wesley, 2003.
[6] E. Golden, B. E. John, L. Bass, The value of a usability-supporting architectural pattern in software architecture design: a controlled experiment, in: Proceedings of the 27th International Conference on Software Engineering (ICSE' 05), ACM, New York, NY, USA, 2005, pp. 460–469.
[7] B. Biel, T. Grill, V. Gruhn, Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application, Journal of Systems and Software 83 (11) (2010) 2031 – 2044.
[8] L. Bass, B. E. John, J. Kates, Achieving usability through software architecture, Technical report, Software Engineering Institute, Carnegie Mellon University (2001).
[9] T. Rafla, P. N. Robillard, M. Desmarais, A method to elicit architecturally sensitive usability requirements: its integration into a software development process., Software Quality Journal 15 (2007) 117-133.
[10] N. Juristo, A. M. Moreno, M.-I. Sanchez-Segura, Guidelines for eliciting usability functionalities, IEEE Transactions on Software Engineering 33 (11) (2007) 744 – 758.
[11] J. Tidwell, Designing Interfaces, O'Reilly, 2005.
[12] M. van Welie, Welie.com patterns in interaction design, http://www.welie.com (last visited 4 April 2011).
[13] M. Schmettow, S. Niebuhr, A pattern-based usability inspection method: first empirical performance measures and future issues, in: B. C. Society (Ed.), Proceedings of the 21st BCS HCI Group Annual Conference (HCI) Lancaster, UK, Vol. 2, 2007, pp. 99–102.
[14] M. A. Jackson, Problem Frames. Analyzing and Structuring Software Development Problems., Addison-Wesley, Harlow, GB, 2001.
[15] C. Choppy, D. Hatebur, M. Heisel, Architectural patterns for problem frames, IEE Proceedings - Software, Special Issue on Relating Software Requirements and Architectures 152 (4) (2005) 196–208.
[16] M. Specker, I. Wentzlaff, Exploring usability needs by human-computer interaction patterns, in: Springer (Ed.), Proceedings of the Sixth International Workshop on TAsk MOdels and DIAgrams (TAMODIA). Toulouse, France, 2007, 254–260.
[17] L. Zhu, M. A. Babar, R. Jeffery, Mining patterns to support software architecture evaluation, in: Proceedings of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA' 04), 2004.
[18] P. Avgeriou, U. Zdun, Architectural patterns revisited – a pattern language, in: Proceedings of the 10th EuroPLoP 2005, 2005, pp. 1–39.
[19] E. Folmer, M. van Welie, J. Bosch, Bridging patterns – an approach to bridge gaps between SE and HCI, Journal of Information and Software Technology 48 (2006), 69–89.
[20] Little Springs Design, Design for mobile: resources for designing and building mobile apps and sites, http://patterns.littlespringsdesign.com (last visited 4 April 2011).
[21] G. Meszaros, J. Doble, A pattern language for pattern writing, http://www.hillside.net/index.php/a-pattern-language-for-pattern-writing.
[22] T. Grill, B. Biel, V. Gruhn, A pattern approach to mobile interaction design, it - Information Technology 51 (2) (2009) 93–101.
[23] J. Kjeldskov, C. Graham, A review of mobile HCI research methods, in: Mobile HCI, 2003, pp. 317–335.
[24] B. Biel, S. Seitz, The saturn laboratory scenarios, http://www.saturn-lab.com/scenarios (last visited 4 April 2011).