Exploiting the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application

Bettina Biel a, Thomas Grill b, Volker Gruhn a

a Software Engineering Group, The Ruhr Institute for Software Technology, University of Duisburg-Essen, Germany
b ICT&S Center, University of Salzburg, Sigmund-Haffner G. 18, 5020 Salzburg, Austria

Abstract
Designing easy to use mobile applications is a difficult task. In order to optimize the development of a usable mobile application, it is necessary to consider the mobile usage context for the design and the evaluation of the user–system interaction of a mobile application. In our research we designed a method that aligns the inspection method “Software Architecture analysis of Usability Requirements realization” (SATURN) and a mobile usability evaluation in the form of a user test. We propose to use mobile context factors and thus requirements as a common basis for both inspection and user test. After conducting both analysis and user test, the results described as usability problems are mapped and discussed. The mobile context factors identified define and describe the usage context of a mobile application. We exemplify and apply our approach in a case study. This allows us to show how our method can be used to identify more usability problems than with each method separately. Additionally, we could confirm the validity and identified the severity of usability problems found by both methods. Our work presents how the increased quantity and quality of results can lead to a reduction of the number of iterations required in early stages of an iterative software development process.

1. Introduction

Mobile applications that run on mobile devices and are used in a mobile environment (Kjeldskov and Graham, 2003) become more and more important. A recent United Nations report shows that more than half of the global population use a mobile phone. Despite the economic downturn, the prognosis given in the report is positive due to the “pervasive nature of information and communication technologies” (International Telecommunications Union (ITU), 2009). Although technological progress has been made – mobile devices are enhanced with more computing power, connectivity, and interaction capability – many limitations and challenges still remain, for example regarding usability.

Usability is defined by ISO 9241-11 as “the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use” (International Standard Organization, 1998). It is therefore a quality attribute of the usage of a software product. A product has no intrinsic usability, it has a capability to be used in a particular context (Bevan, 2001). “Design for usability” incorporates product characteristics that benefit users in a particular usage context.

Considering the usage context of a mobile application, in literature also found as the mobile usage context, limitations and challenges become apparent. Parameters like a changing and distracting environment, mobile devices’ small screens and non-traditional input capabilities as well as task limitations (Hassanein and Head, 2003) increase the complexity of designing usable mobile applications. The mobile usage context is analyzed and described based on mobile context factors (see Section 3) which describe the characteristics of the particular usage context. In the research area of mobile human computer interaction (mobile HCI) the mobile usage context is addressed, and usability evaluations (UE) in the form of user tests are applied to find problems appearing during the interaction between a human and an application’s interface. But there is a lack in mobile usability research related to the technology beyond the interface (Coursaris and Kim, 2007), the technology that realizes interaction. Research shows that software architecture plays an important role in designing usable systems by separating the user interface (UI) from the core functionality and by addressing user interactions that demand architectural support (Clements et al., 2002; Bass et al., 2003; Folmer and Bosch, 2004; Juristo et al., 2007b).
The software architecture (SA) defines the scope and constraints for the quality of a software product. Kruchten defines SA as “[...] the set of significant decisions about the organization of a software system, the selection of the structural elements and their interfaces by which the system is composed, together with their behavior as specified in the collaborations among those elements, the composition of these structural and behavioral elements into progressively larger subsystems, and the architectural style that guides this organization—these elements and their interfaces, their collaborations, and their composition” (Kruchten, 1998). According to this definition, software architecture has an effect on the behavior of a system. The behavior of a system towards users and vice versa is also known as user–system interaction. Interaction and user interface influence usability. Thus usability is a quality attribute that does not only depend on the interface and its design but also on the software architecture (Bass et al., 2003). It is closely related to other quality attributes such as performance, reliability, security, trust, and others that equally need to be regarded when designing an application and its interface.

Our approach combines the methods of software architecture analysis and usability evaluation to exploit the benefits of both methods. We designed a method that analyzes the software architecture of a mobile application for the design of usability and validates its results with a usability evaluation. The usability evaluation finds usage problems of a software application by observing and monitoring users to identify user performance issues and usability problems. It can use the SA analysis to locate problems identified. We use the method “Software Architecture analysis of Usability Requirements realization” SATURN (see Section 4.1) that analyzes the SA of an application. It finds architectural risks that are based on how usability requirements are realized in the software architecture. Further, we conduct a laboratory user test that evaluates a software. Afterwards we combine, compare, and evaluate the results of both methods.

The common basis for both evaluations is the usage context. Focusing on mobile applications, we identify mobile context factors (MCF) representing the parameters that have an effect on the interface and the interaction design and thus the usability of a mobile application.

Contributions of this paper are: (1) in the method, the results of a SA Analysis and a UE (see Section 4) are compared. It shows how SA analysis and a UE method can be combined and how results relate to each other. We evaluate this method in a case study in which we analyze and evaluate a mobile application. (2) We define the parameters determining the usage context of a mobile application (see Section 3). (3) In the case study, we identified specific usability problems and their architectural reasons during a SA analysis. We were able to confirm these problems and define their severity through the usability evaluation. By this the importance of considering usability in the SA is highlighted (see Section 5). (4) By combining the evaluation and the analysis, we identified more usability problems than with each method separately. We could confirm the validity of problems by using the combination of both methods. This enhances our results by the possibility of verifying the severity of the particular problems found.

Compared to the methods currently used (see Section 2), which do not combine software architecture and usability evaluation methods, we can regard more results earlier in an iterative software development process. Thus, we argue that the improved quality and quantity of our evaluation results lead to a reduction of the number of iterations required. We conclude that the combination of a SA analysis and a usability evaluation improves the current state of the art of evaluation methods applied in software architecture as well as in usability evaluations.

This paper is organized as follows: In Section 2 related work regarding the relationship between usability and software architecture, software architecture analysis methods, and usability evaluation methods is covered. Afterwards (in Section 3), we define mobile context factors and the method. The evaluation setup, evaluation methods used and how their results can be compared will be shown in Section 4. In Section 5, the evaluation of the software “Shake Your Mac” (SYM) is described and the results of the SATURN method applied to SYM and the usability evaluation are compared. We conclude the paper discussing the results and giving an outlook for future work in Section 6.

2. Related work

2.1. Software architecture and usability

Regarding the architectural support of user interfaces and user–system interaction, two approaches are followed. The first approach separates the user interface (UI) from the core of the system. Examples illustrating this strategy are architecture patterns such as Model View Controller and Presentation Abstraction Control (Buschmann et al., 1996). Yet, by the separation of the UI and the system alone, not all necessary aspects of user–system interaction can be realized.

The second approach focuses on usability issues that are difficult to implement late in a software engineering process. Based on literature studies, discussions, and personal experience, Bass and John (2003) identified usability issues and described them in a bottom-up way in a collection of usability scenarios and architectural tactics.

In the EU-project “Software Architecture that supports Usability” (STATUS), researchers investigated the relationship between architecture and usability in a top-down way. Folmer et al. (2003) presented a Software-Architecture-Usability framework in which “usability patterns” are combined with “usability properties” and describe the relationship between usability and software architecture. The properties represent general usability requirements that were derived from a comprehensive literature survey. The patterns show abstract design solutions for a usability problem. Juristo et al. (2007a) argue that it is important to analyze and quantify the impact of usability on architecture. They identified and examined HCI recommendations and selected functional usability features. The resulting selection of recommendations and usability features were in the following applied in six case studies where the usability features were added to architectural designs, after the initial design was accomplished. By doing so, they showed that these features have an impact on design.

We approach this field from an analysis perspective to further investigate the relationship between SA and usability. We accomplish an SA analysis regarding the usability of a mobile application and validate its results with the results of a usability evaluation. The goal is to enhance the usability of a mobile application by finding usability problems. Our approach differs from existing approaches by focusing on mobile applications while current software architecture analysis methods provide a general approach to software architecture analysis and the identification of structural problems in the software architecture.

2.2. Software architecture analysis

Software architecture analysis methods are applied in an early design phase of a software project and are used to analyze first designs based on the requirements with respect to the software architecture proposed.

Associated with the STATUS project, Folmer and Bosch presented the Scenario-based Architecture Level Usability Analysis (SALUTA). It represents the first complete method for usability
assessment on architectural level. They define user profiles in the form of scenarios, and for each the impact on improving usability is ranked by a software architect. Based on a selection of scenarios, the software architecture is analyzed and evaluated. The results of the SALUTA method depict a relation between user interface design and software architecture (Folmer et al., 2004).

The most mature analysis method is the Architecture Trade-off Analysis Method (ATAM) (Clements et al., 2002) in which many different quality attributes besides usability are regarded and the findings of Bass and John (2003) integrated. ATAM is scenario-based, considers stakeholders of the system, and develops scenarios from their perspectives. The scenarios are described informally and capture uses of the system, which are related to quality attributes. ATAM-evaluators define and discuss context and goals of the analysis, use an approved description of analysis scenarios, and apply architectural styles and tactics as a reference.

The “Software Architecture analysis of Usability Requirements realization” SATURN (Biel and Gruhn, 2010) is based on ATAM and includes the usage context which helps to select analysis scenarios. The method refrains from using abstract usability attributes and motivates scenarios based on usability requirements instead. Concrete architecture support levels are determined based on the architectural support of usability requirements for the selected analysis scenarios. Scenarios and design patterns form a knowledge base applied in the method. The method presented in this paper shows how these SA analysis results can be validated by a usability evaluation.

2.3. Usability evaluations of mobile applications

When evaluating the usability of mobile applications, different methods, like heuristic evaluations as rule-based evaluations performed by usability experts and user tests, are applicable. User tests include selected end-users as test persons who are usually accompanied by usability evaluators. Depending on the required context that needs to be evaluated, such evaluations are conducted in laboratory settings or as field studies (Scholtz, 2006). Empirical methods applied are, for example, observations, questionnaires, and experiments.

Po et al. (2004) compared different heuristic evaluation methods for a mobile application. As with each method different usability problems could be found, she concluded that heuristic evaluations depend on the quality of the evaluators participating and the scenarios applied. Kjeldskov and Graham (2003) surveyed evaluation methods that focus on user-tests with respect to mobile applications. Due to the mobile context that such applications are usually used in, mobile applications can be evaluated in the field. Conducting field evaluations requires a lot of effort, while the more precise results that can be gained that way do not outweigh higher costs and efforts needed compared to laboratory evaluations (Kjeldskov et al., 2004a).

Depending on the type of evaluation (formative or summative Scholtz, 2006) and the current stage of the design phase, results of usability evaluations differ extremely. In laboratory evaluations and evaluation scenarios where data can easily be gathered, quantitative data is collected, analyzed, and related to certain problems (Kjeldskov et al., 2004b). In field studies, often qualitative data and reports from users and evaluators are the only source of information. In long-term studies mobile applications can be used to provide logs for recording interactions of the mobile user to be observed. They can also be applied for remote usability studies of mobile devices (Paternò et al., 2007; Hartson et al., 1996).

Our research differs from existing work by combining the methods of software architecture analysis and usability evaluation of mobile applications to make use of the benefits of both methods. A mobile usage context on which we base a mobile application scenario is analyzed to identify mobile context factors. The parameters influence both evaluation methods and affect the design and thus the usability of a mobile application.

Existing SAA and usability evaluation methods do not reflect such a combination and do not benefit from each other. This potential is currently unused and provides an essential improvement potential for optimizing software development processes.

In the next section the mobile context factors that define the usage context and that are regarded in both SAA analysis and usability evaluation are described.

3. Mobile context factors

The usage context of a mobile application is defined by mobile context factors (MCFs). It shows the complex dependencies of a mobile interactive system that software and usability engineers are confronted with when designing an interactive mobile application.

For users being mobile “... means that user location, physical, and social context may change, that physical resources cannot be relied upon, and that physical world navigation may have to be accomplished” (Ballard, 2007). From a technology perspective, Pandya differentiates between device mobility, meaning a device is moved physically, user mobility, meaning that users are not restricted to a single device and/or location, and service mobility, which is given when a service can be accessed from anywhere (Pandya, 1999). The research area of software engineering conceptualizes mobility as change of the location of computational components (Roman et al., 2000).

When evaluating mobile applications, the observed parameters are important to the outcome of the evaluation. Coursaris and Kim (2006) present a framework of the contextual factors environment, user, task, and technology. It is based on a survey of empirical mobile usability studies. They argue that the results of mobile usability evaluations benefit from an emphasis on context and a discussion about the technology “beyond the interface”, which means how technology components such as network connectivity, reliability, or memory impact the usability of a device. Tamminen et al. (2004) elaborates on mobile context parameters influencing the mobile usability of an application. Carter and Mankoff (2004) researched challenges of conducting usability evaluations in ubiquitous environments. Due to the context applied, they identified parameters that are difficult to handle and need to be carefully addressed during the design of a usability evaluation in ubiquitous and pervasive environments: the metrics applied, the scalability of the scenario, the ambiguity of collected data, and the unobtrusiveness of a system integrated in our daily lives. Hummel et al. (2008) and Thurnher et al. (2006) elaborate on the exploitation of context parameters that measure environmental parameters and their influence on the usage of mobile applications in a real world scenario. Based on this and the research of Schmidt et al. (1999), Tarasewich (2003), Varshney and Vetter (2002), we identified the contextual factors environment, users, tasks, devices, and applications.

3.1. MCF—mobile environment

The term environment can be defined by the circumstances, objects, or conditions by which one is surrounded. Parameters which complement this snapshot-view comprise information about the mobile environment itself. This is identified through the interpretation of sensed data. Examples are weather conditions
(physical context) and/or privacy or meeting situations (social context). These situations can be interpreted on historical data as well as on the momentary situation perceived and interpreted. Together they are useful to identify the current situation and enable predicting and reacting to future situations properly.

We describe the mobile environment through an environmental profile addressing the parameters defined in Table 1. Depending on the particular requirements of the application, it is necessary to evaluate whether a particular context parameter needs to be reflected in the design and, thus, in the following evaluation of the system, or not.

3.2. MCF—mobile user

Users of a mobile application are described by a user profile, similar to user segmentation known from product development in marketing (Kotler and Keller, 2008). Such profiles can, for example, be used to describe personas (Grudin and Pruitt, 2002; Cooper, 1995), depicting representative users of a mobile application. This provides an important mean to develop software having specific users in mind. Additionally, a user profile is an important prerequisite for selecting the user groups when conducting user evaluations. Using user profiles and personas to analyze and describe requirements regarding a usage context is an established method in a user centered design process (Cooper et al., 2007; Haikara, 2007; Grudin and Pruitt, 2002; Pruitt and Grudin, 2003).

Users can be analyzed regarding characteristics such as demographic data, traits and intelligence, job or task related factors, their expert status (novice, expert), cultural dimension (language, origin) job-specific roles (e.g. physicians, engineers), disability, gender, age, motion (Zhang et al., 2005; Coursaris and Kim, 2007). To find out which parameters describing a user are relevant to mobile usage scenarios, we structured a user profile based on the application scenario given by the case study in Section 5. Table 2 depicts the parameters that are specific for the mobile user and the mobile usage of an application.

3.3. MCF—mobile task

Mobile tasks can be visualized as use cases (Cockburn, 2001) that describe and include given requirements. We categorize the tasks as shown in Table 3.

3.4. MCF—mobile device

Mobile devices differ from stationary devices in computing power, memory resources, input and output capabilities, and network capabilities (Höpfner et al., 2005). Thus, by device profiles, mobile devices can be categorized and described. The profiles contain data about input and output capabilities, hardware, software, and network capabilities.

Table 1 Mobile context factor—mobile environment.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Location, orientation</td>
<td>Geographical data to evaluate movement and location of the user</td>
</tr>
<tr>
<td>Physical properties</td>
<td>Environmental data like temperature, noise, humidity, light</td>
</tr>
<tr>
<td>Social conditions</td>
<td>Co-location, group dynamics, participant/environment relationship, activity (leisure, work, party, meeting)</td>
</tr>
<tr>
<td>Connectivity</td>
<td>Connection type, expected coverage, stability</td>
</tr>
<tr>
<td>Collaborations</td>
<td>Capability to collaborate with other devices</td>
</tr>
</tbody>
</table>

Table 2 Mobile context factor—mobile user.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interaction capabilities</td>
<td>General for all humans: 5–8 s up to 20 s (Oulasvirta, 2005)</td>
</tr>
<tr>
<td>Attention span</td>
<td></td>
</tr>
<tr>
<td>Motoric capabilities</td>
<td>Abilities and disabilities regarding motoric interaction</td>
</tr>
<tr>
<td>Mental capabilities</td>
<td>Abilities and disabilities that regard the cognitive load</td>
</tr>
<tr>
<td>Mobile usage behaviour</td>
<td>At the office, at home, at university, in public transport</td>
</tr>
<tr>
<td>Preferred location</td>
<td></td>
</tr>
<tr>
<td>User type</td>
<td>Active-dynamic, selective-in-communicative</td>
</tr>
<tr>
<td>Multimedia usage</td>
<td>Use of audio, video, VOIP, triple play</td>
</tr>
<tr>
<td>Application usage</td>
<td>Usage and usage frequency of mobile applications</td>
</tr>
</tbody>
</table>

Table 3 Mobile context factor—mobile task.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Functionality</td>
<td>Functionality to be obtained by executing a specific task</td>
</tr>
<tr>
<td>Work-flow</td>
<td>Combination of tasks that need to be combined to perform a more complex functionality</td>
</tr>
<tr>
<td>Interactions</td>
<td>Possibilities to interact with a mobile device to execute a task</td>
</tr>
<tr>
<td>Duration</td>
<td>Minimum and maximum duration</td>
</tr>
<tr>
<td>Type</td>
<td>Open or closed</td>
</tr>
<tr>
<td>Complexity</td>
<td>Task interactivity and complexity</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Dependencies on external parameters (e.g. task cannot be done while on the move)</td>
</tr>
</tbody>
</table>

Table 4 Mobile context factor—mobile device.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>Heavy devices</td>
</tr>
<tr>
<td>Size</td>
<td>Several typical device sizes</td>
</tr>
<tr>
<td>Screen size</td>
<td>Screen size influences layout possibilities</td>
</tr>
<tr>
<td>I/O interfaces</td>
<td>Interaction capabilities put restrictions on requirements</td>
</tr>
<tr>
<td>Power consumption</td>
<td>Necessity of power saving techniques and provisions for power loss</td>
</tr>
<tr>
<td>CPU performance</td>
<td>Necessity of the reduction of complexity</td>
</tr>
<tr>
<td>Operating system</td>
<td>Operating systems to be supported, scheduling matters</td>
</tr>
<tr>
<td>Robustness</td>
<td>Compact, stable hardware and software necessary</td>
</tr>
<tr>
<td>Heterogeneity</td>
<td>Support many device types (high heterogeneity) or only a few (low heterogeneity)</td>
</tr>
</tbody>
</table>

Examples for such profiles are the open source Wireless Universal Resource File (WURFL)\(^2\) and the Delivery Context Library (DELI).\(^3\) The decision about which classes of devices to support and which restrictions to address are important because they considerably influence the design and implementation efforts, and therefore have to be done early in the requirements engineering phase of a project. Table 4 describes parameters we regard with respect to the device selection requirements which need to be established during the early software engineering process.

\(^2\) http://wurfl.sourceforge.net/.

\(^3\) http://delicon.sourceforge.net/.
Table 5
Mobile context factor—mobile application.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile unit</td>
<td>Software architecture design determines a mobile unit</td>
</tr>
<tr>
<td>Spaces</td>
<td>Environments in which the applications is used</td>
</tr>
<tr>
<td>Context</td>
<td>Definition of context, awareness, consequences of changes</td>
</tr>
<tr>
<td>Collaborations</td>
<td>General collaboration capabilities in the environments</td>
</tr>
<tr>
<td>Location management</td>
<td>Definition, awareness, consequences of changes</td>
</tr>
<tr>
<td>Data storage</td>
<td>Distribution of data storage</td>
</tr>
<tr>
<td>Network</td>
<td>Affordances of network connectivity</td>
</tr>
</tbody>
</table>

The identified parameters regarding mobile devices were identified based on literature review and explorative research concerning the parameters influencing the use of mobile devices (Ballard, 2007; Jones and Marsden, 2006; Schmidt et al., 2005; Chang and Ishii, 2006; Coutrix and Nigay, 2008; Grill et al., 2009).

3.5. MCF—mobile application

The mobile application is the result of the software, interaction and user interface design process, followed by the implementation of the software. The design and implementation of a particular application is based on the requirements defined by the needs and constraints of the contributors of a mobile interactive system. To get information on whether the design of an application satisfies these requirements and whether it is an optimal solution to a given design problem, we need to evaluate the mobile application against the previously identified requirements.

Unpredictable context and changes of contexts are indicators of usage errors and need to be addressed in the software design of a mobile application as well as in the design of the interface the user is confronted with (Thurnher et al., 2006).

The business criteria based classification scheme of mobile computing applications by Varshney and Vetter (2002) show basic requirements for mobile applications, i.e. location management, multicast support, network dependability, quality of service, roaming across networks. Roman et al. (2000) describe space, the unit of mobility, location, and context as key concepts for mobile applications design. Important parameters of the mobile context factor application, which are identified based on the literature research conducted, are presented in Table 5.

4. Combining the methods

The method proposed (see Fig. 1) in this paper is based on the combination of a software architecture analysis and a usability evaluation in the form of a user test.

Usability problems occur because of missing or wrong definitions or violations of usability requirements which depend on the usage context.

To be able to combine both methods the evaluation parameters of both methods are based on the same requirements, especially identified through the MCFs defined in Section 3.

While the usability evaluation observes the user as the source of a problem the software architecture analysis evaluates the software architecture of a system.

The outcome of both methods depict usability problems identified through either the SATURN method and/or usability evaluation.

This provides a common basis for comparing the results of both methods.

The combination of both methods allows us to address usability of software in a more holistic way through approaching usability issues from the user’s side as well as from the software architecture’s side. Identified usability problems can be used to update, to enhance and/or to determine new requirements for a software product and to prioritize them. The intention behind this is to reduce software development iteration cycles during a software development process and to enhance the quality of a software product.

In the following sections the method developed in this work is presented.

4.1. Software architecture analysis

Our scenario-based method for “Software Architecture analysis of Usability Requirements realization” (SATURN) helps analysts and software architects to analyze a software architecture draft to find indicators for potential usability problems. Also the method allows to determine which architectural support level (ASL) is achieved by the current architectural design. SATURN can be applied to final versions of the SA. It can also be used as an
architecture analysis and design improvement method in an architecture-based iterative software development process. The method includes a procedure model, questionnaires, and a knowledge base that consists of generic scenarios and a pattern language. The knowledge base is derived from research regarding the relationship between usability and software architecture (see Section 2). The research goals of the method are the efficient support of architectural design, the support of non-usability-experts to obtain traceable and correct results regarding the elicitation of architecture and usability problems.

The process steps conducted during the application of the method SATURN are: (1) describe the analysis context, (2) determine analysis scenarios, (3) evaluate scenarios, (4) interpret results, and finally (5) review method and tools, as seen in Fig. 2.

4.1.1. Phase 1—describe analysis context

This phase focuses on the initial analysis of all available data and documents described in the analysis context document (ACD), including the current user interface prototype and the determination of the usage context as described in Section 3. This phase consists of four steps: after the general business context of the product is presented, the mobile usage context factors are specified. A simple user interface walkthrough is accomplished to show which interactions are important for the application and which usability problems are already visible. Interactions are listed and can later be used to browse the scenario catalog of the knowledge base. Finally, a coarse overview of the architecture is given by describing architectural views in an architectural notation that project partners have agreed upon (a pragmatic approach is to use the de facto standard UML).

The coarse overview should include a component or distribution diagram, a list of neighbor systems, their software interfaces, and an interface description depicting the classes, components, the used communication protocol and medium.

The ACD then holds this architecture description, refers to other product related documentation, the main goal of the application, the MCF, and a list of interactions.

4.1.2. Phase 2—determine analysis scenarios

The usefulness of the analysis results depend on a well-grounded selection of analysis scenarios. By using the ACD and a catalog of generic scenarios, architects and analysts discuss which scenarios are suitable for the application. These scenarios describe usability requirements and can be mapped to the requirements and the interactions that were described in the first phase. Depicted based on ATAM (Clements et al., 2002), they were derived from patterns following an approach based on Zhu and Ji (2004). Scenarios represent brief descriptions of a particular interaction with the system and are initiated by users or the system. They capture uses of the system which are related to concrete requirements that users have when interacting with an application. Scenarios describe stimuli and in what particular way the system should respond to them. Such a scenario is formulated in the following: a source (a certain user group as shown above or the system) has a stimulus (wants to stay informed about the system’s state, activities, changes) towards an artifact (application that allows interaction with an exhibit via bluetooth) in a certain environment (at runtime in a museum). The artifact deals with this usability requirement by providing a certain response (picks the data required and presents them, e.g. the current interaction status), which can be evaluated by a certain response measure (correct reaction and update of status indicator within an appropriate period of time). The knowledge base provided with this method includes generic scenarios and evaluation documents that include references to considered design patterns.

4.1.3. Phase 3—evaluate scenarios

For each analysis scenario, we determine whether the decisions concerning the particular scenario help to realize the response and how this realization can be graded. Use cases and the components and interfaces that realize them are discussed, and design patterns are used to compare a solution found. Patterns hold additional information about architectural elements and their responsibilities that should be regarded for the realization of a scenario, as well as advantages, disadvantages, and related problems of a solution. During the discussion between architect and analyst, decisions are elicited. Some can be classified as sensitivity points, i.e. they affect the realization of the scenario’s response and thus affect usability. The grade of the severity of the usability problem is graded 0–3: (0) if no usability impact could be identified and no problems are expected, (1) if a sensitivity point might lead to problems for several user types, so users will not like using the application but continue to do so; (2) if a usability impact was identified that will lead to problems for sure, and users will not accept the application; (3) if the usability impact is unpredictable. The impact of a scenario depends on its architecture sensitivity. A scenario is architecture sensitive if one or more components or interfaces have to be changed or added. In case there is no usability problem, the architecture does not have to be checked for necessary changes. But if problems might occur, we discuss which impact an architectural change would have. The impact on the SA is also graded 0–3: (0) means no changes are expected, (1) means simple modifications are expected, (2) means that complex modifications are expected, and (3) means that modifications are unpredictable or unknown. In the tradition of ATAM (Clements et al., 2002) we classify a sensitivity point if it affects other quality attributes such as performance or security as a trade-off point, if architectural changes necessary are known, we classify it as a non-risk, and if unpredictable architectural changes are expected, we classify it as a risk.

4.1.4. Phase 4—interpret results

In this phase the findings of the scenario evaluations are summarized and an architectural support level (ASL) is determined. The ASL defines the impact of each evaluated scenario on the software architecture. The findings of the scenario evaluations – identified through the sensitivity points – are discussed and categorized according to their impact on usability and software architecture. Open questions to be addressed by software architects are formulated with the goal to improve the architectural design of a software product. Optionally a future ASL can be described. It defines which ASL the scenarios should achieve in the next analysis.
4.1.5. Phase 5—review method and tools

In order to continuously improve the method described, the ACD, our generic scenarios, and the analysis results are evaluated. Experiences and evaluator feedback regarding the method are collected. The findings obtained are incorporated into scenarios and design patterns contained in the knowledge base. The intention behind this is to optimize our method and to obtain an enhanced and improved knowledge base being in line with technological advances and containing more profound data.

4.2. Usability evaluation

There are different kinds of usability evaluations that elicit different problems: During heuristic evaluations, the interface is evaluated against guidelines, style-guides, design rules, and the experience of the evaluator. By conducting user tests for mobile applications, we may identify problems that relate to the tasks, work-flows, external disturbances, and effects that may influence the usage of an application. Usability evaluation focuses on the usage and usage related problems. SA analysis identifies problems in the architectural design of the software architecture, thus, the evaluations might complement each other.

Several steps need to be accomplished in advance to enable the execution of the evaluation: (1) define the goal of the evaluation, (2) define the evaluation object, (3) define the evaluation scenario and setting, (4) define and arrange the user group to evaluate with, (5) prepare the evaluation, (6) conduct the evaluation, and (7) analyze the results. These basic steps apply to all user evaluations. Different steps, for example, the definition of the evaluation scenario and setting, depend on the goal of the evaluation and the mobile application to evaluate.

In order to evaluate the overall usability of a mobile application, a functional prototype or the final product is required. The user tests are led and conducted by a usability expert who has knowledge about the evaluation and analysis methods to be applied in order to achieve valid and significant results. The results cover usability problems that occurred during the usage of the mobile application. The evaluation of the results is done according to the mobile context factors (see Section 3) defined by the evaluator in advance.

4.3. Comparison of the evaluations

After the analysis and the evaluation, results are analyzed. In order to obtain a common basis for comparing the results, we proceed according to the following steps for both types of results: (1) summarize problems, (2) identify the source of the problems by mapping the results of the usability evaluation and the software architecture analysis, (3) identify possible solutions and explanations for the problems, (4) mark unsolved problems, if necessary, and (5) then evaluators discuss the open items of the list. Open items are problems of which the source or the effects could not be identified clearly. For each problem the results of the discussion may identify difficulties with one or more of the following parameters.

- Agreed. Evaluators agree on a common problem source and identify a solution.
- Further evaluation required. Issues were identified but further evaluation is required to clarify the problem and find an appropriate solution. Regarding the SA analysis, new analysis scenarios are selected or created for a follow-up analysis.
- New problem identified. Either the usability evaluators or software architects identified a problem that could not be identified by the other. This results in further evaluations.
- Problem identified but no solution found. A solution cannot be reached.

Depending on the results, either another iteration of UE and SA or a new user interface design or software architectural design phase is required. In both cases further discussions to solve open issues are initiated.

5. Case study—Shake Your Mac evaluation

The case study “Shake Your Mac” evaluates a system used for interacting within a mobile environment. In order to evaluate and improve the system in an iterative process we apply our approach by combining a usability evaluation with a software architecture analysis. Finally we discuss the results and show how it is possible to improve a software engineering process by considering findings that would not have been found by conducting one evaluation method only.

The SYM system was elicited for evaluation because it is a mobile application that runs on a mobile device and is used in a mobile environment. The mobile context factors themselves represent important requirements for setting up the SATURN method as well as the UE to be conducted.

The primary goal of the case study is to show the feasibility and validity of the approach described in Section 4.

5.1. Shake Your Mac

“Shake Your Mac” (SYM) is a system that uses a device’s acceleration sensors. The sensors are located on a mobile device (in our case an Apple MacBook) and interpreted directly on the mobile interaction device. Particular movements and interactions from a user are translated into events. A software can connect to the interaction device and interpret the interaction events obtained from the device. SYM is a system that represents the software part of an interaction device. It can be used for realizing interaction based on the movement of a device, e.g., shaking the device, as well as for haptic interaction with a device, e.g., knocking on a device.

The mobile context factors underlying the design of the SYM system have been defined during the requirements engineering phase of the development process.

During phase 1 of the SA analysis, we created, amongst other artifacts, the UML representation of the application SYM running on the interaction device (see Fig. 3). It depicts the co-location and relation of the components of SYM as well as the distribution of the different software architecture components over different devices.

The current implementation of the system includes two clients which control applications and use the interaction events obtained from the interaction device to execute functionality within the controlled applications.

- GoogleEarth Output Client. The GoogleEarth (GE) Output Client implements the connection to the Google Earth application which is realized through the COM interface provided by Google Earth. The current implementation contains the task of navigation (into four directions), tilting (tilt in/tilt out), zooming (zoom in/zoom out). All these tasks navigate a globe controlled and visualized by Google Earth. The different tasks can be accessed through the usage of different modes.
- PowerPoint Output Client. The PowerPoint (PPT) Output Client implements the connection to the Microsoft PowerPoint application, which is realized through the COM interface provided by PowerPoint. The current implementation contains the task of navigating a presentation (forward/backwards/to the beginning/to the end). The events obtained from the interaction device are used to perform these tasks. The current design interprets moving right or knocking on the interaction device as navigate forward. Moving the interaction device to the left is interpreted as navigate back-
wards. Analogous moving forward means navigating to the end and moving back navigating to the beginning.

Multiple applications can be started and connected to the interaction device represented by the Sudden Motion Sensor Controller (SMSController class) and obtain data via a wireless data connection from the interaction device.

We followed the approach presented in Fig. 1. Firstly, a software architecture analysis was done. The results of the analysis depict usability problems based on analysis scenarios that are related to requirements. Secondly, a usability evaluation found usability problems based on the usage of SYM. After both studies were conducted, we compared the results.

5.2. Software architecture analysis

A software architecture analysis regarding the usability of SYM was conducted by the system’s architect who is also its programmer with the goal to describe the SA of SYM. An analyst who has SA experience conducted the SA analysis in order to identify usability problems derived from errors in the software architecture of a system. Both architect and analyst are experts in interaction design and programming.

The SA analysis SATURN consists of the preparation phases 1 and 2, followed by a detailed analysis in phase 3 that uses the products of the earlier steps. Results are presented in phase 4, and phase 5 reviews the SA analysis method and its knowledge base (see Section 4).

5.2.1. Phase 1—describe the analysis context

The goal of the SA-analysis of SYM was to find usability problems that are based on software architecture decisions. We used the mobile context factors defined during the requirements phase of the software development process, for example, the system should be used by two types of users (e.g. mainly differentiated by their age, eyesight abilities, IT-experience), in two types of environments (e.g. a museum and a classroom). The mobile context factors (see Section Table 6 Sensitivity points assessment.

<table>
<thead>
<tr>
<th>S</th>
<th>Description</th>
<th>Usability impact</th>
<th>SA impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Interaction handling of the application controller</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S2</td>
<td>Existence of a special distribution component</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S3</td>
<td>Interaction events transformed at application controllers</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S4</td>
<td>Non-existence of a provision to add new input clients</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>S5</td>
<td>Non-existence of a provision to add new output clients</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>S6</td>
<td>TCP/IP used as network protocol</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>S7</td>
<td>GUI at SMSController indicates invoked interaction</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>S8</td>
<td>GUI at application controllers do not indicate currently processed interaction</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>S9</td>
<td>Non-existence of a provision to monitor/handle connectivity</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**Usability impact**

- 0—no usability impact identified
- 1—usability impact identified; users do not like using the application
- 2—severe usability impact identified; users will not accept the application
- 3—unpredictable usability impact

**SA impact**

- 0—no SA changes expected
- 1—simple changes expected
- 2—complex modifications expected
- 3—unpredictable or unknown modifications expected
3) are used to determine requirements that help to select and/or define analysis scenarios in step 2. In that way, they allow to decide which scenarios are important for the product. We conducted a walkthrough and noted that interactions such as cancel, feedback and adding new devices should be addressed during analysis. Architecture documentation in the form of class diagrams, source code of the prototype application, an UML component diagram (see Fig. 3), and CRC cards were created to identify the components, their responsibilities, collaborations, and interactions.

5.2.2. Phase 2—determine analysis scenarios

By using the ACD and the catalog of the generic scenarios, analyst and architect selected the scenarios cancel, add new input device, feedback/progress-indicator, safe the user’s work, multi-level help, operating consistently across views, and non-conflicting device usage in a first round. Due to a short time frame of 1 day only a small group of four analysis scenarios was selected for this first iteration, although the others could have been appropriate as well. After a discussion the scenarios cancel, add new input device,
feedback/progress indicator, and safe the user’s work were agreed upon. No new scenarios were added.

5.2.3. Phase 3—evaluate scenarios
For each of the analysis scenarios the corresponding use cases were discussed. The main focus was on identifying the software architecture, i.e. which components and classes are responsible for implementing the use cases. Each architectural decision was addressed, necessary UML models were drafted. After the discussion, architectural decisions were recapitulated, and sensitivity points (decisions that possibly effect the implementation of the analyzed scenarios) were identified. Table 6 shows the identified nine sensitivity points and their assessment.

5.2.4. Phase 4—interpret results
We identified nine sensitivity points. Six of them lead to usability problems, two of them demand simple architectural changes, two demand complex modifications, and the impact of two scenarios was unpredictable (see Table 6).

The sensitivity points S1, S7 and S8 are concerned with the feedback and the cancel functionality of the system. We found out that the interaction handling of the GoogleEarth application controller GEOOutputClient does not allow direct manipulation (S1). It shows the invoked interaction at the interaction device (S7), but it does not indicate the currently processed interaction of the application (S8). The application needs to finish an action after another event, for example, stopping the globe, may occur. This can lead to a delay of response time. A user will not see a reaction to his or her interaction and might try to invoke it again. Severe usability problems are expected and users might even stop using the software. Regarding the SA, there are only few changes necessary: only the responsibilities of the software interface class of the application controller and the inheriting specific application controllers have to be altered.

The sensitivity points S4, S5 and S6 are concerned with adding new devices to the system. There is no provision to add new input (S4) or output (S5) clients, and the use of TCP/IP as network protocol for the application (S6) can lead to response problems, because, during the request-response cycle, events are pulled depending on a set frequency, not as events happen. The more input and output devices are used, the higher the communication load. An alternative would be to use both the TCP/ IP and the UDP protocol. TCP/IP could be used for the necessary bidirectional communication for, e.g. setting up the connection, while the UDP protocol could be used for communicating the events as they happen. The architects might also consider whether many specific output clients running on different computers should be integrated. Then, it would be better to incorporate the component of the output client and its distribution function on the server and to broadcast interaction events.

The sensitivity points S2, S3, S6 and S9 are concerned with communication. The PPTOutputClient and the GEOOutputClient transform the movement data received into application’s specific interaction events (S2). This solves the problem that each output client requires to enable application specific functionality. This task is accomplished by a distribution component within the general application controller (S3) which encapsulates this functionality. But the non-existence of a provision on the sudden motion sensor controller which monitors whether the application is connected (S9), leads to severe usability problems. If the client running on the interaction device or the general application controller loses the connection, it needs to be re-connected manually due to the use of the TCP/IP protocol (S6). Users might not be skilled enough or do not have access to the controller application. In the second environment targeted (the museum), a wireless connection is used for the necessary bidirectional communication for, e.g. setting up the connection, while the UDP protocol could be used for the required interaction events. The GEOOutputClient does not allow direct manipulation (S1). It shows the invoked interaction at the interaction device (S7), but it does not indicate the currently processed interaction of the application (S8). The application needs to finish an action after another event, for example, stopping the globe, may occur. This can lead to a delay of response time. A user will not see a reaction to his or her interaction and might try to invoke it again. Severe usability problems are expected and users might even stop using the software.

5.2.5. Phase 5—review the method and tools
During the case studies, new design patterns were identified and integrated in our pattern library being part of the knowledge base. This allows us to enlarge the knowledge base available for conducting the SA analysis method. As a result the knowledge base was enhanced by altering the contained scenarios to hold more information.

5.3. Usability evaluation
A usability evaluation for SYM was setup as an in vitro evaluation. The reason for that was that the usability evaluation should test the mobile factors that occurred through the mobile usage of SYM within a classroom.

The mobile context factors (see Section 3) play a major role during the setup of the usability evaluation. The major categories of the mobile context factors identified, i.e. environment, user, task, device, and application, are addressed during the setup phase of the usability evaluation. They provided the basis for identifying the parameters to be analyzed during the usability evaluation.

An example for this is that based on the mobile context factors we described usage scenarios that the test users had to conduct. The user using the navigation functionality of Google Earth was located in a museum which was reflected through mobile context factor environment and which played a major role in the type of interaction to be evaluated during the usability evaluation.

For the evaluation, a user group of 12 test users was chosen in order to get a result with a statistical relevancy sufficient for a formative usability evaluation (Bevan et al., 2003). The test users were selected so that 58% were female and 42% were male with an average age of 31. The sessions were recorded on video and analyzed later using an evaluation framework targeting mobile applications (Grill, 2009).

The MacEval tool (see Fig. 5) which is part of the used evaluation framework allows us to record, replay and annotate the evaluation sessions. The analysis has been done through reviewing the sessions and evaluating the actions done by the user according to the parameters defined in Table 7. After reviewing the sessions the data has been exported and transferred to MS Excel for further statistical evaluation.

The parameters have been developed with respect to the use cases underlying the implementation of SYM and providing the basis for the evaluation scenarios. The use cases have been analyzed regarding the mobile context factors identified in Section 3. This reflects the consideration of the mobile context factors in the parameters analyzed during the usability evaluation.
During the evaluation, the users were asked to perform two scenarios of which each lasted approximately 5 min. The first scenario was to perform a presentation using SYM as an input device. During the presentation the user was asked to perform three tasks. All three tasks are based on the functionality of navigating slides during a presentation using SYM. The second scenario contained the task to navigate the globe of the GoogleEarth application by the SYM interaction device. The users had to perform two tasks which included all three different interaction types of GoogleEarth (navigating, tilting and zooming).

After the usability evaluation had been conducted, the results were analyzed according to the mobile context factors identified for the SYM interaction device. The users did not understand that knocking also initiates a forward navigation of the slide. In the GoogleEarth scenario the usage of the different interaction techniques is distributed with a mean of 34.25% for moving left/right, 40.42% for moving front/back. 13% of the interactions were knocking in order to perform an action, and the rest were unobserved interaction techniques. Anyway for the GoogleEarth scenario the applied interaction techniques vary more than for the PowerPoint scenario, which relates to the structure of the given tasks.

Table 8 shows that the ANOVA analysis comparing the different, independent, values identifies a high relation between all the combinations of the observed parameters for the PowerPoint scenario. For the GoogleEarth scenario a tight relation between the navigation problem and the application response could be observed. As this occurred in both scenarios, we can conclude that these two observed parameters correlate and generally indicate a navigation problem due to the application’s response. Due to the distribution of the usage of the different interaction techniques, it can be concluded that, when focusing on one certain type of mobile interaction, the understanding is much higher together with the observed learning curve where the mean average for the PowerPoint scenario lies at 3.25 while for the GoogleEarth scenario at 2.5 on a 5-point likert-scale.

The relation between navigation problems and the application response is explained by the mobile context factors identified for the mobile usage of SYM. This relation is for example confirmed through the short attention span available during the mobile usage of SYM. This reflects the requirement for a reasonable application response time to allow the user to correctly execute his or her task in the mobile application scenario.

5.4. Comparison of results

When comparing results of the SA analysis with the results of the usability evaluation, we confront the four different possibilities elaborated in Table 9. The first three situations occurred during the execution of the SA method and the usability evaluation. The first situation (type 1) has the advantage that problems were found during the SA analysis which could be analyzed afterwards by an additional usability evaluation. The problem could be proven through results already observed and stated during the usability evaluation. The second situation (type 2) finds problems with respect to the usage and the tested scenarios which could not be covered by the software architecture analysis. The cause of such problems usually lies in human error or cognition problems. The third situation (type 3) where both methods find the same problem provides us with additional information about the severity of a problem with different aspects of a particular problem. While the SA analysis finds problems that are based on architectural issues and might be related to the usage of the application, the usability (both types of interaction were applied in approximately 20% of the total interactions executed during the evaluation). Most of the users relied on moving the notebook left/right to navigate. We can conclude that the users did not understand that knocking also initiates a forward navigation of the slide. In the GoogleEarth scenario the usage of the different interaction techniques is distributed with a mean of 34.25% for moving left/right, 40.42% for moving front/back. 13% of the interactions were knocking in order to perform an action, and the rest were unobserved interaction techniques. Anyway for the GoogleEarth scenario the applied interaction techniques vary more than for the PowerPoint scenario, which relates to the structure of the given tasks.

Table 8 shows that the ANOVA analysis comparing the different, independent, values identifies a high relation between all the combinations of the observed parameters for the PowerPoint scenario. For the GoogleEarth scenario a tight relation between the navigation problem and the application response could be observed. As this occurred in both scenarios, we can conclude that these two observed parameters correlate and generally indicate a navigation problem due to the application’s response. Due to the distribution of the usage of the different interaction techniques, it can be concluded that, when focusing on one certain type of mobile interaction, the understanding is much higher together with the observed learning curve where the mean average for the PowerPoint scenario lies at 3.25 while for the GoogleEarth scenario at 2.5 on a 5-point likert-scale.

The relation between navigation problems and the application response is explained by the mobile context factors identified for the mobile usage of SYM. This relation is for example confirmed through the short attention span available during the mobile usage of SYM. This reflects the requirement for a reasonable application response time to allow the user to correctly execute his or her task in the mobile application scenario.

5.4. Comparison of results

When comparing results of the SA analysis with the results of the usability evaluation, we confront the four different possibilities elaborated in Table 9. The first three situations occurred during the execution of the SA method and the usability evaluation. The first situation (type 1) has the advantage that problems were found during the SA analysis which could be analyzed afterwards by an additional usability evaluation. The problem could be proven through results already observed and stated during the usability evaluation. The second situation (type 2) finds problems with respect to the usage and the tested scenarios which could not be covered by the software architecture analysis. The cause of such problems usually lies in human error or cognition problems. The third situation (type 3) where both methods find the same problem provides us with additional information about the severity of a problem with different aspects of a particular problem. While the SA analysis finds problems that are based on architectural issues and might be related to the usage of the application, the usability
Table 9
Comparison of the results of the SA analysis and UE.

<table>
<thead>
<tr>
<th>Type</th>
<th>Problem found by</th>
<th>Argumentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>SA analysis found problems not covered by the UE. Example: It was found out that the application analyzed did not cover a system requirement. In this case the possibility to use SYM with multiple applications did not work at the time of the analysis.</td>
</tr>
<tr>
<td>2</td>
<td>–</td>
<td>Usability evaluation found a problem that was not covered by the SA analysis. Example: The response time of the system was not appropriate for the users.</td>
</tr>
<tr>
<td>3</td>
<td>X</td>
<td>SA analysis and UE analysis found a problem. Example: The SA analysis recognized that a status indication shows the state of the sensor instead of the state of the interaction device. The UE found out that the user was irritated by the given state.</td>
</tr>
<tr>
<td>4</td>
<td>–</td>
<td>No problem was found by either the SA analysis or the UE.</td>
</tr>
</tbody>
</table>

The identified problems and open questions based on these results need to be addressed during the next cycle of an iterative software development process. Our approach allows us to address more usability problems within one iteration cycle. This reduces the amount of iterations necessary.
6. Conclusion and future work

In this paper we present a method that aligns a software architecture (SA) analysis and a usability evaluation (UE), and thus addresses usability holistically. We exemplified how these methods can be combined and how their results relate to each other. In particular, we proposed to use requirements as a common basis for both the SA analysis (an inspection method) and a usability evaluation UE (a user test), as well as to map and discuss results described as usability problems. Also we categorized and defined factors that describe the usage context of a mobile application, and thus requirements. We evaluated our method which uses previously defined mobile context factors in a case study.

In order to show the validity of the results of our method we applied the software architecture analysis SATURN to a mobile application and were able to identify usability and architectural problems. The follow-up usability evaluation identified usage problems based on application errors and on human errors. While the application errors were related to problems in the quality of the software, the human errors could be observed in workflows that do not fit the intuitiveness of the human behavior as well as through a variety of parameters like environmental context, influencing the user during the usage of an application.

When comparing the results of both methods we obtained a bigger amount of problems than by applying only one single method. Also most open questions could be explained by the particular results of the other method, others lead to new problems and new requirements. We were able to confirm usability problems found by the SA analysis and define their severity through the usability evaluation. We conclude that the increased quantity and quality of results can lead to a reduction of the number of iterations required in the early stages of an iterative software development process.

Our proposed method was tested and is valid under the restrictions given through our application Shake Your Mac (SYM), a simple application with low complexity, with an easy understandable software architecture. This example allowed us to show how the SA analysis and the usability evaluation can be aligned. In further projects we already apply the proposed method to bigger and more complex mobile applications where the first findings already show that both SA analysis and UE become more time-consuming and difficult: The SA analysis phase 3 has to consider more decisions and more interdependencies between them. Also the UE gets more complex through the increased number of functionalities available and thus through more complex tasks. With increasing complexity of applications, the selection of tasks for the UE and SA analysis scenarios become more vital for successfully applying our method. The combination of the SA analysis and the UE allows a better selection of scenarios because we can prioritize them depending on which problems are severe from a usability and/or software architecture perspective.

Future work comprises the application of the proposed method in various software projects with the goal of reducing the costs of a software project and to improve the quality of the software.

Acknowledgements

We would like to thank the reviewers for their comments. They were a valuable input which helped us to improve our paper.

References

URL http://www.tomgrill.info/show.php?name=PhD.
Jones, M., Marsden, C., 2006. Mobile Interaction Design. John Wiley & Sons (Febru-
ary).

Bettina Biel is research assistant and PhD student at the Applied Telematics/e-Business group at the University of Leipzig. Her research interests are mobile interface and interaction design, software architecture, design patterns, and trust in ubiquitous computing. In her thesis, she designs a method for the analysis of architecture-level support of mobile usability.

Thomas Grill is a researcher in the in the HCI & Usability Unit of the ICT&S Center at the University of Salzburg. He received his doctoral degree in business informatics from the Johannes Kepler University Linz in the area of interaction design with a focus on alternative interaction styles and techniques. His research includes tools and methods in the area of human computer interaction with a special focus on ubiquitous computing appliances. He is also working on alternative interaction methods with the goal of improving interaction and performance by using external devices and interaction methods like motion tracking and eye-tracking. Current research also tries to enhance usability evaluation methods through the measurement and interpretation of context information in order to gain more advanced information about the user and the environment he resides in.

Prof. Dr. Volker Gruhn is leading the Software Engineering Group at University of Essen and is a member of the board of Paluno – The Research Institute for Software Technology at the University of Duisburg-Essen. His research interests are software processes, processes in general, software architecture, usability as well as component-based and generative software development of web and mobile systems. He was the general and program chair of the Software Engineering SE 2006, and program co-chair for the International Conference on Software Engineering ICSE 2008. He is author and co-author of more than 120 national and international publications.