Interdisciplinary Design Patterns for Socially Aware Computing

Harun Baraki*, Kurt Geihs†, Christian Voigtmann‡, Axel Hoffmann§,
Romy Kniewel¶, Björn-Elmar Macek‖ and Julia Zirfas**
Zentrum für Informationstechnik-Gestaltung (ITeG)
University of Kassel, Germany
*Email: baraki@vs.uni-kassel.de, †geihs@uni-kassel.de
‡christian.voigtmann@comtec.eecs.uni-kassel.de, §axel.hoffmann@uni-kassel.de
¶r.kniewel@uni-kassel.de, ‖macek@cs.uni-kassel.de, **j.zirfas@uni-kassel.de

Abstract—**The success of software applications that collect and process personal data does not only depend on technical aspects, but is also linked to social compatibility and user acceptance. It requires experts from different disciplines to ensure legal compliance, to foster the users' trust, to enhance the usability of the application and to finally realize the application. Multi-disciplinary requirements have to be formulated, interwoven and implemented. We advocate the use of interdisciplinary design patterns that capture the design know-how of typical, recurring features in socially aware applications with particular concern for socio-technical requirements. The proposed patterns address interdisciplinary concerns in a tightly interwoven manner and are intended to facilitate the development of accepted and acceptable applications that in particular deal with sensitive user context information.**

I. Introduction

Design patterns can be understood as reusable tools that help solving recurring problems in engineering tasks. They emerged from the work of Alexander et al. in the field of architecture [1]. Although Alexander himself questioned the relevance of his architectural patterns [2], ironically nowadays design patterns play an important role in many domains such as software engineering [3] and human-computer interaction [4]. Design patterns may simplify and speed-up the development process as they indicate a direction for a solution that is mostly based on thorough knowledge and experience for the particular issue. At the same time they contribute to a common vocabulary that lets experts communicate more easily about design questions. They also help to avoid potential mistakes.

This paper is about interdisciplinary design patterns for applications that collect and process sensitive personal data. Such applications make use, for instance, of context data and personal preferences and profiles in order to adapt to individual users and their situations, or to analyze behavioral patterns, motion profiles and other characteristics. We will refer to this type of software hereafter as *user-aware applications*.

Personal data might be processed on servers and devices that are not visible to the user, and applications might execute actions the user is not aware of and perhaps would not even allow. While this tight interweaving into the users' everyday lives offers a wide range of exciting application opportunities, it also demands the consideration of non-technical, societal aspects. Questions of privacy, trust, usability, legal compliance etc. have to be addressed during application development. To meet this challenge and to provide proper solutions, software experts have to collaborate with experts from other disciplines.

A comprehensive interdisciplinary development methodology for the design of user-aware applications has been worked out in the VENUS project [5]. Four disciplines were represented in VENUS, i.e. computer science, jurisprudence, ergonomics and trust research, contributing to the research of development methods and tools for user-aware applications and taking into account theories, methods and tools described in the context of socio-technical system design. While ergonomics and computer science are focusing on the usability and technical implementation of applications, jurisprudence and trust research emphasize the embedding of technology in its societal context. Laws and principles define the rules that both the user and the software manufacturer have to comply with. The trust dimension includes the users' perspectives by analyzing their doubts, uncertainties and expectations and tries to strengthen the users' confidence in the application by appropriate measures. Note that this mix of disciplines was due to pragmatic project constraints. Additional disciplines could have been, for example, sociology and psychology. Nevertheless we claim that the four involved disciplines cover a very large part of the relevant concerns.

In the course of the VENUS project, three demonstrators were designed and implemented by separate interdisciplinary development teams in different application domains, i.e. mobile computing, social networking, and ambient assisted living. Thus, recurring patterns were identified related to concerns about social awareness and societal embedding of the applications. Domain experts helped us in an iterative process to consolidate the initial list of patterns and to pick out those that are generally applicable. The contribution of this paper is not only a presentation of a design pattern collection. A further intention of this work is to argue for truly interdisciplinary design patterns that help to address interwoven, crosscutting design concerns for socially aware computing applications.

The remainder of this paper is organized as follows. The following section discusses related work and justifies the need for further research. Thereafter, we explain the development of the design patterns and highlight those steps of the VENUS methodology that are essential for societal compatibility and that are captured by our design patterns. Subsequently, our pattern template is introduced, before we present four examples for design patterns that focus on interdisciplinary, cross-cutting design aspects. For reasons of space only four design patterns are presented here. More patterns are contained in our technical report [6]. Section V discusses the role of interdisciplinary design patterns in software engineering. We conclude the paper with a general reflection on the significance of our design patterns and an outlook to future work.

II. Related Work

In the following we focus primarily on related work that adopts a clear interdisciplinary viewpoint on user-aware applications in domains such as Ubiquitous Computing (UC), Ambient Assisted Living (AAL) and Social Media.

One of the first publications proposing privacy guidelines for web sites and ubiquitous computing applications can be found in [7]. Langheinrich introduces several fundamental principles such as the principle of *notice* in which users have to be informed by announcements first and foremost if data is collected about them. Further principles like the *choice and consent* principle were justified on the basis of the EU Data Protection Directive 95/46/EC and the fair information practice principles (FIPP) that were first described in the Privacy Act of 1974 (5 U.S.C. 552a as amended). Langheinrich suggests approaches of a general manner. As the title of the work implies, principles are proposed, but not design patterns that support their concrete implementation.

In [8], Lahlou et al. propose nine guidelines, called European Disappearing Computer Privacy Design Guidelines. Most of them, i.e. *Think Before Doing*, *Good Privacy Is Not Enough* and *Re-visit Classic Solutions*, give some clues which thoughts developers should take up in general. Others are geared to the principles introduced by the OECD [9]. Altogether, Lahlou et al. indicate problems and challenges developers have to tackle. Many of these guidelines can be considered indeed as motivation for design patterns, but not as intelligible instructions developers can apply.

An extensive work on patterns for social media is presented in [10]. Crumlish et al. introduce social patterns and define them as components and pieces of interactivity that are the building blocks of social experiences. The authors focus on motivating people to take part and interact in social networking by proposing design patterns for social interfaces. Their pattern language goes beyond typical HCI patterns, such as those of [11], as it accentuates the overall state and well-being of customers more substantially. Nonetheless, they do not integrate explicitely the perspectives of law and trust. The trust dimension is rather considered on the level of the user interface, but not in the backend where software manufacturer and service provider access personal data.

Chung et al. introduce in [12] 45 design patterns for ubiquitous systems, grouped into the four groups *Ubiquitous Computing Genres, Physical-Virtual Spaces, Developing Successful Privacy*, and *Designing Fluid Interactions*. Their intention is to provide an initial list of design patterns that can be enhanced and extended. In the context of this work especially the fifteen design patterns devoted to privacy and the eleven patterns related to fluid interactions are of interest here. Their set of design patterns did not emerge from the authors' own software projects but from studying the relevant related literature. For example, their *Fair Information Practices* pattern lists the aforementioned practices of the 1970s as solution and refers to Langheinrich's work [7]. Other patterns reproduce principles like the choice and consent principle, but do not provide clear instructions how to put them into practice. In contrast to our approach, Chung et al. consider privacy and usability separately. In our work, we take requirements from different disciplines at the same time into account in order to resolve conflicts instantaneously and allow for an improved joint design.

Ruiz-López et al. propose in [13] various patterns to address non-functional requirements in ubiquitous computing systems. Most of them refer to adaptivity, reliability and security, but two of them, i.e. the *Pseudonymity* and the *Human Factor* patterns, are classified as patterns concerning ethics. The *Human Factor* pattern, for example, is a hybrid approach that allows the user to perform certain activities on his own instead of leaving it up to the software. According to this pattern, developers should consider the possibility to only assist users or to let users do things on their own to improve, inter alia, their well-being. Here again, detailed information about the pattern and possible scenarios and examples are missing. In fact, Ruiz-López et al. present an excellent analysis, but their conclusions are formulated on a very abstract level and without concrete connections to interdisciplinary design guidelines.

Considering the principles, guidelines and patterns introduced in this section, two main problems can be identified: firstly, the descriptions are often too general and too vague. If indeed the aim is to implement abstract requirements, that have for instance a legal background, the utility of the presented patterns is rather limited. An approach is required that provides support from the initial requirements through to their realization. Secondly, interdisciplinary design patterns have to integrate cross-cutting aspects from different disciplines such as human-computer interface, computer science, trust research and law. The previous works only emphasized one or two of these disciplines per design pattern, but did not include the forces from other disciplines that could oppose them. The goal of our proposed design patterns is to consider cross-cutting concerns for social awareness concurrently in an interwoven manner in order to avoid conflicts and increase efficiency.

III. Pattern Development

The methodology created in VENUS supports interdisciplinary teams that include, inter alia, lawyers, software experts and trust and usability engineers, to work together in an efficient and structured manner [14]. The following sections present first the demonstrators from which the design patterns emerged from and that were developed by our interdisciplinary teams in the course of the VENUS project. After that, the key factors of the VENUS development methodology that were essential for the evolution of our design patterns and that were applied in the demonstrator projects, are introduced.

A. Demonstrator Projects

The three demonstrators implemented for evaluation purposes are called Meet-U, Connect-U and Support-U [15], [16], [17].

Meet-U is a mobile application that supports users to organize meetings. After installation and registration a user can invite other users to participate in an event or create events on his own initiative. It is also possible to take part in public events or to purchase tickets. A specific feature of Meet-U is its context awareness and adaptivity. For example, navigation will be started automatically depending on the time required to arrive at the event and depending on the user's preferences concerning transportation. Reaching the destination the application will connect to the local Web service of the event organizer and will show helpful information like the indoor map of the location to the user [15].

Connect-U assists conference participants in organizing their schedule, in recalling their social contacts they had during the conference and in sharing information with their colleagues. At first, the participants create a profile and register it with the Connect-U website. During the conference the participants wear RFID tags that recognize whether two persons are communicating with each other. RFID readers in every room monitor the movement of the participants. On the Connect-U website participants can check which lectures and talks they are going to visit, which ones they attended and with whom they talked during the conference. They can ask for more information about their dialogue partners by invoking their profiles. If two participants accept each other as friends on Connect-U, they can even look up the other's current position [16].

Support-U is an application in the field of Ambient Assisted Living (AAL) that supports elderly people in their everyday life. To this end, unobtrusively deployed sensors are reporting on the status of the flat and provide information such as activated electrical appliances, open windows and doors and the current temperature, humidity and light intensity. Blood pressure, pulse and movement intensity are transmitted by sensors attached to the elderly person. Family members or care workers who look after this person can display any information received in a clearly arranged overview on their tablet computer. Easy-to-understand colors and symbols facilitate the perception of important status information. In this way, on-site checks by the caring person can be reduced and the elderly person may enjoy living at home instead of residing in a nursing home [17].

B. Incorporating Normative Requirements

The core of the VENUS approach is an iterative development approach consisting of the phases: analysis of needs, requirements management, conceptual design, software design with implementation and in-situ evaluation [14]. The analysis of needs implies the development of specific application scenarios that provide a common understanding about the required application functionality and usage context. The scenarios were derived from application objectives by involving experts and users in joint workshops. Potential users validate the scenarios to detect misunderstandings and misconceptions in the functionality, usage and socio-technical embedding of the application in an early stage.

We are particularly interested in the requirements management and elicitation phase as it is here that requirements from various disciplines are elaborated and merged. Besides usual functional and quality requirements, requirements for legal compatibility, usability and trustworthiness are considered. Domain experts derive these requirements from normative sources by concretizing the provisions with regard to the technical system. The goal is to translate these provisions to technical requirements and then to merge them in a subsequent step.

For example, let us look at the legal dimension. The procedures of the other involved disciplines will be explained briefly after that, before we finally switch to the merging of the different requirements. The following example also intends to give a feeling for the efforts each domain expert has to make and underlines the necessity of interdisciplinary design patterns that help to capture this knowledge for future projects.

After realizing the application goals and scenarios, a lawyer might consider the constitutional laws that can be applied to the application under development. Mostly, the general rights of personality will be taken into account. Depending on the intended application further laws may be included, for example, the fundamental right of ownership and the right of freedom to expression and information in case that the software is used to publish articles and pictures. Subsequently, legal specifications have to be derived from the relevant constitutional laws. They can be elaborated by interpreting the constitutional laws with respect to the social functions affected by the application. In addition, case-laws can be utilized. In the event of the general rights of personality these include in particular the right to informational self-determination and the right to confidentiality and probity in information technology systems. The next step is to find concrete criteria that the application has to fulfill to be in line with these legal requirements. In case of informational self-determination such criteria could be transparency, controllability and secrecy. Such kinds of criteria characterize and shape the application. They are not only regarded in the stage of requirement elicitation and design, but also in the evaluation phase. As with provisions, criteria are described in the language of law. In contrast, the next steps have to be elaborated together with software experts in the language of technology. The criteria have to be transformed into functional requirements or, more explicitly, to technical design goals. For instance, to enable transparency the usage of personal data by the application can be made accessible to the user in log files that are protected by encryption.

In the same way usability engineers have to work out technical goals. Instead of using laws, they follow policies and norms for ergonomic design. A typical example is given by the ISO 9241 series of standards, known as ”Ergonomics of human-system interaction” [18]. In contrast, trust engineers, i.e. experts for increasing the users' confidence in a technical artefact, determine first possible feelings of insecurity which depend on the concrete use case and context. They try to find the relevant trust dimensions that could address these insecurities and elaborate confidence-building measures by means of a multi-step process. Assuming that an application is performing several adaptations autonomously, a trust engineer could identify, for example, traceability as a relevant trust dimension. In a further step, this would be refined by incorporating controllability and comprehensibility. Based on this, functional requirements would be prepared by considering the technical details of the system and the situations of insecurity detected in the first step. The design pattern *Control of Autonomous Adaptation*, for example, is addressing autonomous adaptation in the context of user-aware application.

Finally, the functional requirements and the requirements formulated by the different domain experts have to be merged. For this purpose, the requirements will be assessed and prioritized. Furthermore, all requirements are checked for redundancy and contradictions. In case of inconsistencies, solutions or compromises have to be agreed on within the team. Then, implementation concepts can be worked out, realized and evaluated. The process details can be found in [14] and [19]. A summary is provided in [5].

C. Pattern Extraction

Developing socially acceptable user-aware applications generally is a time-consuming, complex and expensive task, even if using the VENUS methodology that provides tools, methods and a systematic approach to develop proper solutions. Our goal is to capture and share in design patterns the knowledge and the solutions generated by the various disciplines to reduce costs, efforts and conflicts in future projects. It shall speed up and simplify the interdisciplinary development process and provide directions and starting points to inexperienced development teams.

As already mentioned, three demonstrators have been designed and implemented in the course of the VENUS project by separate development teams in different application domains. We analyzed the development phase of the three demonstrators together with their responsible teams and determined first common burdensome and crucial points that the interdisciplinary development teams had to address in the early stage of their projects. In this phase, mainly the functional requirements and application scenarios were given. This stage generates the *Intent* and *Motivation* parts of our design patterns. They refer to the problem that the design pattern has to resolve. The next step was to ascertain the main legal criteria, trust dimensions and ergonomic principles that were included by the development teams to work out a socially acceptable solution. The *Forces and Context* section of the design patterns summarise these. Details that would restrict the design pattern to a very limited number of use cases were omitted. Furthermore, the objective is to indicate the different directions and forces the developers have to take into account, but not to list complete provisions, laws and norms. In fact, they might confuse developers and team members not familiar with those disciplines instead of giving them an initial overview and roadmap. In a final step, the technical goals and implementations elaborated by the development teams were examined. We realized that certain implementation aspects are already addressed by existing human-computer interaction (HCI) patterns. Hence, our design patterns refer also to existing HCI patterns that help implementing the solution. The detailed structure of the design patterns is introduced in the next section.

After identifying candidate design patterns, the different development teams assessed them first against their applicability in the different demonstrator projects. The remaining design patterns were rated with respect to reusability and relevance for user-aware applications in general. In an iterative process we created a consolidated collection of generally applicable patterns.

D. Template for Interdisciplinary Design Patterns

We describe our design patterns by using a template with eight sections. These include, inter alia, typical pattern fields such as intent, motivation, solution, consequences and forces and context of the pattern. But while other pattern collections address single topics unilaterally and from an isolated view, our patterns comprise the abstract as well as the concrete aspects. The solution section of our patterns presents approaches for design and implementation, while the other parts, particularly the forces and context section, emphasize the abstract layers. These may encompass normative goals, norms, and other non-functional criteria. The sections of the template are:

- *Name*: A memorable, descriptive name is chosen that can be connected easily to the pattern's main feature. This will help to recall the pattern. Furthermore, it will simplify and speed up the collaboration between developers and experts from different areas. We will use the pattern's name as the title of its description.
- *Intent*: The Intent indicates concisely when the pattern should be applied. Hence, the problem that is handled by the pattern is described. Note that our proposed patterns focus on normative provisions and non-functional criteria. Besides, the Intent points out in one or two sentences how the problem will be solved. This supports users in browsing the patterns efficiently. In total, the Intent should not exceed five sentences.
- *Motivation*: The Motivation illustrates the problem by giving an example. This example should be representative for the situation and the broad class of scenarios the pattern is addressing.
- *Forces and Context*: The Forces and Context section highlights the non-functional aspects that would come off

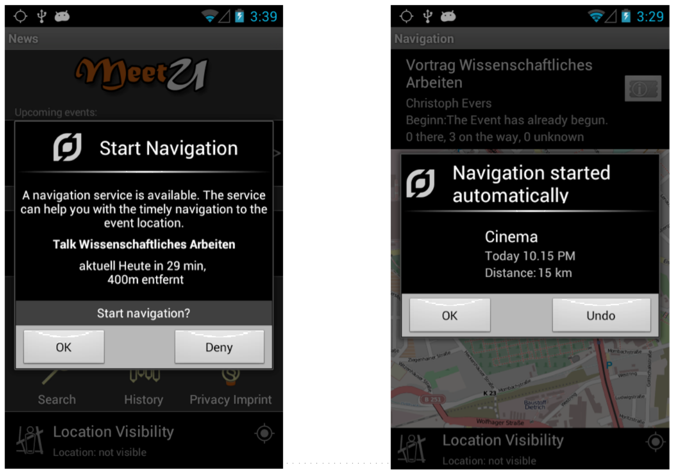
badly in practice if the pattern would not be applied. In particular, it names the conflicting sets of goals, especially if a functional requirement encounters normative policies. The fundamental task of the pattern is to solve this conflict in the best possible way. We will name the normative goals explicitly and will justify the pattern's use by them.

- *Solution*: The Solution section is divided into two parts. The first part proposes a general approach and is not limited to a certain application example. The level of detail is not too high since the concrete implementation depends on the actual application and because it would be out of scope of a pattern collection. However, the description is sufficiently complete for software designers and software developers to put it into concrete design goals. The second part of the Solution section shows the application of the pattern to our demonstrators.
- *Consequences*: The main consequences of the application of the pattern are subsumed in this section. These may include positive as well as negative consequences. We will point usually to consequences related to non-functional aspects. Indeed, we may mention technical consequences if it is of key importance that the development team has to be aware of it.
- *Related Requirement Patterns*: Our patterns also point to existing well-known requirement patterns that assist the development team to identify and document software requirements during the requirement management phase. Requirement patterns are an approach to reuse recurring requirements [20], [21]. They contain templates to describe standardized requirements and other relevant information in tabular form [22], but have to be adapted to the current project. The requirement patterns we refer to are listed in [23].
- *Related HCI Patterns*: The cited HCI patterns shall support developers in applying our design patterns. They were not developed in the VENUS project but were taken from [4] and [24].

IV. Patterns

In this section we present examples for interdisciplinary design patterns. Due to lack of space only four design patterns will be described and exemplary solutions for two applications are presented for each pattern. The complete set of patterns can be found in our technical report [6]. The following list enumerates the design patterns identified so far:

- *Control of Autonomous Adaptation*
- *Emergency Button*
- *Trust and Transparency*
- *Enable/Disable Functions*
- *Abridged Terms and Conditions*
- *Context State Indication*
- *Data Access Log*
- *On Demand Explanation*



A navigation service is available. The service can help you with the timely navigation to the event location.

Talk Wissenschaftliches Arbeiten

aktuell Heute in 29 min,
400m entfernt

Vortrag Wissenschaftliches Arbeiten
Christoph Evers
Beginn:The Event has already begun.
0 there, 3 on the way, 0 unknown

Cinema
Today 10.15 PM
Distance: 15 km

Fig. 1. Two examples in Meet-U for adaptation notifications.

The following sections introduce the patterns *Control of Autonomous Adaptation*, *Emergency Button*, *Trust and Transparency* and *Enable/Disable Functions*.

A. *Control of Autonomous Adaptation*

INTENT: Autonomous adaptations can result in usability problems. The goal of the pattern is to prevent the feeling of loss of control. This feeling occurs with the user if the behaviour of an application is not comprehensible or if the behaviour disturbs the current interaction with the application.

MOTIVATION: The pattern helps to create understandable autonomous adaption and prevents the feeling of loss of control. For example, consider the smartphone application Meet-U that supports the user during the preparation and planning of an event, while traveling to an event, and while participating in the event. The application features autonomous adaptation to provide always the best service to the user. Therefore, the application adapts to the navigation mode if the user needs to depart. If this autonomous adaptation occurs during an user interaction, e.g. the user enters a new event or changes his profile picture, the interaction will be disturbed. In this case, the user should be in control whether the adaptation is to be performed.

FORCES AND CONTEXT:

- Controllability: To support the user's self-determination in case of autonomous adaptation, the ultimate decision-making authority has to remain with the user - otherwise the system can adapt to unintended and irreversible states.
- Plausibility: By showing the user the next adaptation step and giving him the possibility to revert an adaptation, the autonomous adaptation is no longer a pure black-box concept. The user gains not only control, but also an overview about the different states and steps.

SOLUTION: The user should be enabled to keep control of autonomous adaptations. This prevents the feeling of loss of control. Two cases have to be distinguished:

- The user interacts with the application currently. In this case the application should notify the user about the upcoming adaptation and enable the user to determine if the application should adapt. The user should have a choice to accept, decline or delay the adaptation.
- The user does not interact with the application currently. This means that the adaptation can be performed. However, the application needs to provide the user an option to revert the adaptation.

Adaptations with substantial effects on the system should be recorded in a history. Such a change may be the switching off of a monitoring system or the muting of a ringtone. The adaptation design needs to be tailored to the application domain, development platform, and target user group.

- Meet-U: Since Meet-U is a smartphone application there exist several types of notifications to inform the user about already performed or upcoming adaptations. Figure 1 gives an impression of how the adaptation notifications and control interfaces may look like.
- Support-U: In Support-U an autonomous adaption takes place by the automated selection of the room view that shows the room the person is currently located in. If the room view is manually selected by the user and Support-U wants to automatically adapt the view to the current location of the person, the user receives a notification by the application. The notification informs the user that an autonomous adaption of the room view will take place. The user is able to refuse the adaption.

CONSEQUENCES: The pattern is influenced by and influences the user interface design of the application. The adaptation notifications need to be integrated into the user interface design.

RELATED REQUIREMENT PATTERNS:

- Signaling the Function Status: The application shall show the users the status of the functions.
- Level of Automation of Functions: The application shall allow the users to select the level of automation for the functions.
- Control of Processes: The application shall confirm successful completion of processes for users. The application shall make it possible for users to undo processes. The application shall confirm input to users.

RELATED HCI PATTERNS:

- Notifications: "You must provide a method to notify the user of any notifications, of any priority, without unduly interfering with existing processes." [24]

B. Emergency Button

INTENT: This pattern enables the user to halt collection and use of his personal data in a simple one-step manner at any time. It should be applied if the application collects and uses personal data.

MOTIVATION: Many user-aware applications use sensors to collect context information in order to reason about the user's current environment and situation. This implies that sensitive personal data is collected. For example, Meet-U displays the user's own position and the position of his friends who are heading to the same destination. An easily accessible emergency button should be available in the user interface. By pressing the button the user can force the application to stop immediately collecting and sharing his position.

FORCES AND CONTEXT:

- Informational self-determination and controllability: The appliance of this pattern supports the user's right to informational self-determination by disabling any use or gathering of personal data by the application. This right has been derived from the basic personal right (Art. 2 Para. 1 and Art. 1 Para. 1 Basic Law of Germany) and gives every individual the authority to determine, when and within which limits private data should be used or communicated to others. Basically, it enables the user to maintain control of his/her own data. (BVerfGE 65, 1 (43) population census decision).
- Confidentiality: By providing a mechanism to the user to disable the collection and use of personal data, the user's acceptance and trust into the application can increase. This holds especially true in situations where the user wants strict confidentiality.

SOLUTION: The implementation and the user interface design of an emergency button depend on the application domain and development platform. The button should be easily accessible at all times. It is important to give a feedback to the user after activating the button.

After triggering the emergency button the system stops immediately collecting and using personal data. Herein, all data from which other personal data can be inferred is included. If pressing the button impairs application functionalities, the application needs to highlight these functions to provide visual feedback to the user.

- Meet-U: If running on a smartphone, the button can be placed in the context menu of the application. The application receives personal data only through a proxy or manager class. The manager class is used to encapsulate the access to personal data and to prevent unwanted access. After activating the emergency button, the manager class stops providing personal data. Furthermore, the application is not allowed to use personal data anymore. The application informs the user via a pop-up window about any consequential restrictions. Besides, an icon in the notification bar may inform the user that the emergency button is activated.
- Support-U: If the monitored elderly person does not have control over the application, a so called "button-on-the-wall" actuator is needed. If the "button-on-the-wall" has been pushed, the context data collection process, the processing of already gathered context data as well as the transmission to the monitoring party is interrupted. Furthermore, Support-U notifies both parties that the "button-on-the-wall" has been pushed and therefore no

context data can be further displayed by the Support-U application.

CONSEQUENCES: When the button is triggered, all functionalities, which require personal data, need to be deactivated. The Emergency Button Pattern can be combined with the Enable/Disable Functions Pattern which addresses similar concerns, but in a selective manner.

RELATED REQUIREMENT PATTERNS:

- Agreement to Functionality: The application shall ask for users' consent regarding the functionality before use. The application should enable users to alter their consent regarding the functionality.
- Control of Processes: The application shall confirm successful completion of processes for users. The application shall make it possible for users to undo processes. The application shall confirm input to users.

RELATED HCI PATTERNS:

- Button: "You must allow the user to initiate actions, submit information, or force a state change, from within any context." [24]
- Convenient Environment Actions: "Group these actions together, label them with words or pictures whose meanings are unmistakable, and put them where the user can easily find them regardless of the current state of the artifact. Use their design and location to make them impossible to confuse with anything else." [4]

C. Trust and Transparency

INTENT: The intent of the Trust and Transparency Pattern is to visualize sensor devices and inferred context sources that surround a user. The intention of this pattern is to reduce the unobtrusiveness of current sensor technology if needed.

MOTIVATION: Various sensors pervade our daily life and affect us in different situations and areas. Mostly, gathered sensor data and inferred context data are processed by applications without informing the user about the involved context information sources. For example, so-called smart homes and smart rooms adapt their services to the lifestyle habits of occupants by observing and learning their behaviour patterns [25], [26]. Likewise, with the aid of smart badges conference attendees can be grouped by their interests. They can be automatically informed about similar activities of other members [27], [28]. RFID sensors are used to detect whether conference attendees are talking to each other, how long their conversation took, and which talks participants have visited. This information may be processed without knowledge of the participants.

FORCES AND CONTEXT:

- Transparency: The pattern increases a user's trust in using user-aware applications by visualizing the sources for context data, respectively the sensors that surround the user. The user's perception and understanding of her environment are supported.

SOLUTION: A general solution is to add an augmented reality view to the context-aware application. For example,

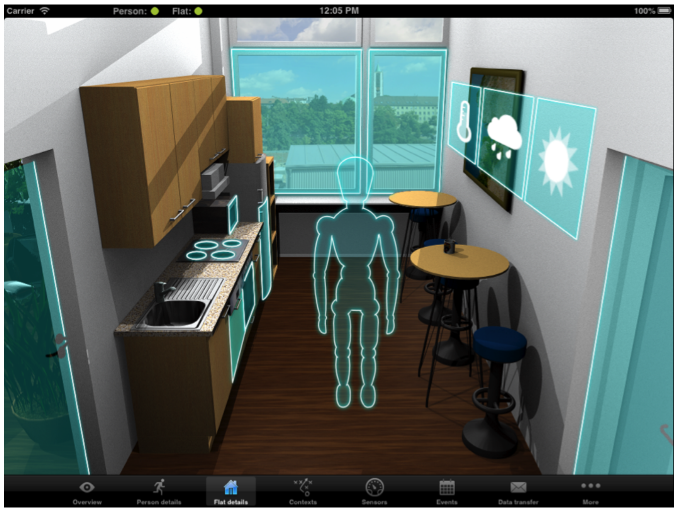


Fig. 2. The example presents how the application of the Trust and Transparency Pattern can look like. All hidden sensors and inferred context information are depicted.

the camera view of the smartphone could be used to visualize the sensors that surround the user.

- Support-U: An overlay technique is proposed that visualizes the sensors installed in the surrounding area. The overlays enable the user to recognize easily the availability and type of installed sensors. In addition, the user is able to access the current data of the sensor and its data history by simply clicking on the sensor in the overlay. An example of the implementation of the proposed pattern is illustrated in fig. 2.
- Meet-U: Meet-U's indoor navigation incorporates RFID tags and readers to be able to localize the user even when a GPS signal is not available. The user can see on the map of the building where nearby RFID readers are installed. The same view indicates that the user is localized by means of a RFID tag instead of using GPS.

CONSEQUENCES: By enabling the user to detect the sensors surrounding her, her confidence in the application will potentially increase by the provided transparency. Furthermore, the user is able to access additional information that has not been visible and accessible to her before.

RELATED REQUIREMENT PATTERNS:

- Signaling the Function Status: The application shall show the users the status of the functions.

RELATED HCI PATTERNS:

- Tooltip: "You need to add a small label, descriptor, or additional piece of information in order to explain a piece of page content, a component, or a control." [24]
- Annotation: "You must be able to attach additional information to a data point within a dense array of information, without leaving the original display context. Any interactive infographic that demands such additional information can generally support an Annotation. Layered display is easy to add to almost any platform, but some

attention must be paid to precise location ability when used in Infinite Area and similar displays." [24]

D. *Enable/Disable Functions*

INTENT: The goal of the pattern is to enable the user to explicitly agree or disagree to certain functions provided by the user-aware application. This choice can be offered to the user by the application at start-up time.

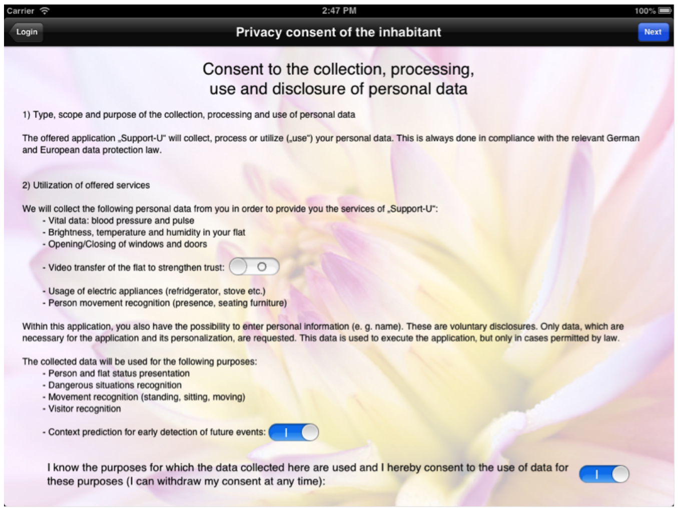
MOTIVATION: Consider users living in an Ambient Assisted Living environment: these users are surrounded by various sensors such as video cameras, motion sensors or electrical current sensors that are used to monitor the actual situation of a person. Further examples can be found in mobile applications that recommend, for instance, places of interest to the user by considering the gathered movement behaviours. With regard to these examples, it becomes obvious that sensors often unobtrusively collect highly critical and personal context data of users. Hence, the proposed pattern enables the user to decide which functions she is willing to use and which function the user renounces because she does not want to provide the personal data needed by the requesting function.

FORCES AND CONTEXT:

- Informational self-determination: The pattern considers a user's basic right of informational self-determination. This is due to the fact that a user is able to explicitly agree or disagree to a certain function depending on the personal context data needed by the function. Therefore, the user has direct control of the context data collection process. This satisfies the principles of necessity, transparency, giving consent and responsibility. They are part of the user's right of informational self-determination and are described in detail in [29] and [30].
- Traceability: The pattern provides an overview and indicates which function requires which personal context data to work properly. For this reason, a user is aware of the context data that is gathered by the sensors that surround her.
- Confidentiality: The pattern increases a user's trust in the application by offering the possibility to prevent the collection and inference of certain personal context data. Hence, a user can be sure that personal data that is critical to her is not gathered, stored or further processed by third parties.

SOLUTION: The user can explicitly agree or disagree to certain functions. For this purpose, the application has to display the crucial functions that make use of personal and context data, together with the corresponding data types. A possible way of displaying these functions and the required data may be the use of a privacy consent form.

- Support-U: Figure 3 displays the privacy consent form of the Support-U application. In the depicted form each function that utilises personal context information, is listed. Furthermore, the user is able to activate or to deactivate the functions, e.g., to enable a live stream or to enable context state prediction.



Consent to the collection, processing,
use and disclosure of personal data

1) Type, scope and purpose of the collection, processing and use of personal data

The offered application „Support-U" will collect, process or utilize („use") your personal data. This is always done in compliance with the relevant German and European data protection law.

2) Utilization of offered services

We will collect the following personal data from you in order to provide you the services of „Support-U":
- Vital data: blood pressure and pulse
- Brightness, temperature and humidity in your flat
- Opening/Closing of windows and doors
- Video transfer of the flat to strengthen trust: O
- Usage of electric appliances (refridgerator, stove etc.)
- Person movement recognition (presence, seating furniture)

Within this application, you also have the possibility to enter personal information (e. g. name). These are voluntary disclosures. Only data, which are necessary for the application and its personalization, are requested. This data is used to execute the application, but only in cases permitted by law.

The collected data will be used for the following purposes:
- Person and flat status presentation
- Dangerous situations recognition
- Movement recognition (standing, sitting, moving)
- Visitor recognition
- Context prediction for early detection of future events: I

I know the purposes for which the data collected here are used and I hereby consent to the use of data for these purposes (I can withdraw my consent at any time): I

Fig. 3. The privacy consent form enables the user to agree or disagree to particular functions of the Support-U application.

- Meet-U: Meet-U provides several functions that make use of localization mechanisms and personal data of the user. That includes the user's interests, buddy list and his preferred means of transportation. For indoor navigation a RFID sensor attached to the user is exploited. The user can switch off the navigation function at any time so that neither the indoor nor the outdoor localization continue to operate. The user's preferences concerning transportation will be no longer available, too.

CONSEQUENCES: By enabling the user to explicitly disagree to certain functions, an application might not be able to provide all of its functionalities to the user anymore. However, the usage of this pattern in the development process of user-aware applications will strengthen the user's confidence in the software.

RELATED REQUIREMENT PATTERNS:

- Configurability: The application shall enable users to activate or deactivate functions.
- Agreement to Functionality: The application shall ask for users' consent regarding the functionality before use. The application should enable users to alter their consent regarding the functionality.

RELATED HCI PATTERNS:

- Choice from a Small Set: "Show all the possible choices up front, show clearly which choice(s) have been made, and indicate unequivocally whether one or several values can be chosen." [4]

V. DISCUSSION

Design patterns are a subject of dispute, both in software engineering and architecture, where they originated from. Christopher Alexander, who is also called the father of pattern languages, introduced a comprehensive pattern language that encompassed architectural design patterns of different granularity [1]. The patterns are linked to each other and can be composed in a hierarchical manner such that they address larger structures as well as their fine-grained details. However, after the application of his pattern language by several architects, he was disappointed about the results and criticised in later works [31], [32], [2] the pure mechanical application of patterns leading to buildings that have "this mechanical death-like morphology" [31].

The interdisciplinary design patterns proposed in our work are not designed to skip any stages in the development process, and our intention is not to provide building blocks or components that can be composed in a hierarchical manner. Our pattern approach emphasizes the need for a combined, interwoven treatment of design aspects from different disciplines. Our patterns aim at giving hints and pointing to expectations and restrictions that the different disciplines should bear in mind when collaborating and striving for a common goal, i.e. the societal embedding of software applications.

Our VENUS development methodology facilitates the engineering of software solutions for user-aware applications by involving the main dimensions that address societal aspects, such as norms and principles, and that reflect the requests and doubts of potential users. The development methodology is supported by a collection of interdisciplinary design patterns that indicate potential starting points, keep important aspects and point of frictions in mind and speed up the development by presenting related solutions and best-practice know-how. To say it with the words of Chung et al. [12], patterns represent solutions that others have already thought through. They conclude after evaluating their patterns for ubiquitous computing that not all of them were valuable, but that they helped in generating new ideas, in communicating them and in speeding up the developers work. Certainly, it takes time for a pattern to mature and prove its value.

VI. Conclusion

User-aware applications regularly make use of sensitive personal data such as the user's location, preferences, contacts or health status in order to reason about the situation and react autonomously. Aspects like privacy, trust and usability have a tremendous impact on the acceptance and acceptability of such applications. These aspects should be included and considered throughout the entire development process. We propose a set of interdisciplinary design patterns that focus on the necessary societal embedding of user-aware applications. They were derived from three different case studies where usability-enhancing and trust and confidence-building measures were applied and compliance with laws was taken into account. They go beyond earlier pattern languages that have concentrated mainly on functional and security-related concerns or looked solely at usability issues. The interdisciplinary nature of the patterns makes development teams aware and reminds them of requirements and concerns from other disciplines even if the respective discipline experts are not present in the team. Thus, conflicting disciplinary requirements can be resolved early in the development process. The patterns support the re-use of important design know-how in order to reduce the likelihood of repeating mistakes, speed up the development process, and lower the design effort. Last but not least, the patterns facilitate the discussions among the discipline experts by creating a common conceptual foundation. We are confident that these benefits will be confirmed in future development projects that make use of our pattern collection.

A further intention of this paper is to raise the awareness for the need for a socially aware design of software. We believe that our evolving set of reusable interdisciplinary design patterns facilitates substantially the development of applications that are well embedded into the societal environment. However, we are well aware that our pool of experiences is not large enough and that the target problem space is so huge and multifaceted such that much more experience is needed until a final word on such a pattern collection can be spoken. We hope that other research and development teams will share their experiences with the community and will contribute to establishing solid guidelines and best practices for the development of socially aware computing applications.

Acknowledgment

Funded by the federal state of Hesse within its LOEWE program for promoting cutting-edge research.

References

[1] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language: Towns, Buildings, Construction*, ser. Center for Environmental Structure Berkeley, Calif: Center for Environmental Structure series. OUP USA, 1977.

[2] R. P. Gabriel, *Patterns of software*. Oxford University Press New York, 1996, vol. 62.

[3] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[4] J. Tidwell, "A pattern language for human-computer interface design," *Washington University Tech. Report WUCS-98-25*, 1998. [Online]. Available: http://www.mit.edu/~jtidwell/common_ground.html

[5] K. Geihs, S. Niemczyk, A. Roßnagel, and A. Witsch, "On the socially aware development of self-adaptive ubiquitous computing applications," *it - Information Technology*, vol. 56, no. 1, pp. 1–41, 2014.

[6] H. Baraki, K. Geihs, A. Hoffmann, C. Voigtmann, R. Kniewel, B.-E. Macek, and J. Zirfas, "Towards interdisciplinary design patterns for ubiquitous computing aplications," Zentrum für Informationstechnik-Gestaltung (ITeG), Tech. Rep., 2014. [Online]. Available: http://www.uni-kassel.de/upress/online/OpenAccess/978-3-86219-557-2.OpenAccess.pdf

[7] M. Langheinrich, "Privacy by designprinciples of privacy-aware ubiquitous systems," in *Ubicomp 2001: Ubiquitous Computing*. Springer, 2001, pp. 273–291.

[8] S. Lahlou and F. Jegou, "European disappearing computer privacy design guidelines, version 1.1," 2004, ambient Agora-s IST 2000-25134.

[9] OECD, O. for Economic Co-operation, and Development, *Privacy Online: OECD Guidance on Policy and Practice*. OECD Publishing, 2003. [Online]. Available: http://books.google.de/books?id=GlXGYFJ6ANgC

[10] C. Crumlish and E. Malone, *Designing social interfaces: Principles, patterns, and practices for improving the user experience*. " O'Reilly Media, Inc.", 2009.

[11] J. Tidwell, *Designing User Interfaces*. OReilly, 2006.

[12] E. S. Chung, J. I. Hong, J. Lin, M. K. Prabaker, J. A. Landay, and A. L. Liu, "Development and evaluation of emerging design patterns for ubiquitous computing," in *Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*. ACM, 2004, pp. 233–242.

[13] T. Ruiz-López, M. Noguera, M. J. R. Fórtiz, and J. L. Garrido, "Requirements systematization through pattern application in ubiquitous systems," in *Ambient Intelligence-Software and Applications*. Springer, 2013, pp. 17–24.

[14] K. David, K. Geihs, J. M. Leimeister, A. Roßnagel, L. Schmidt, G. Stumme, and A. Wacker, *Socio-technical Design of Ubiquitous Computing Systems*. Springer, 2014.

[15] D. E. Comes, C. Evers, K. Geihs, A. Hoffmann, R. Kniewel, J. M. Leimeister, S. Niemczyk, A. Roßnagel, L. Schmidt, T. Schulz, M. Söllner, and A. Witsch, "Designing socio-technical applications for ubiquitous computing," in *Distributed Applications and Interoperable Systems*, ser. Lecture Notes in Computer Science, K. M. Göschka and S. Haridi, Eds. Springer Berlin Heidelberg, 2012, vol. 7272, pp. 194–201.

[16] M. Atzmueller, B. E. Macek, A. Hoffmann, M. Kibanov, C. Scholz, M. Söllner, and G. Stumme, *Connect-U Development of Ubiquitous Systems for Enhancing Social Networking*. Springer, 2014, pp. 261–274.

[17] S. Hoberg, L. Schmidt, A. Hoffmann, M. Söllner, J. M. Leimeister, C. Voigtmann, K. David, J. Zirfas, and A. Roßnagel, "Socially acceptable design of a ubiquitous system for monitoring elderly family members," in *Sozio-technisches Systemdesign im Zeitalter des Ubiquitous Computing (SUBICO)*, ser. Lecture Notes in Informatics (LNI), U. Goltz, M. Magnor, H.-J. Appelrath, H. K. Matthies, W.-T. Balke, and L. Wolf, Eds., vol. P-208, Bonn, 2012, pp. 349–364.

[18] I. O. for Standardization, *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability*. ANSI, 1998.

[19] M. Atzmüller, H. Baraki, K. Behrenbruch, D. Comes, C. Evers, A. Hoffmann, H. Hoffmann, S. Jandt, M. Kibanov, O. Kieselmann, R. Kniewel, I. König, B. Macek, S. Niemczyk, C. Scholz, M. Schuldt, T. Schulz, H. Skistims, M. Söllner, C. Voigtmann, A. Witsch, and J. Zirfas, "Die venus-entwicklungsmethode eine interdisziplinäre methode für soziotechnische softwaregestaltung," Zentrum für Informationstechnik-Gestaltung (ITeG), Tech. Rep., 2014. [Online]. Available: http://www.uni-kassel.de/upress/online/OpenAccess/978-3-86219-550-3.OpenAccess.pdf

[20] X. Franch, C. Palomares, C. Quer, S. Renault, and F. Lazzer, "A metamodel for software requirement patterns," in *Requirements Engineering: Foundation for Software Quality*, ser. Lecture Notes in Computer Science, R. Wieringa and A. Persson, Eds. Springer Berlin Heidelberg, 2010, vol. 6182, pp. 85–90.

[21] S. Robertson and J. Robertson, *Mastering the requirements process*. Boston: Addison-Wesley Professional, 2006.

[22] A. D. Toro, B. B. Jiménez, A. R. Cortés, and M. T. Bonilla, "A requirements elicitation approach based in templates and patterns," in *Workshop em Engenharia de Requisitos*, 1999, pp. 17–29.

[23] A. Hoffmann, M. Söllner, and H. Hoffmann, "Twenty software requirement patterns to specify recommender systems that users will trust," in *20th European Conference on Information Systems (ECIS)*, 2012, pp. 185–198.

[24] S. Hoober and E. Berkman, *Designing mobile interfaces*. O'Reilly Media, Inc., 2011.

[25] R. Andrich, V. Gower, A. Caracciolo, G. D. Zanna, and M. D. Rienzo, "The dat project: A smart home environment for people with disabilities." in *ICCHP*, ser. Lecture Notes in Computer Science, K. Miesenberger, J. Klaus, W. L. Zagler, and A. I. Karshmer, Eds., vol. 4061. Springer, 2006.

[26] M. Danninger and R. Stiefelhagen, "A context-aware virtual secretary in a smart office environment," in *MM '08: Proceeding of the 16th ACM international conference on Multimedia*. New York, NY, USA: ACM, 2008, pp. 529–538.

[27] J. A. Paradiso, J. Gips, M. Laibowitz, S. Sadi, D. Merrill, R. Aylward, P. Maes, and A. Pentland, "Identifying and facilitating social interaction with a wearable wireless sensor network," *Personal and Ubiquitous Computing*, vol. 14, no. 2, pp. 137–152, February 2010.

[28] M. Atzmüller, D. Benz, S. Doerfel, A. Hotho, R. Jäschke, B. E. Macek, F. Mitzlaff, C. Scholz, and G. Stumme, "Enhancing social interactions at conferences," *it - Information Technology*, vol. 53, no. 3, pp. 101–107, May 2011. [Online]. Available: http://dx.doi.org/10.1524/itit.2011.0631

[29] C. Kuner, *European Data Protection Law, Corporate Compliance and Regulation*. Oxford University Press, 2007.

[30] G. Hornung and C. Schnabel, "Data protection in germany: The population census decision and the right to informational self-determination," *Computer Law & Security Review*, vol. 25, no. 1, pp. 84–88, 2009.

[31] S. Grabow, *Christopher Alexander: the search for a new paradigm in architecture*. Oriel Press Stocksfield, 1983.

[32] C. Alexander, *The nature of order: an essay on the art of building and the nature of the universe*. Taylor & Francis, 2002.