Structured and Unobtrusive Observation of Anonymous Users and their Context for Requirements Elicitation

Olesia Brill and Eric Knauss
Software Engineering Group
Leibniz Universität Hannover
Hannover, Germany
{olesia.brill, eric.knauss}@inf.uni-hannover.de

Abstract—Today, people find themselves surrounded by IT systems in their everyday life. Often they are not even aware that they are interacting with an IT system. More and more of these systems are context adaptive. Requirements to such systems may change for various reasons: The context may fundamentally change when other systems are introduced. New trends and fashions may evolve. Operators need to react quickly to such changes if they want to keep their systems competitive. Traditional approaches to requirements elicitation start to fail in this situation: context adaptive systems serve many users with different profiles. In addition, users may be reluctant to participate in improving it. Thus, it is hard to establish a representative model of requirements. Furthermore, it is hard to capture the context of requirements by subsequent interviews. In this paper we present a systematical approach for requirements elicitation based on observing anonymous users. The interaction of users with the system is observed in the normal working context. Observation is based on assumptions on how interaction should take place. Deviations from these assumptions point to new requirements. Observing a large number of users leads to a quantitative map of requirements in context. Preliminary evaluation shows that the approach is promising. It allows efficient observation of many stakeholders and the derivation of new requirements.

Keywords—requirements evolution; user observation; requirements elicitation; context adaptive systems

I. INTRODUCTION

Today, more and more context adaptive systems are encountered in everyday life. Because of their strong integration in daily tasks, these systems are often not even perceived as IT systems by its users. A good example of such environments are IT Ecosystems that consist of various, constantly interacting and partly autonomous subsystems [1], e.g. in a SmartAirport. Requirements for such pervasive subsystems may change quickly. Other subsystems may change their environment or users become accustomed to the new support in everyday life and start to demand new features. Operators of such subsystems need to discover changed stakeholder needs and new trends fast, if they want to remain competitive. Because the number and complexity of such subsystems constantly grows, users might not know of or may not be interested in the capabilities of a specific subsystem at all [2]. Thus, they might not have enough motivation to actively participate in requirements elicitation efforts. This hinders the further development of the subsystem and leads to our Problem Statement: Operators need support to elate requirements of heterogeneous, anonymous, and generally non-accessible users. Such support should allow to unobtrusively observe the users’ activities in their natural context in an effective and efficient way.

Many established techniques for requirements elicitation (cf. [3], [4]) are not directly suitable for context adaptive systems, because they rely on stakeholder participation. Especially, modeling the system’s adaptation to its context is still a difficult task [5] with only insufficient support in traditional approaches to requirements engineering [6]. For these reasons new approaches are required [2], [5], [6].

Main Contribution: In this paper we present concepts for identifying new requirements or indicators pointing to new requirements. We link user goals with user behavior and relevant context information to define the expected User Behavior. Deviations in the observable user behavior in a specific context can point to new requirements. Once identified, these new requirements can be modeled and refined by other methods (e.g. [7]). Our approach contributes a systematic and pro-active way of supporting continuous improvement and evolution of a context adaptive system. Many information systems (e.g. web applications or phone apps) already provide feedback facilities (such as emails to webmaster or comments in the marketplace). Beyond such applications, we focus on situations, where users may not be aware of using a context adaptive system and consequently are not able to give feedback. For best results, our non-interfering observation can be combined with additional established feedback mechanisms (e.g. interviews, questionnaires, think aloud tests) with randomly selected users.

In two evaluation scenarios we show that analysts are able to i) prepare the observations based on our concepts, ii) capture the required data in the field, and iii) are able to derive requirements from the contextual observations. Based on the results, we also discuss the scalability of our approach and its potential for continuous improvement.

We give an overview of related work in Section II and motivate the need for a new observation based approach.
We present our concepts and a supportive tool prototype in Section III. In Section IV we present the evaluation of our approach, based on applying the prototype in two evaluation scenarios. In Section V we conclude this paper with a discussion of future research directions.

II. RELATED WORK

In goal based approaches to requirements engineering, goals are analyzed and refined, and then mapped to requirements in order to realize them in the system under construction [8], [9]. Kolo-Mazuryk et al. discuss the applicability of established goal based approaches for adaptive systems in a literature survey [6]. Accordingly, goal based approaches are important for requirements engineering for context adaptive systems but currently have weaknesses in capturing contextual requirements. Thus, our approach starts from a given set of user goals and helps to identify indicators for changed or new requirements in the working context. These requirements should also be used to update existing goal models, thus reflecting the changing requirements of the system under investigation.

In this Section we show how our concepts relate to other work in the areas of observation for requirements engineering, defining context in requirements engineering, and requirements engineering for context adaptive systems.

A. Observation Approaches in Requirements Engineering

According to SWEBOK, observing users during requirements engineering is an established approach [10]. Existing approaches [3], [4] lack dedicated concepts to capture, analyze, and document contextual requirements. Furthermore, these approaches usually rely on stakeholder interaction [4], [11], [12]. Typically, users are observed while doing a defined task. The analyst might interrupt the users, take notes for later interviews, or ask users to think aloud [13]. An even more interactive technique is apprenticing: the analyst is trained by the domain expert in doing a specific task [4].

Sørby and Nytrø describe a framework for structured observations in hospital wards [14]. In this domain, important stakeholders must not be interrupted and may not be available for long interviews. Furthermore, a domain expert is needed for interpreting the observations. Thus, students of medical science were hired for observation. This work inspired many ideas in our approach. We address a more general domain of context adaptive systems and focus on the evolution of their rather local context specific functionality.

Several frameworks for observing users of mobile applications have been proposed in literature [15], [16]. Both, Fröhlich et al. and Poppinga et al. report that users are very willing to give feedback for mobile systems.

Kim et al. propose a comprehensive instrumentation solution which they applied to observe users of video games [17]. Their solution has many elements that do not require observers to interact with the users. In virtual environments of video games, most observation data can be obtained automatically. In real world environments, we are currently limited to human observers but plan to use sensors in the near future.

According to Robson, the approach in this paper can be classified as an unobtrusive and structured observation approach [18]. Such approaches have the advantages of a relatively high reliability and validity, bought by a loss of complexity and completeness of the observations, caused by the narrow focus of the pre-structure. Thus, less structured and more interactive methods and techniques can (and should) be applied to complement our observations.

Action Theory (AT) and Distributed Cognition (DCon) are two theoretical frameworks that proved valuable in guiding analysis and observation in the fields of human-computer interaction (HCI) and computer supported cooperative work (CSCW) [19], [20]. Nardi argues that an activity can only be fully understood, when it is observed in its everyday context with focus on its integration into social practice [19, pg. 14]. AT focuses on the activity of a subject as unit of analysis. Halverson offers a comparison of AT and DCog in the context of CSCW [20]. Accordingly, AT emphasizes on the differences between humans (subjects) and other actors (tools). The subjects activity is the unit of analysis. In contrast, DCog treads humans and computer-tools uniformly and allows to define custom units of analysis. This allows to describe working processes independent from the work distribution over human and computer actors. Our approach incorporates the notion from AT that subjects and their context should be distinguished during analysis. Both, AT and DCog help to find suitable context and behavioral attributes for our structured and unobtrusive observation. Additionally, AT can offer psychologically founded support to understand, why a person acts in one way or another.

B. Context in Requirements Engineering

As software intensive systems pervade more and more aspects of life, the notion of context becomes more important in requirements engineering. Some of the requirements for such systems are only valid at specific positions [21], [22]. Furthermore, specific behavior of a system might be undesired depending on the context [23].

The concept of context is not limited to the spatial position. Schmidt et al. propose a hierarchical model of context attributes [24]. This work is an excellent starting point, when searching for relevant context attributes in a given situation. Still, a complete taxonomy of all existing context attributes is missing. For this reason the definition of context by Abowd et al. remains the baseline, when considering context in requirements engineering [25]:

“Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” – Abowd et al. [25]
Usage context and user expectations have to be considered for requirements elicitation as well, for which Sitou and Spanfelner propose a set of possible models [26]. Beyond location, context information includes participants, activities, and environment [26]. Salifu et al. propose a comprehensive approach to model context and requirements [7]. As opposed to these contributions, we do not focus on modeling requirements, but on their elicitation.

Recently, different approaches have been proposed to capture context during requirements elicitation. Either validation of scenarios in the original context of use is proposed [27], [28], or users are given the opportunity to articulate their own requirements via mobile devices [29], [30]. Our approach can complement these approaches with requirements from users that are reluctant to participate in such efforts.

C. Requirements Engineering for Context Adaptive Systems

Cheng et al. describe adaptive systems as systems that are able to adapt their behavior based on their perception of the environment and the system itself [5]. Welsh and Sawyer argue that adaptive systems are well suited in situations characterized by many non-functional requirements, if the relative priorities of these requirements vary depending on the context [31]. In this case, our observation framework may help to capture the contextual priority of goals and requirements. Thus, important information for the roll out and optimization of such systems is given.

According to Cheng et al. the central task of requirements engineering for adaptive systems is to capture the relevant context attributes for the adaptivity of the system [5]. Based on these context attributes, analysts have to assess, which requirements might change during runtime of the system. Cheng et al. argue that not all environmental influences can be defined during design time of a system. Consequently, they warn that the set of requirements will always be incomplete. Based on this conclusion, we decided to focus on deriving changing requirements from observing users during their interaction with already existing systems.

We use the term context adaptive system in contrast to self adaptive system to emphasize our focus on the systems’ context instead of on its adaptivity. Other approaches exist that help analysts to understand the consequences for the adaptivity of a system (e.g. [32]). We assume that these approaches could benefit from the output of our observations.

III. CONCEPTS: OBSERVATION OF ANONYMOUS USERS FOR REQUIREMENTS ELICITATION

Our goal is the early, systematic, and pro-active support of continuous improvement and evolution of a context adaptive system. Figure 1 gives an overview of our proposed solution in this scenario.

Initially, an operator rolls out a new context adaptive system. The first version is based on an initial set of requirements. Our proposed solution adds an additional task for the analyst: The specification of the expected user behavior (Setup (1) in Figure 1).

During operation in its complex environment, users of the context adaptive system evolve new requirements. This can happen when users identify new ways of using the system in its environment or when competitors launch related services. In this situation, the operator can discontinue the development of the system or adapt it to the changed requirements of its users; future systems might even adapt themselves.

Our concept is based on the assumption that users change their observable behavior during interaction with the context adaptive system, when they evolve new requirements. We propose that operators regularly conduct observations to identify deviations from the expected User Behavior (Observation (2) in Figure 1). Our method supports the systematic preparation and the efficient conduction of such observations.

The analysis of the observation data is the third part of our proposed solution. The observation data is preprocessed. Predefined rules for analysis help to create a solid information base to discuss new requirements with the product manager who is responsible for defining which requirements get implemented in the next release (Analysis (3) in Figure 1). Combined with other sources, new requirements can be specified for the next release of the system.

A. Process of Observation

In this Section we describe the process of requirements elicitation by user observation. The process can be partitioned into three parts:

1) Setup of Observation: Observation goals are systematically derived from the initial requirements for the context adaptive system (Section III-A1 to III-A4).
2) Observation: In this step, the observation is prepared and executed (Section III-A5 and III-A6).
3) Analysis: Finally, the observation data is used to derive requirements (Section III-A7, III-A8, and III-A9).

Our systematic approach helps to gather quantitative observation data as a reliable foundation for requirements elicitation. However, the structured approach leads the observation into a certain narrowed direction. To mitigate this bias, our approach could be combined with other methods, e.g., interviews or unstructured observations.

We illustrate each step with the help of a fictive example from the airport domain: a context adaptive baggage claim.

1) Define Action Area: When preparing elicitation of requirements for the next version of a context adaptive system, it is a good starting point to visit the systems environment and to identify important areas where the system will be used. For this task we introduce the concept of Action Areas:

- **Action Area**: A delimited location, where one or more specific user interactions with the context adaptive system are expected. An Action Area can be composed of multiple smaller Action Areas.

The Action Area concept allows to focus the observation on a limited area, where relevant user behavior is expected. In our example, the Action Area is the area in front of the adaptive baggage claim.

2) Specify User Goals for each Action Area: In order to determine the user goals, existing requirements are used. Use cases, i* models, etc. are valuable sources for user goals. Sometimes it makes sense to execute Step 1 and 2 in reversed order: start with this step and then look for areas where users might want to achieve the given goals. In our example of the context adaptive baggage claim, we assume the goal “claim baggage”, the softgoal “claim baggage quickly”, and the task “pickup baggage”.

A user at a specific location can have different kinds of goals: Some of the goals are location-independent, while others are dependent on a special position. We focus on the goals that are dependent on a position. **Claim baggage** is an example for this type of goals because it can only be reached at the baggage claim.

3) Define the Set of Expected Actions: With Action Areas and user goals specified, the analyst needs to derive the expected actions a user supposedly will perform in order to reach a given goal:

- **Set of Expected Actions**: A finite set of defined user interactions with the context adaptive system that are expected in an Action Area.

Often, the Set of Expected Actions can be derived from the main course of a use case. Best results can be achieved if this is done in the working context of the system [33].

In our example, we expect passengers to wait in front of the baggage claim until their baggage arrives. Then we expect them to approach the baggage claim, get ready for pickup, pickup their luggage, and leave. We define the Set of Expected Actions in this scenario as:

<table>
<thead>
<tr>
<th>Set of Expected Actions</th>
<th>Actions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>pickup baggage, leave with baggage</td>
<td></td>
</tr>
</tbody>
</table>

4) Specify the Expected User Behavior: In this step the analyst defines under which conditions (i.e. in which context) a specific action of the user is expected.

- **User Behavior**: The set of observable actions a user could perform in an Action Area.

The expected User Behavior should be defined based on specific requirements. It links the Expected Actions of a user in an Action Area with context attributes and observable behavioral attributes of the user. The user context is important, when reasons for deviations in the observed user behavior are to be determined. Robson refers to this activity as defining the coding scheme [18] and describes a trade-off. On the one hand the coding scheme has to reflect the specific research question (i.e. if new requirements for the specific system under investigation have evolved in its working context). On the other hand, creating such a coding scheme is difficult and time consuming. We suggest a brainstorming session for an initial set of behavioral and context attributes. A framework (e.g. Action Theory [19]) or catalogue of important attributes can help to achieve higher completeness. Reusing an existing coding scheme is definitely an option for later observations of similar systems in related contexts. Figure 2 gives an overview of behavioral attributes relevant in our evaluation scenarios. Others may find this overview useful, but might need to extend it. Concerning the context attributes, we recommend existing taxonomies or frameworks as described in [19], [26]. Since the complete set of relevant context attributes is hard to determine and as it might change, our method allows to readjust the context attributes between two observation sessions.

The expected User Behavior is specified by the following information aspects:

1) Action from the Set of Expected Actions.
2) Description of the User Behavior while performing the action (based on the behavioral attributes in Figure 2).

These attributes are part of the observation.
3) Definition of the expected values of the behavioral attributes (based on the requirements for the context adaptive system).

4) Context attributes that define the context during the execution of the action. These attributes are part of the observation.

5) User group the observation should focus on. Optional if all users should be observed.

In our example, we assume the following requirement for the adaptive baggage claim:

The baggage claim shall adjust the speed of baggage so that each piece of baggage is in the immediate proximity of each passenger for one second.

Table I gives the expected User Behavior for our example. The airport wants to focus on business travelers, therefore we want to optimize the baggage claim for this user group.

5) Prepare observation: Based on the expected User Behavior, observation forms are prepared. Keep in mind that the observations must be easy to interpret if a large number of observations is expected. The expected User Behavior defines the user group and the behavioral attributes that should be observed. In our example, business travelers at the baggage claim, in front of their baggage (position) should be observed. Observers should focus on the posture and how long the subjects stay in front of their baggage.

When preparing a specific observation, it is important to define the context attributes that should be observed. In our example, a brainstorming has lead to the assumption that the number of persons in the area is an important context attribute:

| Context attributes: context (1m) of the user |

<table>
<thead>
<tr>
<th>User</th>
<th>Posture</th>
<th>Duration of stay in front of baggage</th>
<th>Other persons in context (1m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Passenger A</td>
<td>standing</td>
<td>1 sec</td>
<td>0</td>
</tr>
<tr>
<td>Passenger B</td>
<td>moving</td>
<td>4 sec</td>
<td>3</td>
</tr>
</tbody>
</table>

7) Preprocess Observations: In this step the observed attributes are mapped to the expected user behavior. Deviations are logged in order to create a quantitative structure of data. This allows to determine the average deviations and to identify hot spots.

In the baggage claim example, we can combine Table I and II. Passenger A behaves as expected, whereas passenger B’s behavior differs from both the expected posture and the expected duration of stay.

8) Derive new requirements from observations: Based on the preprocessed observations, the first task is to determine whether significant deviations in user behavior exist. If this is the case then the data should be analyzed to determine whether new requirements exist. The following questions help to interpret the observations, derive new requirements, and assess possible options for action:

1) Does the user goal depend on the user’s location (Action Area)?
2.a) Does the user goal depend on the user’s environmental context?
2.b) If so, is the user goal hindered by the user’s environmental context?

Table III gives an overview of the important cases based on these questions. Each case defines a relevant class of indicators for possible new requirements.

In Case 1 the location is important to reach a user goal, but there are no visible influences from the context that explain the deviation in user behavior. Either a context attribute is
missing in the observation or the context adaptive system fails to serve customers in the desired quality and general improvement (e.g., of usability) should be considered.

Case 2 describes problems that are independent from the Action Area. Perhaps users need context specific assistance beyond the scope of the context adaptive system. Possibly the context awareness of the context adaptive system can be used to create additional value by offering context sensitive assistance, e.g., directing the way to beverages on hot days.

In Case 3 the (soft) goal of the user depends on context, e.g., the priorities of the user change according to context. In the baggage claim example, a public announcement (speakers) could change the softgoal *claim baggage quickly*, because the user might want to listen to the announcement. Observations in this case could lead to new adaptation rules, e.g., slowing down the baggage during announcements.

Case 4 depicts a context attribute that hinders the user in reaching a goal. A context adaptive system could try to compensate this interference. In the baggage claim scenario, passenger B might be hindered to reach his baggage by the other persons around. The adaptive baggage claim could compensate this problem by slowing down the baggage in crowded sections.

9) Derive new Requirements for next Version: Based on the weighted indicators, requirements are specified. In this step, traditional approaches to requirements elicitation can be applied to capture missing details. In this case, the indicators help to approach the right stakeholders with good questions. The quantitative structure of indicators helps to prioritize these efforts. In our example, a new requirement could be derived:

When more than three persons are present in one section (2m) of the baggage claim, the baggage claim shall adjust the speed of baggage such as that each piece of baggage is in the immediate proximity of each passenger for three seconds.

B. Cognitive and software tools

Our goal is to support observation of a large number of users. Thus, our process should be supported by tools. Such tool support should have the following features:

1) Allow analysts to define Action Areas, User Goals, the Set of Expected Actions, and the expected User Behavior step by step.
2) Support the creation of forms that allow to document observations (user behavior and context).
3) Allow submission of observations from multiple observers.
4) Allow analysis of the observed data.

For our preliminary evaluation a prototype was implemented as a web application. A wizard guides the user through the process of preparing the observation and creating observation forms. Observers can use these forms to submit observations through a simple web interface. The simplicity of the interface allows observers to open the observation forms in the web browser of a smartphone (see Figure 3). In the future we plan to extend the web interface to allow other systems (i.e., sensors) to submit observation data. For data analysis, we currently export the observation as a csv file and load the data into a spreadsheet.

IV. Evaluation

A. Evaluation Goals and Method

The goal of the evaluation is to gain insight about the application of our approach. We want to

- Determine the feasibility of our approach.
- Evaluate the scalability of the approach.
- Identify opportunities for continual improvement.

In order to show the feasibility, we approach each part of our process separately. During setup of observation we need to investigate, if a typical analyst will be able to create suitable observation forms. During observation we need to investigate, if it is possible to observe and document relevant behavioral and context attributes. Finally, during analysis, we need to investigate if the observation data can be analyzed and if requirements can be derived.

The scalability of our proposed solution is also evaluated based on the three parts of our process. In the setup phase, we do not expect any threats to scalability. The effort in this part should be in the same magnitude as other requirements engineering activities, with the exception that the analyst should visit the location. Thus, we watch for unexpected efforts in this part during evaluation.

During the analysis, we do not expect threats to scalability, as well. If our approach is feasible, it will produce observation data that is easy to analyze. The amount of deviations has to be calculated and relations between deviations in user behavior and context influences have to be determined. The first part is done by the spreadsheet, the latter part is still easier than analyzing transcribes of numerous interviews.

The only part of our approach that could fail to scale up to real world problems is the observation. Here, we need
to get an impression on how many behavioral and context attributes can be observed by one observer. We also need to investigate the frequency and duration observable events can have.

**B. Evaluation Context**

Our observation approach was evaluated in two scenarios: a basement garage and a bus stop both situated in the center of Hanover on a workday in the afternoon. These locations offer a complex environment with numerous contextual influences. Three students took part in the evaluation: While two students performed the observation, the third student (the analyst) performed the setup and analysis. At the time of the evaluation, all students were about to finish their Masters in computer science. Only the third student was familiar with the foundations of requirements engineering.

1) **Evaluation Scenario I – The Basement Garage:** Our first evaluation scenario is situated in a basement garage with two floors. The context adaptive system is a guidance system. Two LED arrows show the direction to a floor with free parking lots and are located at the entrance of the first floor. This system adapts itself by showing users the direction to the highest floor with empty parking lots. The analyst defined two Action Areas: the basement garage as a whole and the first floor.

2) **Evaluation Scenario II – The Bus Stop:** The location of the second evaluation scenario is a bus stop. The analyst defined the observation goal to investigate how the number of people boarding and getting off the bus is affecting the waiting time of a person who intends to board the bus. Currently, there is no context adaptive system present in this scenario. We planned to investigate, if our approach helps to find out whether a context adaptive system could be useful in this Action Area. Such a context adaptive system could balance the passengers between all the doors of a bus.

The expected user behavior is that it takes a user ten seconds to enter the bus. The analyst also expects that other persons leaving or entering the bus will hinder the user.

**C. Execution and Descriptive Statistics**

Table IV gives an overview of the observations. Each row represents an Action Area. For evaluation, we observed only context adaptive systems with simple user interaction. Therefore, we have only one behavioral attribute per Action Area (choose first free floor, choose first free parking lot, enter the bus in 10 sec). Context of the environments is more complex. The analyst identified the following relevant context attributes:

- **Basement Garage, entrance:** a) Free parking lots exist on the first floor and b) free parking lots exist on the second floor. Note that these context attributes are monitored by the context adaptive system and could easily be observed on the LED arrow display.
- **Basement Garage, 1. floor:** a) Number of free parking lots and b) number of free parking lots next to pillars (users might favour parking lots not directly next to pillars because of the lower risk of damaging the car).
- **Bus Stop:** a) State of the door, b) number of leaving persons, and c) number of entering persons.

For the Basement Garage, a deviation in the expected user behavior at the entrance of the basement garage was observed, without visible influence of context (cf. case 1, Table III): Users choose the second floor, even if free parking lots are available at the first floor. Observations in the second Action Area shed more light on this phenomenon (case 3): Users omit free parking lots adjacent to pillars. Our approach led the analyst to two candidate requirements: i) the system shall equally distribute boarding and deboarding passengers to doors.

**D. Discussion**

The evaluation scenarios differed in the observations per time and in the observed attributes per observation. In the Basement Garage Scenario, many observations could be done in a short time period. Sometimes it was difficult to capture the relevant context attributes in the short time period between two evaluations on the first floor.

The scenario shows a weakness of the prototype implementation: It was planned to do the observation with two smartphones, but there was no signal in the basement. The students performed the observations with pen and paper and transcribed the observation data into the tool later. Thus, the evaluation was not affected by this weakness. When applying
our approach in practice, the additional effort of transcribing the data endangers the efficiency of our approach.

While documenting the choice of the garage floor, it was possible to capture both the user behavior (choose garage floor) and the relevant context parameters (by observing the display of free parking lots). The high frequency of incoming cars allowed to capture a meaningful amount of datasets in only a short time period. Filling in the data took considerably longer than doing the observation.

In the second Action Area, on the first floor of the basement garage, it was more difficult to capture behavior and context attributes. Because of the large and confusing area it was hard to capture the current state of the context attributes (free parking lots) during the time of one observation. In addition, the values of the context attributes changed frequently, because several cars arrived and left per minute. These events (i.e. occupying, leaving parking lots) had to be observed in parallel at many locations. Using video cameras or other sensors for assessing the context attributes would be helpful in such situations.

In the bus stop scenario, user behavior and context attributes could easily be observed, because of the small and well defined Action Area. The documentation of the observations proved to be difficult. The duration of the events to be observed was less than one minute. Documenting these observations with smartphones took longer than expected. In addition, the observers reported problems with the latency of their smartphones’ web browser and Internet connection. Loading an observation form took too long in many cases. With some training, the observers managed to load the required forms beforehand.

Despite these problems, the application in the two scenarios proved the feasibility of our approach. Observers are able to do many observations, even in complex situations and when multiple behavioral and context attributes are involved. If too many context attributes have to be considered, technical solutions can help. It is also possible to partition the context attributes over different observers. Data analysis is straightforward and can efficiently be done as long as the data can be loaded into a typical spreadsheet. It took the analyst only one hour to create a rudimentary scatter plot that allowed to assess, whether the choice of a parking lot is correlated to the relation of free parking lots and free parking lots next to pillars. Thus, we assume that the approach scales to real world problems.

E. Threats to Validity

1) Internal Validity: We tried to ensure that our results follow from the data. In our evaluation, the useful observation forms, the relevant observations, and the analyzable observations might follow from the participants’ personal taste rather than from our approach. As we had no control group, we cannot fully eliminate this threat. The strict separation between the roles of analyst and observer was done to minimize this effect.

2) Construct Validity: We tried to ensure that the theoretical constructs were well mapped to the evaluation scenario. A potential flaw is that we did not observe a context adaptive system in the bus stop scenario. We did this on purpose, because we wanted to evaluate whether our approach is suitable to investigate the demand for a context adaptive system and to define initial requirements. The context adaptive system we choose for evaluation was very simple. It is possible that more complex systems are more difficult to observe. We assume that experts are required to define the relevant behavioral and context attributes for complex environments. Still, our approach helps to employ these experts in an efficient way during observation setup.

3) Conclusion Validity: We are confident that other observers would not affect the outcome of the study. The observation forms are simple and only minimal training is required. Concerning the reproducibility of our study it is more important to investigate whether other analysts would choose similar behavioral and context attributes for observation and come to similar conclusions based on the observation data. Therefore, we confronted two more analysts with this assignment. It was hard to describe the setup task sufficiently, thus we did not reach satisfactory results here. During the analysis phase, both additional analysts came to similar conclusions.

4) External Validity: Students are the target group of our approach, when it comes to observing. We would expect organisations that apply our approach to hire students for this task. Again, we are more concerned with the setup of the observation and with the analysis of the data. We choose a student for setup and analysis, because this task would even be easier for a professional requirements engineer.

F. Lessons Learned

The observation effort depends on the number of context attributes, the size of the Action Area, and the number of parallel events. Despite the limitations described in the discussion, we did not overburden our observers in the evaluation scenarios. Concerning the number of context attributes, we learned that if in doubt, the context attribute should better be taken into account. Ignoring a relevant context attribute causes more harm than observing a few irrelevant ones. Changing context attributes between two observation sessions could lead to incompatible data during analysis. The advantage of having the relevant context attributes makes the loss of data from previous observation sessions due to incompatibilities acceptable. Thus, we suggest to decide before each observation session, which context parameters should be observed.

If the context adaptive system is unable to satisfy a user requirements, the user will behave more or less different than expected. Among other variables, this is influenced by
the type of system. Our approach works best with systems that cause a large difference between the expected and actual behavior of its users, when requirements are not satisfied.

Especially, when using technical solutions for user observation, legal aspects have to be considered. These considerations should also include the storage of data. Legal aspects depend on the context of the observation and are not in scope of this paper.

V. CONCLUSION AND OUTLOOK

This paper presents an approach for requirements elicitation that does not require the active involvement of the systems’ users. Our approach focuses on deriving new requirements from the users’ behavior and context. Related work in observing users for requirements elicitation usually relies on explicit dialogues with users. In contrast, our approach is distinguished by the absence of such interaction.

Knowledge derived from pure user observation is limited and only of heuristic nature. How a user perceives her context and which factors influence her behavior cannot be answered clearly without a dialogue. Finding a correlation between contextual influences and user behavior seems to be vague and imprecise. Nevertheless, even the dialogue with a user does not always reveal all requirements. Especially, if the user is not aware about how context attributes influence her requirements in a given situation, our approach is useful for identifying new requirements or indicators for new requirements. These requirements and indicators can then be validated and refined with traditional approaches.

Our preliminary evaluation showed the feasibility of the approach in two evaluation scenarios. The results reveal no reason to doubt the scalability.

This paper presents an important first step toward requirements elicitation support for context adaptive systems based on user observations. Nevertheless, future research is needed in all major steps of the proposed process – the setup, the observation, and the analysis of the observation data.

Setup of observation: One of the biggest challenges is the definition of suitable context attributes. A complete, detailed and concrete system or taxonomy of context attributes would be of great value for this task. In addition, methods and heuristics could help to limit the amount of context attributes that need to be observed. This would simplify the measurement and increase the quality of the observation data.

Observation: One of our next goals is to take advantage of sensors and smartphones for automatically capturing the context and the user behavior. In addition, fast-paced simultaneous activities could be recorded by video cameras. We plan to empirically compare our structured and unobtrusive approach with non-structured observational data gathering.

Analysis: Once huge data sets with observations exist, a more sophisticated approach to analysis and data mining should be considered. There exist promising approaches to use data mining concepts for requirements engineering.

In conclusion: A fundamental need for method and tool support for requirements elicitation for context adaptive systems exists. Our research and prototype presented in this paper as well as our experiences could be helpful for researchers and practitioners who focus on this area.

ACKNOWLEDGEMENTS

This work is partly funded by the NTH School for IT Ecosystems. We thank everybody who supported our work, especially Kurt Schneider who made this research possible. Special thanks go to Christian El Boustani whose Master’s thesis is important preliminary work for our concepts in Section III, led to the prototype and to the observation data we present in Section IV. Figure 2 is based on and Figure 3 is taken from this thesis.

REFERENCES


1NTH (Niedersächsische Technische Hochschule) is a joint university consisting of Technische Universität Braunschweig, Technische Universität Clausthal, and Leibniz Universität Hannover.

2Available in German at: http://static.se.uni-hannover.de/f/pub/File/pdfpapers/Boustani2010.pdf