New Pattern Language Concepts for Designing UbiComp Applications Connecting to Cloud Services

René Reiners
Fraunhofer Institute for Applied Information Technology FIT
Schloss Birlinghoven
53754 Sankt Augustin, Germany
rene.reiners@fit.fraunhofer.de

Abstract: Ubiquitous computing applications that make use of Cloud services need to be designed in a way that the access to the Cloud can be easily performed by the user. We stress that successful design concepts need to be made reusable and combinable with each other in order to construct new kinds of applications. *Design patterns* have proven to be a successful and applied method to describe, capture and structure successful design knowledge.

For the large field of ubiquitous computing, we extend the concepts of the ”traditional” pattern languages as well as the pattern structure. The aim is to cover a larger domain of ubiquitous computing applications in this field. Additionally, our approach introduces collaborative features to discuss and refine existing patterns as well as to add new knowledge from different application designers. This way, we want to increase the pattern languages' liveliness.

1 Ubiquitous Computing and the Cloud

The concept of *Ubiquitous Computing* (*UbiComp*) introduced by [Wei91] in 1991 can be regarded as the common conceptual root for many current hardware designs and applications. Each of them concentrates on special aspects of ubiquitous computing and partially varies them. Currently, devices that can be used in a very ”ubiquitous” way are entering the market; mainly these are netbooks, smartphones or tablets. With a high degree of connectivity and new generations of different kinds of sensors, new applications and ways of interaction become possible that were still visionary some years ago. Mobile devices, however, often lack of computational power due to space restrictions and the need for saving battery. They concentrate on a set of special features for which they are optimized.

Processing power in the field of high performance computing (HPC), was primarily designed for science and industry to tackle computational hard problems and allow for a scalable provision of services. Nowadays it finds its way to every user as the *Cloud*. Together with the shaping of this denomination, providers begin to open their infrastructures. This in principle enables every application to benefit from a large amount of memory and processing power. More details on the Cloud concepts can be found in [Rho10].

erschienen im Tagungsband der INFORMATIK 2011
Lecture Notes in Informatics, Band P192
ISBN 978-3-88579-286-4

1.1 Connecting the Cloud and UbiComp Applications

The concepts and visions of ubiquitous computing together with emerging Cloud services support the development of lightweight applications for smart and mobile devices which outsource hard computational tasks or storage to remote systems. This way, rich applications for mobile applications on resource restricted devices can be realized. Amazon or Google, for example, provide access to their processing powers by introducing the Amazon Elastic Compute Cloud[1] or Google App Engine [2]. In the field of location-based services, a combination of using GPS data together with permanent network access by mobile providers are used in projects like Layars[3] or Wikitude[4].

More approaches than just receiving information are realizable which is for example realized by the UbiLens project at Fraunhofer FIT (cf. [RW09]). In this project, smart services are virtually attached to any kind of real world object and can then be accessed by a mediator. Their purpose ranges from information retrieval over triggering actions up to the combination of different services and devices. In a similar way, the *Smart Home Project* displays the energy consumption of different devices and allows the user to manage them remotely [JJP+10]. The mobile phone's camera pictures are sent to a remote recognition server in the Cloud.After recognizing the objects, the energy consumption is pulled from another information source and displayed on the mobile phone's screen.

In our view, due to the manifold of applications and ubiquitous services, accessing and the interacting with them can be very different. A lot of services are currently encapsulated into small applications for mobile devices. That way, a certain generalizability in terms of interaction and application design is achieved. However, we need to find application design methods that bridge the gap between available *functionality at hand* (i.e., ubiquitous devices) and accessing *remote processing power* (i.e., cloud services).

Our approach aims at capturing best design practice in an extensible pattern language structure. First steps in this direction were taken in [Rei10] where attempts for generalizing ubiquitous services and, in our opinion, missing features of the current pattern language approach are described. With regard to that work, we now extend the features of the traditional pattern language approaches and the pattern format itself. In contrast to the work by [Nob98] and [GHJV95], we describe *application design pattern* and their relations on a more informal and conceptual level in contrast to the technical notion of software design patterns. The next chapter describes the traditional pattern language approach for application design patterns, sections 3.1 and 3.2 discuss our conceptual approach concerning the extensions on the pattern language features and the patterns' structure.

[1] http://aws.amazon.com/ec2/
[2] http://google.com/appengine
[3] http://layars.com
[4] http://wikitude.org

ISBN 978-3-88579-286-4

2 Design Patterns

Reoccurring design problems are a well-known problem affecting many different domains. Not only in the technical sector but also in design areas like architecture, handcraft or construction or interface design, people see themselves faced with problems they do not know a solution for. However, it could be the case that the same or a similar problem has already been treated by someone else and that a solution or guideline was found. In a best case scenario, this ”best practice” was written down or conserved in a way that it can be shared and distributed. For example, distribution channels can be the oral or written dissemination of knowledge that are used during education or that can be consulted on demand.

A design pattern represents design knowledge for a specific domain, discusses the context in which it can be applied and explains ways to solve a certain problem. Collections organize and structure particular units of information in a way that they can be organized hierarchically thus having different degrees of information detail. The patterns are hierarchically organized, e.g., by making use of the spatial dimension as described in [Ale77] or the level of detail of a pattern reflecting the time dimension in the design process [Bor01, SL07]. Others, like the GoF patterns remain on a very technical level and are structured by their purpose in the scope of programming languages [GHJV95].

Besides this informal way of describing design solutions, semiformal and formal approaches were developed like in [RU04, CH00]. For our approach, however, we make use of the informal pattern language formulation since we aim at leaving them readable for many stakeholders in the application design process.

3 New Concepts for a Pattern Language

The pattern language structure described in this work will follow the general concepts as described by [Bor01] and [SL07]. We see the need for extensions aiming at widening the application domain of pattern languages which are currently mainly bound to specific application domain problems. Another aim is increasing the vividness of a pattern languages; in current approaches, a group of authors, i.e. domain experts, extracts and formulates sets of interconnected patterns. The collections are published in books or online. However, revising the generated patterns or refining them is not easily realizable, especially with paper prints. The same holds true for the introduction of new patterns or the disproval of an existing one. We also aim for external authors' contributions. In the following, we introduce conceptual features extending the conventional pattern language concept.

3.1 New Concepts for the Pattern Language Structure

Branching: While browsing a pattern language, we introduce decisions at which the reader needs to decide for a subset of design recommendations. For example, the deci-

ISBN 978-3-88579-286-4

sion may be based on interaction style, privacy demand or device footprint. After making a decision, other subsets of design patterns are no longer available since they do not match the criteria anymore that were set by the decision. Concerning the pattern language visualization, this leads to a dynamically chaning graph structure of causally related patterns.

Community Involvement: The pattern language approaches discussed so far all relied on the knowledge of one smaller group of authors and their domain knowledge. For some pattern languages, writer's workshops were held to formulate and refine patterns. Users can also propose new patterns to be linked within the pattern language structure. This way, the *vividness* of the pattern language is to be increased making it adoptable or extensible for more application domains.

Sequencing and Recommendation: Pattern sequences can be inferred from rankings, most read patterns or proposed combinations by other readers. This way, different alternative paths can be weighed by recommendations given by the community or by (semi-) automated proposals. Thus, proven combination of design patterns can be given the application designer.

Discussion and Refinement: Collaboration also includes the rating of patterns, references to related work and pattern descriptions. Since the pattern language can be adapted due to community involvement, rules and processes for retrieving and refining pattern are needed.

3.2 Extended Pattern Structure

In our approach, we apply the widely known pattern format of [Ale77, Bor01] and [SL07] which consists of the follwowing fields: name, rating, illustration, context, problem and forces, example, solution and diagram. We extend this set by new the features described in the following. Features marked by a star represent an extension of already existing properties in the referred pattern language approaches.

Rating*: In the approaches described so far, patterns were ranked based on the experience of the author group providing the patterns - eventually influenced by groups of first readers. The ranking will be extended in such a way that readers and implementers of the pattern are enabled to provide feedback about the quality of the pattern based on their own knowledge and experience.

Context*: Normally, the context is clearly defined by the pattern language structure. However, due to the restriction of design space given by decision nodes, readers have to keep track of taken decisions on the path to the particular pattern. It is possible that the problem domain is narrowed or changed to different problem cases by taking different decisions.

Solution / Reason*: This part provides a generally applicable design solution based on the lessons learned from the examples and literature references. It summarizes the central message of the design pattern telling the reader how to handle a specific design problem. In case that the pattern provides an anti-solution, this part turns into a description of the reasons for the failure of the concept. Both, the solution and the reasons should be described succinctly and therefore deliver the key message of the design (anti-) pattern.

ISBN 978-3-88579-286-4

Evidence in Literature: Besides examples that already provide some evidence for a working solution, we regard references in literature as very important for supporting the design recommendations given by a pattern. That way, application designers and researchers have another way of judging a design pattern besides the collaborative ranking mechanism described above. They can strengthen or weaken the design pattern's statement by a custom literature references or published evaluations.

References*: These are pointers to other patterns of interest. We extend the concept by introducing *decisions on references*; a set of succeeding patterns can be marked as mutually exclusive alternatives. This way, the reader needs to decide for one successor. The different alternatives concentrate on different aspects and realizations of application design contexts. E.g., concepts for private together with public information display or personal and shared devices are contradicting features and cannot be realized in parallel.

Anti-Pattern-Attribute: The intended pattern language approach also incorporates the explicit formulation of anti-patterns. Marked as anti-pattern, the described design approach should be read as "DON'T rules" for design. Only surprising, non-trivial results should be documented. Thus, application designers can avoid similar design flaws. Anti-patterns can also provide ideas for alternative concepts; once the application designer has followed a pattern sequence, she could also think of changing certain design parameters and try out another concept that is not yet documented.

4 Summary and Outlook

In this work, we addressed the requirements for establishing a pattern language structure for connecting UbiComp applications to Cloud services. Based on existing concepts from architectural, HCI and HCHI design patterns formulated by C. Alexander, J. Borcher and Schuemmer and Lukosch [Ale77, Bor01, SL07], we formulated new features for the existing pattern language and pattern structures. Key elements are the introduction of decisions, pattern sequences and anti-patterns. Additionally, we intend to integrate new collaboration features to better involve communities to increase the vividness of formulated patterns and the pattern language. Ways to express pattern feedback by rating and refining existing patterns or contributing new findings will be integrated into the pattern language resulting in an open, collaborative community platform.

Future work will cover the formulation and arrangement of a first set of design patterns. Since the connection between ubiquitous computing applications and Cloud services must be made transparent and understandable for the user, one class of patterns will concentrate on best practices describing successful approaches for these design challenges. In a further step, a first version of an online community portal that allows for reading and rating the existing patterns will be deployed, followed by rules and processes for rating, discussing, refining and contributing to the pattern language.The mechanisms for the pattern discussion and refinement will be based on the concepts applied during writersâ workshop at Pattern-Language of Patterns (PLoP) conferences[5].

[5] http://www.hillside.net

ISBN 978-3-88579-286-4

5 Acknowledgements

This work was supported by the European Commission within the FP7-SECURITY project BRIDGE, project No. 261817)

References

[Ale77] Christopher Alexander. *A Pattern Language: Towns, Buildings, Construction*. Oxford University Press, New York, New York, USA, 1977.

[Bor01] Jan Borchers. *A Pattern Approach to Interaction Design*. John Wiley & Sons, 1st edition, 2001.

[CH00] A. Cornils and G. Hedin. Tool support for design patterns based on reference attribute grammars. In *Proceedings of WAGAâ00*, Ponte de Lima, Portugal, 2000.

[GHJV95] Erich Gamma, Richard Helm, Ralph E. Johnson, and John Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software*. Addison-Wesley Longman, Amsterdam, 1 edition, 1995.

[JJP+10] Marco Jahn, Marc Jentsch, Christian R Prause, Ferry Pramudianto, Amro Al-akkad, and René Reiners. The Energy Aware Smart Home. In *Proceedings on the 5th International Conference on Future Information Technology (FutureTech), 2010*, pages 1–8, Busan, South Korea, 2010.

[Nob98] J. Noble. Classifying relationships between object-oriented design patterns. In *Software Engineering Conference, 1998. Proceedings. 1998 Australian*, pages 98 –107, nov 1998.

[Rei10] René Reiners. Towards a Common Pattern Language for Ubicomp Application Design - A Classification Scheme for Ubiquitous Computing Environments -. In *Proceedings of the International Conferences on Pervasive Patterns and Applications (PATTERNS 2010)*, pages 28–33, City, Portugal, November 2010. IARIA Conference, Think Mind(TM) Digital Library.

[Rho10] John Rhoton. *Cloud Computing Explained: Implementation Handbook for Enterprises*. Recursive Press, 2 edition, 2010.

[RU04] Jean-Marc Rosengard and Marian F Ursu. *Ontological Representations of Software Patterns*, volume 3215, pages 31–37. Springer Berlin / Heidelberg, 2004.

[RW09] René Reiners and Vina N. Wibowo. Prototyping an Extended Magic Lens Interface for Discovering Smart Objects in a Ubiquitous Environment. In Weghorn Hans and Pedro T. Isaías, editors, *Proceedings of the IADIS International Conference Applied Computing 2009*, Rome, November 2009. IADIS, IADIS Press.

[SL07] Till; Schümmer and Stephan Lukosch. *Patterns for Computer-Mediated Interaction*. John Wiley & Sons, Chistester, West Sussex, England, 2007.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.