Technical Reports 2

Zentrum für Informationstechnik-Gestaltung
ITeG



Harun Baraki, Kurt Geihs, Axel Hoffmann,
Christian Voigtmann, Romy Kniewel,
Björn-Elmar Macek, Julia Zirfas

Towards Interdisciplinary Design Patterns for Ubiquitous Computing Applications

kassel university press

ITeG Technical Reports

Band 2

Herausgegeben vom
Zentrum für Informationstechnik-Gestaltung (ITeG)
an der Universität Kassel

Zentrum für Informationstechnik-Gestaltung
ITeG

Universität Kassel
ITeG Zentrum für
Informationstechnik-Gestaltung
Pfannkuchstraße 1
D-34121 Kassel

Towards Interdisciplinary Design Patterns for Ubiquitous Computing Applications

Harun Baraki, Kurt Geihs, Axel Hoffmann, Christian Voigtmann, Romy Kniewel, Björn-Elmar Macek, and Julia Zirfas

kassel university press

Gestaltung technisch-sozialer Vernetzung in situativen ubiquitären Systemen (VENUS)

Gefördert durch:

Exzellente Forschung für Hessens Zukunft

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.dnb.de abrufbar

ISBN: 978-3-86219-557-2
URN: URN: http://nbn-resolving.de/urn:nbn:de:0002-35572

www.upress.uni-kassel.de

Vorwort

Der vorliegende Band ist ein zweiter Technischer Bericht, der aus dem Forschungsschwerpunkt LOEWE-VENUS hervorgegangen ist. Aufbauend auf Band 1 der ITeG Technical Reports, der die VENUS-Entwicklungsmethode als kompletten Entwicklungszyklus vorstellt, wird hier herausgearbeitet, inwieweit Ergebnisse aus der Entwicklung dieser neuen interdisziplinären Gestaltungsmethodik bereits als Vorgaben in Form von Mustern standardisiert und wiederverwendet werden können. Durch diese Muster sozialverträglicher Informationstechnikgestaltung soll der Aufwand in Entwicklungsprozessen in der Praxis erheblich reduziert werden.

Wie in ITeG-TR Band 1 beschrieben, ist das Ziel der interdisziplinären VENUS-Entwicklungsmethode, Software für Ubiquitous Computing (UC)-Systeme systematisch sozialverträglich zu entwerfen. Dies erfordert die Einbeziehung von Experten aus unterschiedlichen Disziplinen. Um die praktische Anwendbarkeit der VENUS-Entwicklungsmethode zu verbessern, hat sich eine Arbeitsgruppe auf die Entwicklung interdisziplinärer Entwurfsmuster für UC-Anwendungen konzentriert. Anhand der drei in VENUS entwickelten Demonstratoren Meet-U, Connect-U und Support-U wurden konkrete interdisziplinäre Software-Design-Pattern für Anwendungen in ubiquitären Umgebungen herausgearbeitet und getestet. Die Muster repräsentieren wiederverwendbares Best-Practice-Wissen und ermöglichen es zukünftigen Entwicklern - wenn diese vor der Herausforderung stehen, vertrauenswürdige, akzeptable und rechtsverträgliche ubiquitäre Systeme zu gestalten – systematisch während der Entwicklung der Software auf technische und sicherheitstechnische sowie gleichzeitig auf rechtliche Aspekte und die Problematik von Akzeptanz und Gebrauchstauglichkeit einzugehen.

Das Besondere für die hier vorgeschlagenen Entwurfsmuster ist, dass sie keine einfachen Musterlösungen für rein technische Probleme bieten, sondern ganz bewusst interdisziplinäre Aspekte integrieren, denn gesucht wurden Lösungen, die sich auf die „Programmierung“ einer nachhaltigen Beziehung Mensch-Technik richten und auf einem ganzheitlichen Entwicklungsansatz fußen. Ausgehend von generellen Anforderungen wurden konkrete Gestaltungslösungen entwickelt und dabei für die Akzeptanz der Resultate wichtige nicht-funktionale Perspektiven (Benutzbarkeit, Recht, Vertrauen) einbezogen. Bereits seit den 1990er Jahren haben Forscher versucht, dieser gesellschaftlichen Dimension von UC beizukommen und Prinzipien für die Entwicklung menschgerechter IT aufzustellen. Im LOEWE-VENUS –Projekt ist es nun gelungen, diese Prinzipien in konkreten Szenarien und prototypischen Beispielanwendungen von einer abstrakten Ebene bis hin zu handhabbaren musterhaften Gestaltungslösungen fortzuentwickeln.

In diesem Sinne hoffen wir, hier Best-Practice-Lösungen bieten zu können, die zukünftige Entwickler ubiquitärer Systeme inspirieren und die helfen, Unzulänglichkeiten und Fehler schon während des Entwicklungsprozesses zu vermeiden. Entwurfsmuster sind Vorschläge, die sich in der Praxis bewähren müssen. Wir sehen der weiteren Erprobung der Muster mit Spannung entgegen und freuen uns über jedes Feedback hierzu.

Prof. Dr. Kurt Geihs

(Sprecher von LOEWE-VENUS)

Towards Interdisciplinary Design Patterns for Ubiquitous Computing Applications

Technical Report

Harun Baraki, Kurt Geihs, Axel Hoffmann, Christian Voigtmann, Romy Kniewel, Björn-Elmar Macek, and Julia Zirfas

University of Kassel
Zentrum für Informationstechnikgestaltung (ITeG)
Pfannkuchstraße 1, 34121 Kassel, Germany

Abstract. To draw on the full potential of Ubiquitous Computing (UC) systems, they have to be designed with awareness for their social embedding in order to increase the user acceptance. To this end, not only the compliance with laws has to be ensured, but also usability-enhancing and trust and confidence-building measures have to be applied. This makes the development of UC applications a challenging task that involves experts from different disciplines. The main contribution of this report is a set of design patterns for UC applications that specifically focus on the interweaving and implementation of multidisciplinary requirements. The patterns capture the design know-how of typical, recurring features in context-aware adaptive UC applications with particular concern for the sociotechnical requirements. First, we present a detailed discussion of the related work on design patterns in the realm of UC. Afterwards, we explain our research methodology and the template structure of our pattern specifications. The core of the report consists of eight interdisciplinary UC design patterns. This initial list of patterns was derived from several application case studies in the interdisciplinary research project LOEWE-VENUS. We view this collection as a starting point for an evolving set of commonly accepted reusable design patterns that facilitate the development of accepted and acceptable UC applications.

1 Introduction

Design patterns are about not re-inventing the wheel. They are meant to transfer knowledge about how others viewed and solved recurring problems in engineering tasks, and thus they should help to avoid potential mistakes. Another important role of design patterns is providing a common vocabulary that lets experts communicate more easily about design questions. Design patterns emerged from the work of Alexander et al. in the field of architecture [1]. Today design patterns are known in many domains (e.g., [2,3,4,5,6,7]). In software engineering, a design pattern is an abstract, generally reusable solution to a commonly occurring problem in the design of software systems. A design pattern is a template that

can be used in many different applications. Thus, design patterns are formalized best practices that ease the job of the software developer [8].

This report is about interdisciplinary design patterns for ubiquitous computing applications. In a nutshell, ubiquitous computing (UC) is a computing concept where computing is everywhere around us while the computing devices are made effectively invisible. The recent increase of smartphone and tablet computer usage has resulted in technology that is increasingly "interwoven into the fabric of everyday life" [9]. The multitude of pervasive applications running on these devices are those which provide attractive services for our private as well as professional needs. Key characteristics of ubiquitous systems are context awareness and dynamic adaptation to context changes. These features require monitoring the user's environment and preferences in order to provide context-dependent information and make appropriate adaptation decisions. 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, social aspects during system development. Questions of privacy, trust, usability, legal compliance etc. have to be addressed [10]. Thus, user acceptance and general acceptability of the new technology must be overarching concerns in the development process.

The presented design patterns have emerged from the interdisciplinary research project VENUS. The overall goal of VENUS has been the definition and evaluation of a comprehensive interdisciplinary development method for the design of socially aware ubiquitous computing systems [11]. The project has focused particularly on design support for the social embedding of ubiquitous computing technology, i.e. understanding the interactions between technology, individual user and society, and translating abstract norms and rules into concrete technical requirements and design artifacts. Therefore, four disciplines are represented in VENUS, i.e. computer science, trust research, ergonomics and law, contributing to the research of development methods and tools for ubiquitous applications and taking into account theories, methods and tools described in the context of socio-technical system design.

In the course of the VENUS project three demonstrators have been designed and implemented by separate development teams in different application domains, i.e. mobile computing, social networking, and ambient assisted living. The VENUS development method helped them in realizing the applications and supported the different disciplines to work together in a structured way. Nonetheless, it remains a time-consuming and complex task to develop socially compatible UC applications. For example, domain experts from different disciplines such as law, ergonomics and trust research, have to find out first the normative requirements that are relevant for the intended application. Then, the normative requirements have to be translated to technical design goals. Besides that, the requirements can conflict with each other and, hence, have to be coordinated and balanced. Although the VENUS development method assists throughout the whole development process, it can be quite costly to create so-

cially acceptable UC applications. We studied related work to find supportive design patterns that address different disciplinary concerns and that give advice about their implementation. However, most works consider either privacy or usability issues, but not both at the same time [4,12,13]. In addition, none of them covers the same spectrum of disciplines as VENUS does. To facilitate and speed up the development of socially acceptable UC applications, we analyzed the demonstrators for recurrent design questions and solutions that were related to interdisciplinary concerns about how to achieve certain aspects of the social embedding of the applications. In an iterative process we consolidated the discovered candidate design patterns and created the collection of patterns that is presented in this report.

Section 2 of this report discusses the related work. Section 3 explains the structure of our design patterns. In Section 4 we present our design patterns that focus exclusively on interdisciplinary, cross-cutting design aspects. Section 5 concludes the report with a general reflection on the significance of our design patterns and an outlook to future work.

2 Related Work

Interdisciplinary design patterns for ubiquitous computing applications is a research field that has received limited attention so far. We believe that this is due to the fact that the technology itself is still under development and constantly evolving and that the socio-technical implications of using this new technology are becoming apparent and understood only recently with the actual deployment and usage of ubiquitous computing technologies.

There are pattern-related publications that focus on specific technical features of UC such as enabling application adaptivity or context-awareness, but do not consider cross-cutting aspects like transparency, trust, privacy and informational self-determination. For example, concerning adaptivity the authors of [14] analyze 30 open-source and research projects and identify 12 patterns in these projects. Their patterns support the software developer in designing and implementing features for monitoring, adaptation decision-making and reconfiguration in order to realize adaptivity. Likewise, the authors of [7] describe four patterns that are helpful for certain adaptation problems in context-aware applications. However, non-functional requirements, that can restrict the appliance of such patterns, are not taken into account.

Other works deal in fact with non-functional requirements and how they can be translated to design artifacts through patterns. Since the term non-functional requirements is a wide ranging notion, a distinction must be made between requirements that address more conventional quality of software attributes (e.g. performance, reusability, maintainability), and requirements that address concerns cross-cutting through various disciplines and stimulate considerations related to law, trust and usability. An example of the former can be found in [15].

Our main interest is on the latter kind of requirements. Thus, in the following we focus primarily on related work that adopts a clear interdisciplinary viewpoint.

Langheinrich proposes in [13], which is one of the first publications looking at privacy and ubiquitous computing together, a set of guidelines to design privacy-aware ubiquitous computing systems. He claims that the most fundamental principle is the principle of *openness*, which he also calls the principle of *notice*. The user has to be informed by announcements first and foremost if data is collected about him, particularly what type of data is collected. For example, if the user enters a room equipped with microphones and cameras, he has to be notified about audio and video recordings. Further information like the purpose of use, the recipients of the data and the data retention period can be listed in the announcement, too. The notice principle is inspired by the Platform for Privacy Preferences project (P3P) [16] where a web site can prescribe in a machine readable form which data will be collected, whether it will be anonymized, and for what purpose and for how long the data will be retained. Langheinrich considers the notice principle as the bottom-line of any privacy-aware UC system.

In P3P the user may also configure, e.g. in his browser, which web site policies should be accepted or declined automatically. For this purpose, Langheinrich introduces the *choice and consent* principle which he considers as appertaining to the notice principle. The user has to consent explicitly to the use and collection of personal data. Both the notice and the choice and consent principles were justified on the basis of the EU Data Protection Directive 95/46/EC (It should be noted that the Directive will be replaced soon by the General Data Protection Regulation (GDRP)). As an alternative to the aforementioned guidelines the author proposes to use anonymity in case that only certain types of data are gathered. As a result, user consent is not required anymore, because anonymous data is no longer considered as personal data. If authentication or some kind of personalization is needed, then pseudonymity could be a solution. A user may change the pseudonym at any time to be not traceable. Nonetheless, the author points out that pseudonyms might be linkable under certain circumstances and warns against data-mining applications which could assemble such data into a single coherent picture. He recognizes that the implementation of this guideline might be too difficult and presents to this end the principle of *proximity and locality* and the principle of *access and recourse*. The principle of proximity and locality can be considered as a practical solution to the notice and consent problem in certain situations. The user's consent is assumed as long as he is in proximity to the corresponding device. For instance, recording and playing back of the user's voice is only allowed if the user's presence is detected. The principle of locality implies that data is not dissiminated indefinitely, but stays, for example, within a geographic boundary or local network. By applying the principle of locality the unauthorized use of personal data, for instance for surveillance purposes, may be impeded. With the principle of access and recourse the author addresses the implementation of specific legal requirements such as collection and use limitation and mechanisms for non-repudiation in case of legal disputes. For instance, a protocol derived from the P3P protocol can be augmented with

digital signatures, and special privacy-aware storage technologies can enable the use of personal data in compliance with the declared privacy practices. In this connection the data collector should stick to the fair information practice principles (FIPP) that were first described in the Privacy Act of 1974 (5 U.S.C. 552a as amended). These require primarily that the data collector gathers only relevant data for the well-defined purpose and keeps it only as long as necessary for the purpose. Furthermore, the user has to be able to find out what information is recorded about him and how it is used. This is also known as individual participation. For that reason, Langheinrich proposes usage logs similar to call-lists of phone bills [17].

In summary, Langheinrich's work is oriented towards the fair information practice principles and considers them from the perspective of ubiquitous computing. The suggested solutions were formulated in a general manner. As the title of the work implies, he proposes principles, but not design patterns that support their implementation. In addition, at that time (2001) no smartphones were available which nowadays provide not only new opportunities but also new challenges. Nevertheless, his work can be considered as an excellent introduction to the discussions about UC patterns.

In [12] Lahlou et al. propose nine guidelines, called European Disappearing Computer Privacy Design Guidelines. The first two, i.e. the *Openness* and the *Privacy Razor* guideline, are geared to the principles introduced by the OECD [18]. They are similar to the openness, the data quality and the use limitation principles of the OECD. The *Third Party Guarantee* guideline advises the developer to let the user choose whether third parties may be included or not. The *Consider Time* guideline requires that data should have set an expiry date as default setting. When applying the *Make Risky Operations Expensive guideline*, privacy-sensitive operations are made costly for everyone, including the system, the user and third parties. Systematic costs may be small time delays, small amounts of money or obligations of tracing records. According to the authors, this should discourage potential abuse. The *Avoid Surprise guideline* recommends notifying the user if a user's action influences the system or if significant state changes are about to take place. Regarding this, the authors propose to find a trade-off between cognitive overflow and awareness, and hence to enable the customization of acknowledgements. Other guidelines proposed by the authors, 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. Altogether, Lahlou et al. indicate problems and challenges developers have to tackle. The nine guidelines are described very briefly and only a few provide reasonable solutions. Many of them can be considered indeed as motivation for design patterns, but not as intelligible instructions developers can apply.

Ruiz-López et al. propose in [19] 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. Their notion of

pseudonymity is not the same as in many other works. The *Pseudonymity* pattern describes a hierarchical structure where the top node contains all private data and the bottom node none. The nodes in-between have different amounts of information. The user can configure his preferences concerning the disclosure of data according to the trustworthiness of the situation and the context variables required. The authors do not go into detail and do not explain which information the intermediate nodes may or should retain or how the trustworthiness of a context can be estimated. The *Human Factor* pattern 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 their well-being, for instance. 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.

In [4] Chung et al. propose 45 design patterns for ubiquitous systems, grouped into the four groups *Ubiquitous Computing Genres, Physical-Virtual Spaces, Developing Successful Privacy*, and *Designing Fluid Interactions*. The patterns are called *pre-patterns* since the patterns are not yet in common use by the design community and end-users. Their intention is to provide an initial list of design patterns that can be enhanced and extended. In the context of this work we are particularly interested in the fifteen design patterns devoted to privacy and the eleven patterns related to fluid interactions. The technically oriented patterns will be omitted in the following sections.

Chung et al. number their patterns consecutively such that lower-numbered patterns are more abstract than higher-numbered ones. Hence, the first privacy pattern is the most abstract one and reflects mainly the fair information practices mentioned above. The second pattern is about trust between the sensed and the sensing and collecting party. To strengthen the user's trust the authors propose, for example, to respect the fair information practices, to involve potential users in the design process and to let people know what is sensed and what is not. Pattern number 3 is also emphasizing trust and credibility and distinguishes between the relationship of two individuals and that between an individual and a company or organization. For both of them the authors recommend the appliance of the fourth and the fifth pattern which prescribe how to allow users to control the sharing of their data and to enable them to check what information is visible and collected by whom. In case of direct exchange of data between two individuals the patterns do not have to be implemented as strictly as in case of data exchange with a company or organization. The sixth pattern shall support the implementation of the previous patterns by privacy-sensitive architectures. To this end, it refers to pattern number 15 which proposes to keep personal data on personal devices. Furthermore, pattern six advises to apply encryption or to use a trusted computing base. A further solution to realize privacy-sensitive architectures is to create physical privacy zones by deploying sensors only in appropriate places and to inform the users about these. The authors clarify this

in their eighth pattern. Pattern seven is called *Partial Identification* and deals with the fact that not every application requires the full identity of the user. Keys, tickets, passwords or sensor data like the weight or the size of a person may be sufficient for certain applications. In this connection, pattern nine can be incorporated too. It proposes to blur specific personal data by providing data at a coarser granularity or by aggregating multiple pieces of data. Pattern ten and eleven relate to pattern four and, thus, try to put the user in a position to control the sharing of their data. Pattern ten recommends limiting the access to personal data by identity access. Alternatively, spatial or temporal access control can be used in certain situations. For instance, the user's location may be only visible during working hours. The eleventh pattern makes an invisible mode available to the user so that the flow of data to others will be stopped. An instant messenger would indicate, for example, that the user is off-line. Pattern thirteen and fourteen are part of pattern five that supplies the user with privacy feedback, more precisely with feedback about what is monitored and accessed by whom. Consequently, pattern thirteen suggests options to inform a user appropriately. One is to notify the user immediately when an access occurs, the other is to give notice after the access. The authors recommend the former if the user is allowed to accept or deny a request. The fourteenth pattern, called *Privacy Mirrors*, displays whom, what and how the system is monitoring. In general, it should provide useful information without distracting the user too much. The last pattern in the privacy group aims at limited data retention. Collecting and storing too much data facilitates revealing patterns of behaviour and increases the risk of outdated personal data. A solution might be to keep a limited number of entries in the history or, if possible and reasonable, to store only the latest information that is relevant. For example, in case of website administration it is often sufficient to store the time of the last visit of the user and the total number of his visits.

The eleven patterns devoted to fluid interactions are arranged in the same manner as the privacy patterns. Thus, the first patterns are the more abstract ones and may be applied to a wider range. The first pattern is called *Scale of Interaction* and addresses different classes of devices and the type and complexity of interaction with them. According to this pattern, people will interact in the future with many small or embedded devices, several personalized devices and few personal computers. The interaction with small devices has to be extremely simple or nonexistent since users will not be willing to learn the usage of every single device. Personalized devices like PDAs or phones may require simple interactions and low to medium user complexity and attention as the interactions will be for a short period of time. The highest attention can be expected when the user is interacting with a personal computer. In this case, the complexity of interaction can range from low to high. The second pattern, which is named *Sensemaking of Services and Devices*, exposes the problem of service discovery in ubiquitous computing. The authors present therefore three ideas and their issues. The first idea is to make services available everywhere. However, this would make little sense to services with a local focus. Furthermore, scalability

problems could appear due to the large number of services. Alternatively, the services can be made discoverable through the network. For this purpose, the services should be described in a standard digital format so that, for example, a browser can be used to explore them. The last idea concerning this is to relieve the user by consuming the services as autonomously and maintenance free as possible. The subsequent pattern is the *Streamlining Repetitive Tasks Pattern*. It suggests to automate simple tasks like setting up a clock to the current time or switching to DVD input if the DVD player's Play button is hit. Nonetheless, errors and unexpected results may lead to users feeling helpless. So, the main prerequisite is that the task is highly predictable. Otherwise, the user must be enabled to automate his tasks. Further issues designers have to consider are exception handling and accountability, for instance, in the case of autonomous cars crashing into another. The authors do not go into detail but refer to the following *Active Teaching* and the *Serendipity in Exploration* pattern as alternatives. The *Active Teaching Pattern* proposes to show helpful hints to the user and to teach him subtly instead of automating a task. The patterns *Serendipity in Exploration* and *Keeping Users in Control* are both about keeping the user in control. The aforementioned active teaching could be one option, but in case the user is not demanding for guidance but for freedom to explore and to leave pre-determined courses, then the applications have to be designed controllable and flexible. One possibility is to allow user-created content. Another option, especially in the context of navigation systems, is to provide different paths. The *Context-Sensitive I/O Pattern* recommends that input and output mechanisms has to adapt according to the user's context. For instance, the silent mode of a smartphone has to be activated when the user enters a theatre. However, the user has to be kept in control such that he can override undesired adaptations. In this connection, the *Resolving Ambiguity Pattern* can be helpful, too. In general, it advises to display the user what the system knows and what it is going to perform. If the system's inference is an erroneous conclusion, the user is able to give appropriate feedback. In case of ambiguity in sensed data, the system has to ask the user unobtrusively to resolve the ambiguity. The *Ambient Displays Pattern* proposes to support the user to understand and process the flood of information without requiring his full attention. For example, the stock market's performance can be indicated by colors; green for positive and red for negative. The *Follow-me Displays Pattern* deals with moving the user's working settings, personal information and workspace to that computer the user is currently working with. Chung et al. mention that designers should pay in such scenarios particular attention to privacy issues since no-one wants to publish his data on another computer without his knowledge. The last pattern the authors present suggests a convenient way to share data across devices. It proposes to use a drag-and-drop operation between two devices. Accordingly, the pattern's name is *Pick and Drop*.

In contrast to our approach, this very comprehensive set of pre-patterns did not emerge from the authors own software projects but from studying the relevant related literature. For example, the *Fair Information Practices* pattern lists the

aforementioned practices of the 1970s as solution and refers to Langheinrich's work [13]. Other patterns reproduce principles like the choice and consent principle, but do not provide clear instructions how to put them into practice. According to the authors and their evaluation, the patterns related to privacy did not help designers significantly to handle these issues; they realized that their privacy patterns are too vague to be helpful at the design stage. Regarding the patterns for fluid interaction, the evaluation with eight designers using patterns and ten designers not applying them, showed that particularly novice designers benefit from these patterns. The designs of the participants were assessed by three HCI designers. In case of designers with high experience they found no improvements concerning the quality of the designs. But in general, the patterns helped in generating new ideas and communicating them. Furthermore, they supported the designers speeding up their work. According to the authors, this is due to the fact that patterns represent solutions that others have already thought through.

Landay et al. detected in their work [20] the following patterns for ubiquitous computing: Context-Sensitive I/O, Physical-Virtual Associations, Global Data, Proxies for Devices, Follow-Me Display, Appropriate Levels of Attention, and Anticipation. Only the patterns *Context-Sensitive I/O* and *Physical-Virtual Associations* are described in detail in their article. The former one is already mentioned above since Chung et al. also refer to the work of Landay et al. and their context-sensitive I/O. The latter pattern aims at sharing information between several devices. The authors propose to simplify connecting devices and sharing information over the life of a session if users are near one another. In particular, the time to configure the devices appropriately shall be reduced.
Landay et al. also wonder about validating and evaluating design patterns for ubiquitous computing. The rule they follow is that a pattern has to be found in at least three good implementation examples before it can be identified as a design pattern. We tried to be in compliance with this rule and to determine patterns that were present in all of our own three ubiquitous computing applications. Unfortunately, Landay et al. do not specify where their patterns were applied.

Other authors consider privacy enhancing technologies (PET) from a more general viewpoint. In [21] Hafiz proposes twelve patterns to support developing PET. The twelve patterns aim at enabling anonymity in various domains but do not bridge trust gaps nor do they inform users to enable them taking appropriate decisions. Besides, most of the patterns, such as Layered Encryption, Covered Traffic, Chaining, Anonymity Set and so forth, require a lot of additional computing resources and might be not suitable for resource-scarce ubiquitous computing devices. The authors concede that they describe design decisions to implement a mix-based system. In such systems messages are encoded and decoded and sent via several so-called mixes. Mixes are proxies that enable hard-to-trace communications.

Likewise, other works related to PET cannot be applied on ubiquitous systems without further considerations. Most of them focus on pure Internet applications or on traditional application areas. The characteristics of ubiquitous systems, especially their restrictions and their differentiating capabilities, necessitate adapted or new design patterns.

3 Pattern Definition

The interdisciplinary design patterns proposed in this work have emerged from the VENUS project. VENUS aimed at developing and evaluating an interdisciplinary method for designing socially aware ubiquitous computing systems. For evaluation purposes the three demonstrators Meet-U, Support-U and Connect-U were implemented by three separate teams. These demonstrators are also used in the pattern descriptions to show possible solutions and, therefore, will be described here briefly.

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. The 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 [22].

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 consumers, opened windows and doors and the current temperature, humidity and light intensity. The blood pressure, the pulse and the movement intensity of the person are transmitted by sensors attached to him or her. 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 information. In this way, on-the-spot checks by the caring person can be reduced and the elderly person may enjoy home or any other assisted living accomodations for a longer time instead of residing in a nursing home [23].

Connect-U is an application assisting 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 are wearing RFID tags that recognize whether two persons are communicating with each other. RFID readers in every room detect where the participants sojourn currently. On the Connect-U website the participants can check

which lectures and talks they are going to visit, which ones they attended and with whom they talked already in the conference. They can ask for more information about their dialogue partner by invoking his profile. If two participants accept each other as friends on Connect-U, they can even look up the other's current position [24].

All three demonstrators are UC applications that make use of sensor data and personal data and process them on servers or devices that do not belong to the person observed. Aspects like privacy, trust and usability have a considerably large impact on the design and implementation of UC applications and should be included and considered throughout the entire development process. The method created in VENUS supports interdisciplinary teams that include, inter alia, lawyers and trust and usability engineers, to work together in an efficient and structured manner [25].

The core of the VENUS approach is an iterative development consisting of the phases: analysis of needs, requirements management, conceptual design, software design with implementation and in-situ evaluation. In addition to the in-situ evaluation there are evaluations in the analysis (application scenarios) and the implementation phase (functions and prototypes). In the requirement management phase the approach considers, besides usual functional und quality requirements, requirements for legal compatibility, usability and trustworthiness. Experts derive these requirements from normative sources by concretizing the provision regarding the technical system [25].

We applied the VENUS development method [26] to the three demonstrator projects mentioned above and recognized that certain solutions were present in more than one demonstrator. Since the method encompasses requirements, design and implementation, we subdivided the recurrent structures into requirement patterns and design patterns. The former are not part of this work, but will be introduced briefly in one of the following chapters as they are referred to by the interdisciplinary design patterns. Furthermore, we will touch on HCI design patterns. They also assist developers to apply our design patterns. Before that, we present the structure of our design patterns.

3.1 Structure of the Interdisciplinary Design Patterns

In contrast to requirement patterns, which focus particularly on normative and functional requirements and have to be seen as a tool for requirement engineering, interdisciplinary design patterns deal with their technical realization. However, they should not be thought of as patterns targeted only at software developers. The design patterns are linked strongly to abstract normative requirements and criteria. Thus, the intention is also to support the development team to tackle and to concretize the abstract non-functional requirements and criteria they have defined for their application.
We will describe the design patterns by using a template with eight sections.

These include, inter alia, typical pattern fields such as intent, motivation, solution, forces and context, and consequences 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 concrete approaches for design and implementation, while the other parts, particularly the forces and context section, emphasize the abstract layers. These may encompass laws, norms, normative goals and further 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 can be solved. This section supports users in browsing the patterns efficiently. In total, the Intent section 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*: Mostly patterns only refer to other related patterns in the same pattern collection or to patterns dedicated to the same

subject and domain. Our patterns may also point to requirement patterns they can be associated with. They support developers in the requirements management phase.

– *Related HCI Patterns*: The HCI patterns we link to shall support developers in applying the design pattern. The supportive HCI patterns listed were not developed in the VENUS project. We will refer to the collections and works where they are explained and originated from.

What is special about the documentation is not only that the patterns' users are lead from an abstract layer to a concrete solution, but also that it looks at and justifies a solution from a variety of different non-functional perspectives, including trust, usability and legal policies. That means that a holistic approach is adopted for common problems in UC.

Table 1. Example of a trust-based requirement pattern (excerpt)

Requirement Pattern: Signalling the Function Status			
Metadata	Goal	The user is able to recognize which functions the system executes right now, and knows why something happens and how it happens.	
	Antecedent	Transparency	
	Relations	Provide Comprehensible Functionality, Information of Functions	
Template: Display of Execution	Standardized Requirement	The system shall display running tasks to the users.	
	Extension	The system shall display $< tasks >$ to the users.	
		Parameter	Values
		$< tasks >$	Running tasks, Background tasks, User tasks

3.2 Requirement Patterns

Requirement patterns are an approach to reuse recurring requirements [27]. It helps requirement analysts to identify and document software requirements [28]. The use of patterns for defining requirements is quite a new development. They are applied for eliciting and analysing requirements. Requirement patterns are a collection of knowledge and experience, which can be reused in current projects by adaptation [29]. According to the example in table 1 requirement patterns contain templates to describe a standardized requirement and other relevant information in tabular form [30]. These are, for example, the goal of the standardized requirement and relations to other patterns. To ease adaptation, attributes can be defined with usable content. However, only content that already has been elaborated and tested carefully should be predefined. Here, the principle of reuse is applied, because already created and successfully applied requirements are reused.

Table 2. Requirement Patterns and Related Design Patterns

Requirement Pattern	Design Pattern
Information About Functions	On Demand Explanation
Explanation of Processes	Abridged Terms and Conditions
Signaling the Function Status	Trust and Transparency
	Control of Autonomous Adaptation
Level of Automation of Functions	Control of Autonomous Adaptation
Control of Processes	Control of Autonomous Adaptation
	Emergency Button
Agreement to Functionality	Emergency Button
	Enable/Disable Functions
Configurability	Enable/Disable Functions
Assessment of the Output	Context State Indication
Logging Processes	Data Access Log

In accordance with the VENUS development method, the requirement patterns address socio-technical enablers [31]. These enablers are requirements regarding legal-compatibility, usability and trustworthiness. To derive these requirements the aforementioned VENUS method, which incorporates experts from the respective domains and provides them techniques to derive requirements from normative provisions, was applied [10]. We documented recurring requirements as requirement patterns [32,33,34]. We developed trust-based requirement patterns that aim at increasing the users' trust in the application before the system is used the first time [32]. That comprises of aspects like the provider's and application's reputation, the privacy statement of the provider and guarantees that third parties are providing. Furthermore, we developed requirement patterns which were targeted at increasing the user's trust in the application while using it [33]. Additionally, we also developed requirement patterns based on recurring requirements from legal-compatibility [34]. Table 2 shows some of the requirement patterns and their associated interdisciplinary design patterns.

3.3 HCI Patterns

Human-Computer Interaction (HCI) design patterns facilitate the development of user interfaces. The main goal of these patterns is to support designers in developing usable and ergonomic user interfaces [35]. Usability is a demanded quality of user interfaces. It can be defined as the "extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use" [36]. An HCI design pattern describes a recurring user interface design problem together with a proven solution.

According to van Welie et al. [37] an HCI design pattern should consist at least of the main elements problem, context, solution and examples. Furthermore, HCI design patterns should be focused on the user. For example, the problem

element describes a usability problem of the system in use. It is related to the usage of the system and relevant to the user. It should be user task oriented. However, the form or structure of HCI patterns varies a lot according to the authors' preferences [38]. Several additional elements might be used for a better understanding of the design idea. An HCI design pattern can be part of a collection or a pattern language. If so, a pattern may have references to other patterns in that collection or language.

Similar to patterns for software engineering, HCI design patterns originated from Alexander's concept of design patterns and pattern languages. Kruschitz et al. [38] date the first appearance of HCI design patterns in the year 1996 when Coram et al. [39] published the first design patterns of a pattern language for user interface designers to support them in building graphical user interfaces which are pleasurable and productive to use. Since then several other pattern collections and pattern languages have been published. The first HCI pattern collections were compiled by Tidwell [40] [38].

Today, there exists a vast amount of patterns written by many different authors, published on Web repositories [35], in scientific papers, and books [41,42,43]. Moreover, there are patterns written for several domains, such as desktop applications, websites [44] and recently mobile applications [45]. Our interdisciplinary design patterns will mainly refer to the HCI design patterns of Tidwell [40,43] and Hoober et al. [45].

4 Interdisciplinary Design Patterns

Each subsection of this chapter describes a particular design pattern. Since the area of UC is very broad and because the number of projects we examined is still rather small, this initial collection is bound to evolve. We will extend it when working on further UC projects and we hope that other development teams share their experiences about these patterns with us or probably generate new patterns. The overall goal is to provide a comprehensive list that does not only speed up and simplify the interdisciplinary development process, but also supports developers who cannot afford a full team of different domain experts, to create proper socio-technical systems.

4.1 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 UC application. This choice should 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. Another example are the acceleration sensors included in smartphones. A UC system can recommend 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 context data needed by the required 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 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 [46] and [47].
- Trust: 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.
- Transparency: The pattern provides transparency to the user by giving an overview, which function needs which personal context data of a user to work properly. For this reason a user is aware of the context data that is gathered by the sensors that surround her.

Solution A solution is given if the user can explicitly agree or disagree to certain functions. For this purpose, the UC system has to display every function and its required context data. A possible way of displaying these functions and the used context data may be the use of the privacy consent form, which is included in every application.

- Support-U: Figure 1 displays the privacy consent of the Support-U application. In the shown privacy consent form each function, which 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 predicting her next context.
- Meet-U: Meet-U provides several functions that make use of localization mechanisms and the personal data the user supplies. 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 now switch off the navigation function so that neither the indoor nor the outdoor localization continue to operate. The user's preferences concerning transportation will be no longer available. Further functions can be disabled

correspondingly. Turning off, for example, the advanced search engine would stop using the user's interests.

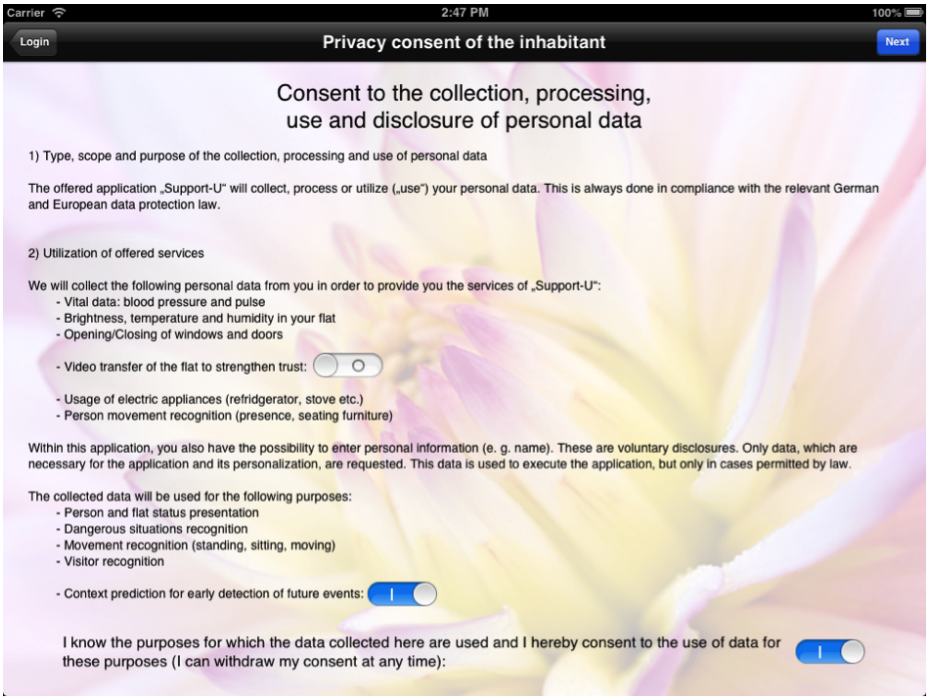

Fig. 1. The privacy consent form enables the user to agree or disagree to certain functions of the Support-U application.

Consequences By enabling the user to explicitly agree or disagree to certain functions, a context aware application like Support-U might not be able to provide all of its possible functionalities to the user anymore. However, the usage of this pattern in the development process of context-aware applications might additionally strengthen the user's confidence in the usage of UC systems.

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." [40]

4.2 Trust and Transparency

Intent The intent of the Trust and Transparency Pattern is to visualize hardware sensors or even inferred context information that surrounds a user, and that are used by a UC system. Furthermore, 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 user contexts are provided to the user without visualizing the sensors used by the UC system. For example, in the field of health care, possibilities were elaborated to give patients the opportunity to be monitored even if they are outside of a hospital using ubiquitous sensors built into smartphones [48]. So-called smart homes and smart rooms adapt their services to the lifestyle habits of occupants and the working routines of clerks by observing and learning their behaviour patterns [49,50]. The automotive application domain represents another area which is strongly influenced by ubiquitous sensors. Sensors such as infrared, radar, laser or GPS sensors may help to prevent possible collisions between a car and a pedestrian [51]. 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 [6]. Furthermore, RFID sensors can be used to detect whether conference attendees are talking to each other, how long their conversation took, and which talks participants have visited. Based on that, helpful information about other interesting talks at the conference can be provided to the user. Including additionally the user profiles of other attendees, those with similar interests can be displayed to a participant [3].

Forces and Context

– Trust: The pattern increases a user's trust in using UC systems by visualizing the context sources, respectively the sensors that surround the user.
– Transparency: The pattern provides transparency from a legal perspective, not from a technical perspective. Hence, the user possibly gets a better understanding of her environment. This means that a user is enabled to see what sensor technology surrounds her in her daily environment.

Solution A general solution is to add a view that provides augmented reality in a context-aware application. A user can, for example, use the camera view of her smartphone to visualize the sensors that surround her.

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.

- Support-U: With regard to the Support-U application, which offers a view directly in the flat of an elderly person, an overlay functionality is proposed that visualizes the sensors installed in the ubiquitous environment. These overlays enable the user to recognize easily the type of sensor that is installed, e.g., in his living or working place. In addition, the user is able to access the current data of the sensor and its data history by simply clicking the overlay that represents the sensor. An example of the implementation of the proposed pattern is given in fig. 2.
- Meet-U: Meet-U's indoor navigation incorporates RFID tags and readers to be able to localize the user without depending on a GPS signal. The user can view 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 will potentially increase and transparency will be provided. 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." [45]
- 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." [45]

4.3 Abridged Terms and Conditions

Intent The Abridged Terms and Conditions pattern shall enable the user to easily understand and overlook the terms and conditions (TAC) of the context-aware application. This can be achieved by displaying only the most important facts of the complete TAC in a condensed way such that it fits on one page.

Motivation Very often, lengthy TACs that are displayed at the first start of an application are ignored, respectively, browsed only very quickly by the user. The reason for this can be the representation, size and complexity of the TACs that may overwhelm the user. TACs can comprise up to 20 or sometimes more pages. In addition, the TACs are mostly not comprehensible to the user because they are written in legal language. Legal terminologies are utilised by companies in order to ensure that they are legally safeguarded.

Forces and Context

- Trust: Due to the fact that the TACs of an application are condensed to a size that is easily comprehensible, a user's trust in the application can be increased.
- Transparency: The usage of the pattern ensures a greater transparency to the user since possible implications for the user, which may result through the usage of the application, can be recognized more easily beforehand.

Solution A solution to provide comprehensible TACs to the user is the condensation of the TAC to the most relevant points. These have to be prepared from the user's perspective and not from the company's perspective. Therefore, the abstract should only include facts that affect the user. To enable the company to

stay legally safeguarded a full version of the TAC must be offered, too. Users can optionally select the full version if they need further information. The abridged version of the TAC may not exceed one screen page.

- Support-U: An example of an abridged TAC is given in fig. 3. The figure shows the results of the abridged TAC pattern used for the Support-U application.
- Connect-U: The user has to sign a license agreement of the size of one page in A4 format. On this page the agreement about the data usage is described in clear detail.
- Meet-U: The key points of TAC that affect the user's privacy the most, are displayed on one screen. Hence, the gathering and processing of data are addressed and summarized briefly. The long version of the TAC is linked. The user has to agree on that before continuing with the application.

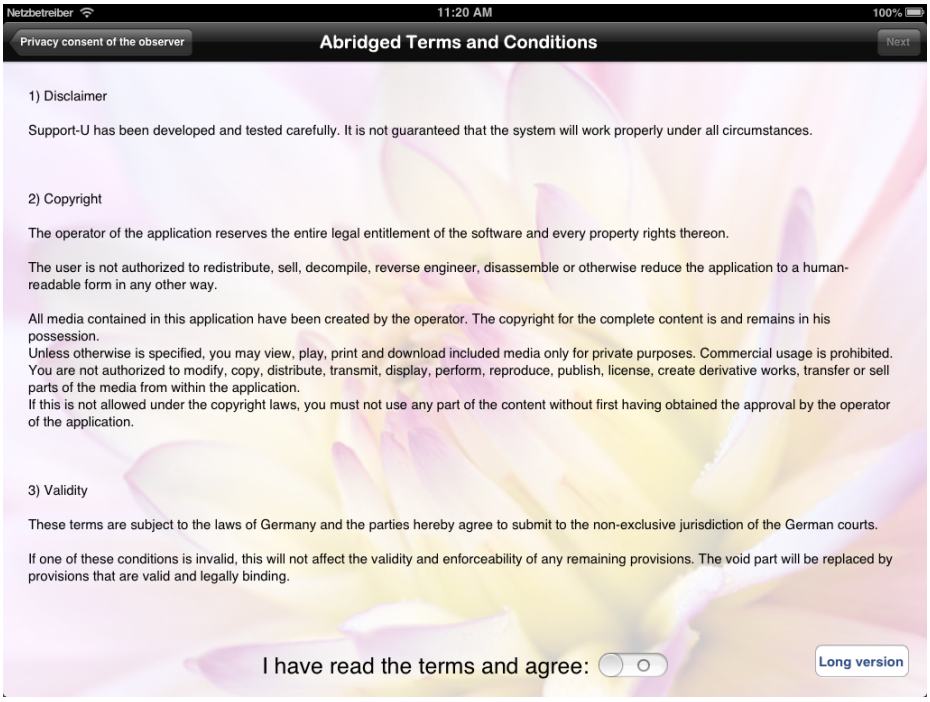

Fig. 3. The abridged TAC fits exactly on one screen. The long version can be accessed manually.

Consequences The pattern influences the way TAC are presented to the user. Instead of displaying all possible information to users, only those clauses will be presented using the abridged TAC pattern that are most relevant. If the user is

interested in the long version or needs further details she has to manually select the long version of the TAC. Moreover, a user can comply to the TAC by only reading the abridged version of the TAC.

Related Requirement Patterns

- Explanation of Processes: The application shall inform users on demand about processes.

Related HCI Patterns

- Extras on Demand: "Show the most important content up front, but hide the rest. Let the user reach it via a single, simple gesture." [43]

4.4 Context State Indication

Intent The intent of the Context State Indication Pattern is to ensure that all UC applications provide a similar way in how they handle, present and make important context information available to the user. Therefore, the pattern should advise the developer how context data can be categorized and how context data provided by an application can be indicated in a suitable way.

Motivation It can be quite a challenge to provide a good presentation of context data because context data is multifarious and is generated very frequently by sensors installed in UC environments or by services that infer context data directly from sensor data. Providing a good presentation supports the usability of the application and increases a user's confidence and trust in the application and its provider. Finally, a uniform concept for context data presentation and context access increases the recognition of certain functionalities in different applications.

Forces and Context

- Controllability: After starting the application, the user can easily obtain an overview of the relevant information. A quick overview supports her in making decisions faster.
- Effectiveness: The user does not has to interpret certain context data on her own but the user is supported by using usual indicators such as symbols, tones and colours, which automatically categorize a given context.
- Comprehensibility: The classification of contexts enhances a user's trust by preventing the user misinterpreting a certain context information, which would be visualized, e.g., only as a number.

Solution A solution to provide a clear presentation of context data that simultaneously enables the user to use the application more efficiently is given by a categorisation of context data, by a prominent placement of context data and finally by an easily accessible visualisation of the context information and its changes over time. The idea of categorising context data is to simplify the usage and speed up the interpretation of context data. In contrast to plain text, which has to be read first, the user can interpret known symbols, colours and tones within a very short time.

- Support-U: The categorisation of context data is defined by means of different colours. Green colour signals to the user that the context has a positive status, yellow colour means that the user should have a closer look to a particular context and red colour alerts the user. The most relevant context data is always displayed on the first screen of the application. This ensures that a user cannot miss it. Context changes are indicated by a fixed status bar, irrespective of whether or not the user is currently using the overview. An example of the usage of the proposed pattern is given in fig. 4. In the example the status of the person is marked as positive (green). The status of the TV is highlighted in blue, which indicates a neutral situation.
- Meet-U: One of Meet-U's functionalities is to inform the user in due time about his next step, such as the next appointment or event. To this end, Meet-U incorporates the user's location to estimate the time required to reach the destination. A yellow notification points out that the user has to start to arrive on time. In contrast, a red one indicates that he will be most likely too late.

Consequences The most relevant context information and the corresponding status levels have to be identified at design time. Furthermore, the navigation through the application is affected by the pattern and has to be adapted as well.

Related Requirement Patterns

- Assessment of the Output: The application shall offer users on demand an assessment of the output.

Related HCI Patterns

- Home & Idle Screens: "You must display a default set of information and actions once the device has started, and to return to when all other user activities are exited or completed." [45]
- Extras on Demand: "Show the most important content up front, but hide the rest. Let the user reach it via a single, simple gesture." [43]
- Zoom & Scale: "You must provide a method for users to change the level of detail in dense information arrays, such as charts, graphs, and maps. Design a zooming function or metaphor to provide this control." [45]

– Overview plus Detail: "Place an overview of the graphic next to a zoomed "detail view." As the user drags a viewport around the overview, show that part of the graphic in the detail view." [43]

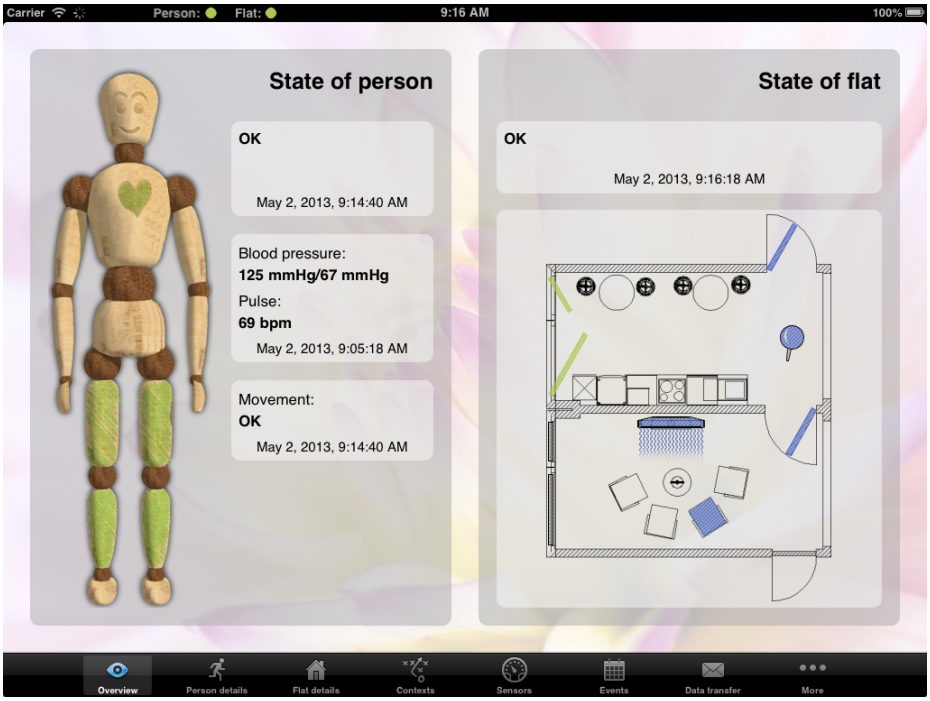

Fig. 4. The example shows the application of the Context State Indication Pattern in the Support-U application.

4.5 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. Users may sense a loss of control if the behaviour of an application is not comprehensible or if the behaviour disturbs the current interaction with the application. The pattern helps to create understandable autonomous adaption and prevents the feeling of loss of control.

Motivation 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 a user interaction, e.g. the user is inputting 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

- Informational self-determination: 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.
- Transparency: 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:

1. The user is currently interacting with the application. 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.
2. The user is currently not interacting with the application. 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 surveillance system or of a ringtone. The adaptation design needs to be tailored to the application domain, development platform, and target user group. The cooperation with a usability engineer and/or a trust engineer is recommended.

- Meet-U: Since Meet-U is a smartphone application there exist several methods to inform the user about already performed or upcoming adaptations. Possible HCI patterns to create notifications are listed below. Figure 5 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.

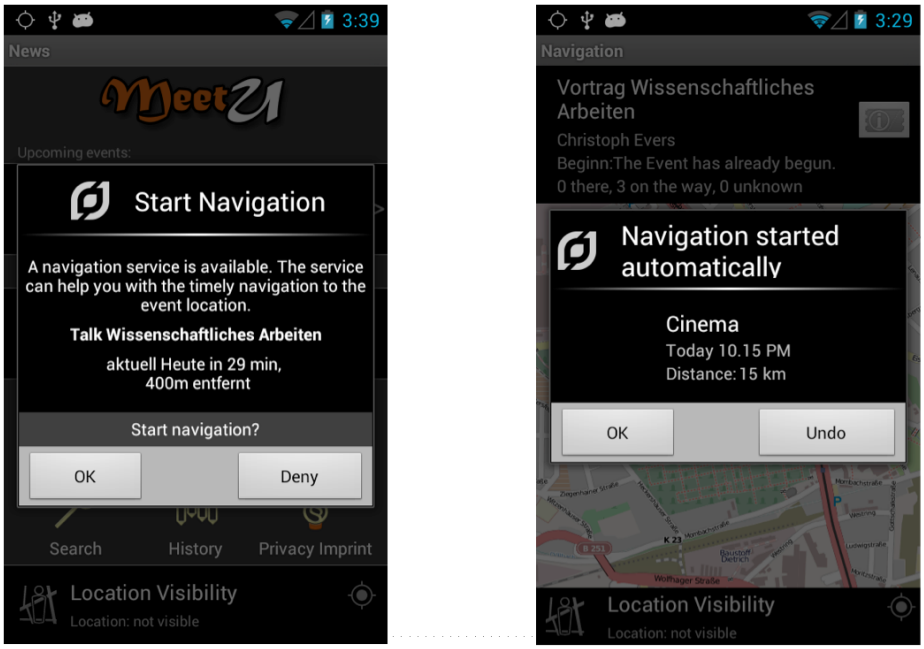

Fig. 5. Two examples in Meet-U for adaptation notifications.

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." [45]

4.6 Emergency Button

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

Motivation Many UC systems 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, a smartphone application may support the user during navigation to a meeting with a group of friends. The application 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 collecting and sharing his position.

Forces and Context

- Informational self-determination: 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).
- Trust: 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 that situations where the user wants to be invisible to the application.

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 pressing 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: In case of a smartphone application 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 which functionalities cannot be provided without restrictions. Besides, an icon in the notification bar may inform the user that the emergency button is activated.
- Support-U: In the case the person whose personal data are utilised by an ubiquitous computing application is not the person that has control over the application, a so called "button-on-the-wall" is needed. An example of such a monitoring application is given by Support-U that transmits the collected data to a person who takes care of an elderly person living under observation. After the "button-on-the-wall" has been pushed, the context data collection process, the processing of already gathered context data as well as the provision respectively 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 pressing the button, all functionalities, which require personal data, need to be deactivated to prevent errors at runtime. The Emergency Button Pattern can be combined with the Enable/Disable Functions Pattern which addresses similar concerns.

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." [45]
- 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." [40]

4.7 Data Access Log

Intent The usage, processing and gathering of personal data by an application have to be traceable and comprehensible for the user. This means that the user

must be able to find out when, by whom and for what purpose the data was collected. Hence, this pattern should be applied if personal data is collected, used or processed. Even in case of pseudonymized data the application of this pattern is recommended since linked data sets may enable drawing inferences about the user or his environment.

Motivation It is common that UC systems collect personal data to reason about the user's situation and provide appropriate adaptive services. In most of these systems the user has to grant the respective rights to the requesting application. However, in order to prevent a misuse of these rights or to be at least able to identify that, the access to personal data and its use should be logged. Let us assume an application is asking for access rights to collect some personal data like the location of the user, allegedly to set up the date format and language of the application appropriately. In case the application does not directly provide any location-based services, the gathering of location data remains questionable. The user can trust the application blindly or verify the usage of the collected data by checking the log files provided by an implementation of this pattern. The frequency of gathering location data can already indicate a misuse of the granted right. Further information listed in the log file may concretize this suspicion. This applies not only to location data but also to other sensor information or personal data.

Forces and Context

- Transparency: Using this pattern the user is put in a position to test whether the application sticks to the principle of data avoidance and to the intended purpose of use.
- Informational self-determination: A key pillar of informational self-determination is to know first which data is actually collected about you. On this basis, further patterns or methods can be applied to influence the use and processing of the data.
- Trust: By enabling transparency the pattern may strengthen the user's trust in the application. Besides, the user can cite the logged data as proof in case of legal disputes.

Solution The user has to consent first that the data is recorded with a tamper-proof method to the local storage. This can be done, for instance, during the installation, but before any personal data is submitted. However, the user should also be given the option to install the application with deactivated logging and to activate it if necessary afterwards. In case of activated logging the access to and the use of personal data and exchanges of them with remote or third parties are recorded in a tamper-proof manner to the user's device. It is important to ensure that the log files are copy protected and that a user has access only to his own log file. For that reason, the files should be encrypted and protected by passwords.

In general, the user should be enabled to detect infringements of the applications against national data protection laws. The appendix of section 9 of the German Federal Data Protection Act (BDSG) prescribes, for instance, various requirements to comply with the data protection law formulated in section 9 (§9, BDSG). The requirements concern, inter alia, the access, the use and the transfer of data. By being capable to uncover offenses against these requirements, the user is able to supervise the application's compliance with the data protection law to a certain extent. Nevertheless, it would be advantageous if every access, use and exchange of personal data would be logged. This is to firstly ensure that the application sticks to the principle of data avoidance and to the intended purpose of use, and secondly to be independent of national regulations. However, limited resources like the available storage and the principle of proportionality may restrict this. Depending on the application and its context a trade-off between the extent of data logging, its usefulness and its complexity and cost has to be determined.

- Meet-U: To implement the data logging pattern for smartphone applications a proxy can be used that notes down automatically the time of access, the collected personal data and the name of the invoking application. Further information like the dependent functionality or whether the data is processed remotely or by a third party have to be transmitted to the proxy by the invoking application. To save storage the proxy should summarize or aggregate frequent information, especially frequent sensor data. Besides, it is advisable that the user can define the maximum size of the log file. For inexperienced users a default value should be predefined.
- Support-U: To fulfill the data logging pattern, Support-U stores collected context data and data derived by context recognition or prediction algorithms to a data base. The information that has been stored in the data base can be accessed using Support-U.
- Connect-U: Concerning context data Connect-U measures the face-to-face proximity between users and the position of users by incorporating RFID readers and tags. Further information like the user profile and the conferences he or she attended can be combined with data from social networks such as BibSonomy and Twitter to recommend experts for certain topics. By using explanation aware computing [52,5,53], users can gain an insight into the inferences derived from the data.

Consequences By incorporating a tamper-proof recording the log file could be used as evidence in court. However, the question arises whether the data has been recorded truthfully. A solution would be to certify the data logging or to

integrate a certified third party that records the applications' access, use and processing of personal data.

Related Requirement Patterns

- Logging Processes: The application shall inform users on demand about processes carried out by the application.

Related HCI Patterns

- Interaction History: "Record the sequence of interactions as a 'history'." [40]

4.8 On Demand Explanation

Intent The user receives a detailed description about how the system reacts on his or her inputs. It is especially important to transparently inform the user how the data entered by him or her is employed by the system. When applying this pattern, the user is more aware of the consequences of his or her actions. This increases the usability and the user's trust in the system at the same time.

Motivation Functions, buttons and settings like the privacy options are often not as clear and understandable as they should be. For example, it may be unclear for a user in a social network, even after configuring the settings, if his or her location, pictures and social relations are visible to third parties or not. It should be unambiguous at any time to whom the personal data is visible, how it will be used and which functionality is depending on that. This will also apply to other forms of inputs and interactions the user performs.

Forces and Context

- Transparency: The provided explanations point out what is affected by the action the user performs and how it is effected.
- Controllability: After understanding the meaning and impact of the different functions and settings, the user is able to use and configure them according to his preferences.

Solution Whenever the user accesses functions, settings or other forms of interactive elements, a detailed description of the available options should be provided. This applies especially to settings related to privacy. The explanations should be displayed without violating important ergonomic constraints. Particular attention has to be paid to small display sizes. In certain cases the explanations need to be displayed permanently or in a triggered way including the push of an "interface-help-button" such as F1 on desktop computers.

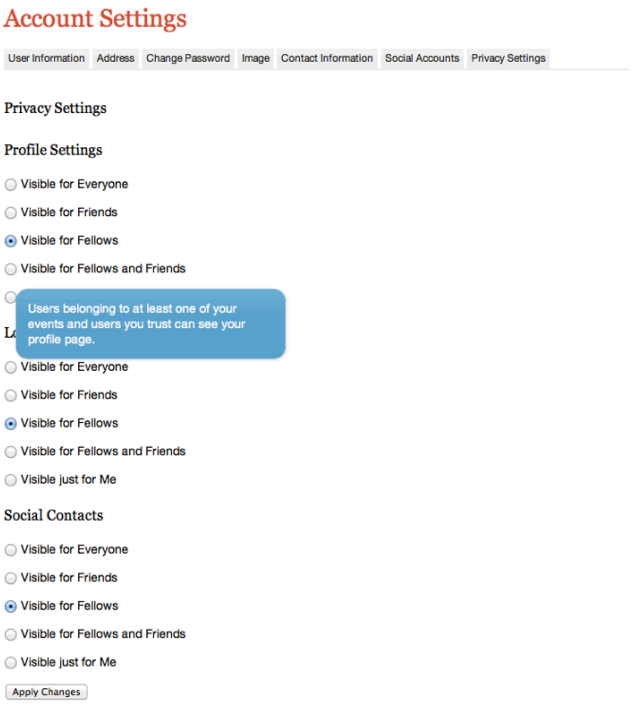

Fig. 6. The example shows the application of the On Demand Explanation Pattern. The picture outlines the possibility for the user to gain additional, detailed information. The HCI pattern used here is called Tooltip.

- Connect-U: According to the example in fig. 6 every interactive GUI element in Connect-U uses a tooltip to explain its functionality: this ranges from user interactions to detailed explanations for privacy options. Using tooltips no additional space is needed within the user interface.
- Meet-U: Certain settings and functions that may be unclear to the user at first sight, provide a question mark icon next to the according button or input field. As soon as the user clicks on them, a help text is popping up. In case of the user's settings concerning the privacy options and the user profile auxiliary texts inform him or her about the required data, the purpose of collecting them and whether the data is processed locally or remotely.

Consequences The explanations should also mention inferred data and data that is transferred to third parties.

Related Requirement Patterns

- Information About Functions: The application shall inform users before carrying out a function.

Related HCI Patterns

- Input Hints: Beside an empty text field, place a sentence or example that explains what is required. [43]
- Short Description: "Show a short (one sentence or shorter) description of a thing, in close spatial and/or temporal proximity to the thing itself." [40]
- 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." [45]

5 Conclusions

This report has presented a set of interdisciplinary design patterns for UC systems that were derived from three different case studies. While we are not the first to deliver design patterns for UC, our contribution is unique because it puts specific emphasis on concerns related to the inevitable social embedding of the technology. This social awareness is especially important for UC systems that collect, store and process a large amount of highly personal user data. Due to the criticality of such data our proposed design patterns reflect abstract requirements stemming from law, ergonomics, and user trust considerations. Thus, they go beyond earlier pattern languages for UC that have concentrated mainly on functional and security-related concerns.

The presented pattern collection is not meant to be the final word on design patterns for UC applications. The usefulness of our patterns has to be evaluated in further application studies. Probably more and different kinds of patterns will emerge in other scenarios. Perhaps our pattern specifications are not concrete enough for other development teams in order to directly derive an implementation from the specification. It is not clear yet how much the inherently interdisciplinary approach raises conflicts with established software design principles such as "separation of concerns" and "modularity". Actually, our patterns are the opposite of separation of concerns because they merge concerns from different disciplines into combined design proposals. More research and development experience is needed.

However, working on the definition of the interdisciplinary patterns and studying their implementations in the three UC case studies Meet-U, Connect-U, and Support-U has revealed already major benefits of a pattern-based approach:

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. Thus, they 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.

Acknowledgement

Funded by the federal state of Hesse within its LOEWE program for promoting cutting-edge research (from 2010 to 2013).

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. M. Atzmüller, M. Becker, S. Doerfel, M. Kibanov, A. Hotho, B.-E. Macek, F. Mitzlaff, J. Mueller, C. Scholz, and G. Stumme, "Ubicon: Observing social and physical activities," in *Proc. 4th IEEE Intl. Conf. on Cyber, Physical and Social Computing (CPSCom 2012)*, 2012.
3. 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
4. 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.
5. B. Forcher, S. Agne, A. Dengel, M. Gillmann, and T. Roth-Berghofer, "Semantic logging: Towards explanation-aware das," in *ICDAR*. IEEE, 2011, pp. 1140–1144.
6. 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.
7. G. Rossi, S. Gordillo, and F. Lyardet, "Design patterns for context-aware adaptation," in *SAINT 2005, Workshop on Context-aware Adaptation and Personalization for the Mobile Internet*. IEEE, 2005, pp. 170–173.
8. E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
9. M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, Sep. 1991.

10. A. Hoffmann, M. Söllner, A. Fehr, H. Hoffmann, and J. M. Leimeister, "Towards an approach for developing socio-technical ubiquitous computing applications," in *Sozio-technisches Systemdesign im Zeitalter des Ubiquitous Computing (SUBICO 2011) im Rahmen der Informatik 2011 - Informatik schafft Communities*, vol. GI-Edition - Lecture Notes in Informatics (LNI), P-175. Bonner Köllen Verlag, 2011, p. 180.
11. K. David, K. Geihs, J. M. Leimeister, A. Roßnagel, L. Schmidt, G. Stumme, and A. Wacker, Eds., *Socio-technical Design of Ubiquitous Computing Systems*. Springer, 2014.
12. S. Lahlou and F. Jegou, "European disappearing computer privacy design guidelines, version 1.1," 2004, ambient Agora-s IST 2000-25134.
13. M. Langheinrich, "Privacy by designprinciples of privacy-aware ubiquitous systems," in *Ubicomp 2001: Ubiquitous Computing*. Springer, 2001, pp. 273–291.
14. A. J. Ramirez and B. H. Cheng, "Design patterns for developing dynamically adaptive systems," in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 2010, pp. 49–58.
15. D. Gross and E. Yu, "From non-functional requirements to design through patterns," *Requirements Engineering*, vol. 6, no. 1, pp. 18–36, 2001.
16. L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle, "The platform for privacy preferences 1.0 (p3p1. 0) specification," *W3C recommendation*, vol. 16, 2002.
17. M. Langheinrich, "A privacy awareness system for ubiquitous computing environments," in *UbiComp 2002: Ubiquitous Computing*. Springer, 2002, pp. 237–245.
18. 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
19. 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.
20. J. A. Landay and G. Borriello, "Design patterns for ubiquitous computing," *IEEE Computer*, vol. 36, no. 8, pp. 93–95, 2003.
21. M. Hafiz, "A pattern language for developing privacy enhancing technologies," *Software: Practice and Experience*, vol. 43, no. 7, pp. 769–787, 2013. [Online]. Available: http://dx.doi.org/10.1002/spe.1131
22. D. 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.
23. 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.
24. M. Atzmüller, 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 (im Erscheinen), o.J.
25. 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.

26. A. Hoffmann and S. Niemczyk, Eds., *Die VENUS-Entwicklungsmethode. Eine interdisziplinäre Methode für soziotechnische Softwaregestaltung.* kassel university press GmbH, 2014. [Online]. Available: http://www.uni-kassel.de/upress/online/OpenAccess/978-3-86219-550-3.OpenAccess.pdf
27. 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.
28. S. Robertson and J. Robertson, *Mastering the requirements process.* Boston: Addison-Wesley Professional, 2006.
29. C. Wahono, "On the requirements pattern of software engineering," in *Proceedings of the Temu Ilmiah XI*, 2002, pp. 1–7.
30. 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.
31. K. Geihs, J. M. Leimeister, A. Roßnagel, and L. Schmidt, "On socio-technical enablers for ubiquitous computing applications," in *The 12th IEEE/IPSJ International Symposium on Applications and the Internet (SAINT).* IEEE, 2012, pp. 405–408.
32. A. Hoffmann, H. Hoffmann, and M. Söllner, "Fostering initial trust in applications - developing and evaluating requirement patterns for application websites," in *21th European Conference on Information Systems (ECIS)*, 2013.
33. A. Hoffmann, M. Söllner, H. Hoffmann, and J. M. Leimeister, "Deriving requirement patterns to increase users trust in sociotechnical systems," *Transaction on Management Information Systems (under review)*, o.J.
34. A. Hoffmann, T. Schulz, J. Zirfas, H. Hoffmann, A. Roßnagel, and J. M. Leimeister, "Rechtsverträglichkeit als Eigenschaft soziotechnischer Systeme - Ziele und Anforderungsmuster," *Wirtschaftsinformatik — Business and Information Systems Engineering (under Review)*, o.J.
35. M. van Welie. (2008) Patterns in interaction design. [Online]. Available: http://www.welie.com/patterns/
36. I. O. for Standardization, *ISO 9241-11: Ergonomic Requirements for Office Work with Visual Display Terminals (VDTs): Part 11: Guidance on Usability.* ANSI, 1998.
37. M. Van Welie, G. C. Van Der Veer, and A. Eliëns, "Patterns as tools for user interface design," in *Tools for Working with Guidelines.* Springer, 2001, pp. 313–324.
38. C. Kruschitz and M. Hitz, "Analyzing the hci design pattern variety," in *Proceedings of the 1st Asian Conference on Pattern Languages of Programs.* ACM, 2010, p. 6.
39. T. Coram and J. Lee. (1996) Experiences - a pattern language for user interface design. [Online]. Available: http://www.maplefish.com/todd/papers/experiences/Experiences.html
40. 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
41. J. O. Borchers, "A pattern approach to interaction design," *AI & SOCIETY*, vol. 15, no. 4, pp. 359–376, 2001.
42. I. Graham, *A pattern language for web usability.* Addison-Wesley Longman Publishing Co., Inc., 2002.

43. J. Tidwell, *Designing interfaces*. O'Reilly Media, Inc., 2005.
44. D. K. Van Duyne, J. A. Landay, and J. I. Hong, *The design of sites: patterns, principles, and processes for crafting a customer-centered Web experience*. Addison-Wesley Professional, 2003.
45. S. Hoober and E. Berkman, *Designing mobile interfaces*. O'Reilly Media, Inc., 2011.
46. C. Kuner, *European Data Protection Law, Corporate Compliance and Regulation*. Oxford University Press, 2007.
47. 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.
48. S. L. Lau, I. König, K. David, B. Parandian, C. Carius-Düssel, and M. Schultz, "Supporting patient monitoring using activity recognition with a smartphone," in *The Seventh International Symposium on Wireless Communication Systems (ISWCS'10)*, York, UK, 2010.
49. 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.
50. 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.
51. K. David and A. Flach, "An innovative car-2-x system concept for pedestrian safety," *IEEE VTC Journal*, pp. 70–76, mar 2010.
52. C. S. Sauer, A. Hundt, and T. Roth-Berghofer, "Explanation-aware design of mobile mycbr-based applications," in *ICCBR*, ser. Lecture Notes in Computer Science, B. Díaz-Agudo and I. Watson, Eds., vol. 7466. Springer, 2012, pp. 399–413.
53. B. Forcher, T. Roth-Berghofer, M. Sintek, and A. Dengel, "Explanation-aware software design of the semantic search engine koios." in *ExaCt*, T. Roth-Berghofer, N. Tintarev, D. B. Leake, and D. Bahls, Eds. University of Lisbon, Portugal, 2010, pp. 13–24. [Online]. Available: http://dblp.uni-trier.de/db/conf/exact/exact2010.html#ForcherRSD10