From Structural Analysis to Scenarios and Patterns for Knowledge Sharing Applications

Claus Kaelber[1] and Christian Märtin[2]

[1] teamwissen
Entenbachstraße 35
81541 Munich, Germany
kaelber@teamwissen.de
[2] Augsburg University of Applied Sciences
Faculty of Computer Science
Friedberger Straße 2a
86161 Augsburg, Germany
Christian.Maertin@hs-augsburg.de

Abstract. In this paper we present a pragmatic development approach for knowledge sharing applications that encompasses both design and software engineering aspects. It starts from scenarios and leads to patterns that help application developers and user interface designers on the one hand to separate relevant content from unimportant data and on the other hand propose techniques for qualitatively structuring knowledge management and knowledge sharing tasks for enterprises and individuals.

Keywords: HCI patterns, structural patterns, domain patterns, knowledge sharing, knowledge management, design strategy, GUI generation, pattern-based modeling.

1 Introduction

A steadily growing information flood faces people both in business and everyday life. The many existing commercial solutions for knowledge management and knowledge sharing systems, however, in most cases even raise the complexity level for accessing relevant information or come with poorly designed user interfaces that require experts and keep other users away from working with the systems. We present a new approach starting with a requirements analysis driven by contextual design techniques that leads to scenarios from which structural, domain and HCI patterns are derived.

These patterns can be exploited by tools and generators in order to arrive at high-quality web-based user interfaces for knowledge sharing applications. These knowledge sharing applications are aimed at optimizing user experience and usability in contrast to powerful, but complex existing knowledge sharing environments and workflow management systems.

The paper discusses the final results of project *p.i.t.c.h. (p*attern-based *i*nteractive *t*ools for improved *c*ommunication *h*abits in knowledge transfers) which was conducted by the *A*utomation in *U*sability *E*ngineering group *(AUE)* at Augsburg

J.A. Jacko (Ed.): Human-Computer Interaction, Part I, HCII 2011, LNCS 6761, pp. 258–267, 2011.

University of Applied Sciences in cooperation with two medium-sized enterprises with engineering and production background and several partners from communication sciences and the knowledge management domain.

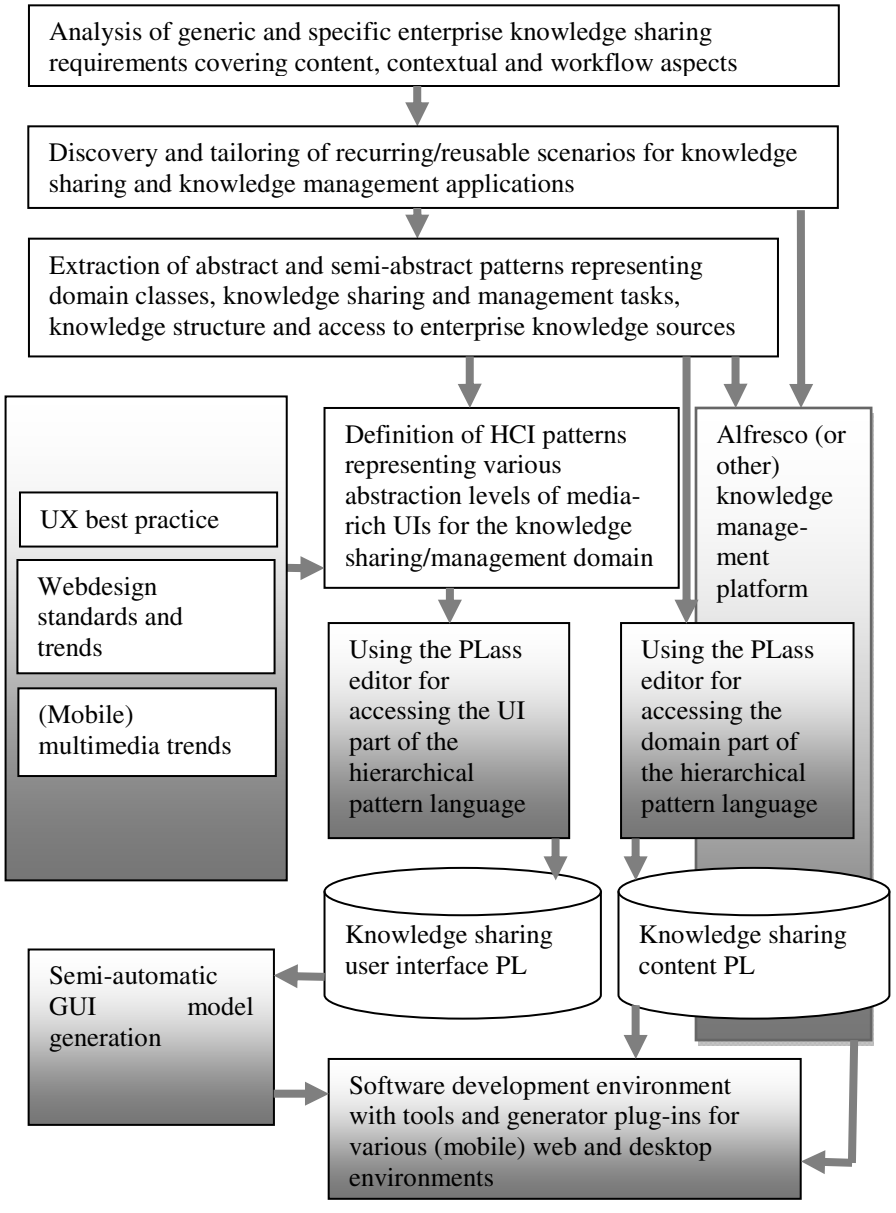

Fig. 1. Revised p.i.t.c.h. development workflow

The p.i.t.c.h. development environment is described in [6]. Figure 1 shows a revised version of the application design workflow that was devised in p.i.t.c.h. It illustrates how the content-specific analysis phase interacts with our scenario- and pattern-based development environment.

This paper focuses on the scenario-based modeling and design process that also includes techniques from contextual design [1], the resulting patterns and the final translation into interactive applications.

2 Design Strategy

One of the most important goals of the p.i.t.c.h. project was to set up a design strategy that would lead to more intuitive, simple functionality for searching, finding, storing and accessing data. See [4], [5] and [8] for background information on knowledge management systems. An appropriate strategy should enable the extension of existing content and the exchange of content between users in the same or in different organizations. It should also identify the individual skill-levels of the various users and their topics of interest.

Therefore the central requirements for the design strategy in p.i.t.c.h were

- reduction in knowledge sharing system complexity
- intuitive system usage
- fast system access
- interactive knowledge access and communication tools
- individual system configuration

In order to arrive at the required goals the project was structured into the following phases

- documentation and analysis of existing IT- and communication workflows in the participating enterprises
- definition of a design rationale for the flexible integration of multi-media formats into the target system
- definition of typical lead-scenarios
- derivation of reusable HCI patterns from requirements and scenarios
- tool-based automated development and test of interactive target applications

The aspects of joy of use and of rich user experience are still far away from being considered by most application developers, because they are mainly domain-focused or driven by application functionality. These rather psychologic properties of digital media access can be exploited to add substantial value to knowledge sharing environments. Driven by the prevalence of social networks future methods of information brokerage within organizations will need more collaborative and personalized communication tools. Therefore design strategies and development processes used in these application domains must provide a (semi-)formal basis for also including such rather informal user experience and personalization aspects into the development work.

Using scenarios that lead to patterns, enabling the patterns to also express UX [11] and non-formal individualization aspects, and using semi-automatic tools for generating the interactive parts of the final target applications was the key to the design strategy applied by the p.i.t.c.h. project.

In order to evaluate the design strategy with a prototype in a demanding and useful target context, during the initial project phase typical knowledge sharing and knowledge communication requirements of three medium-sized technology-driven enterprises were analyzed using contextual design inquiry techniques. The discovered requirements were clustered and scenarios that cover the most important workflow, knowledge access and communication aspects were derived.

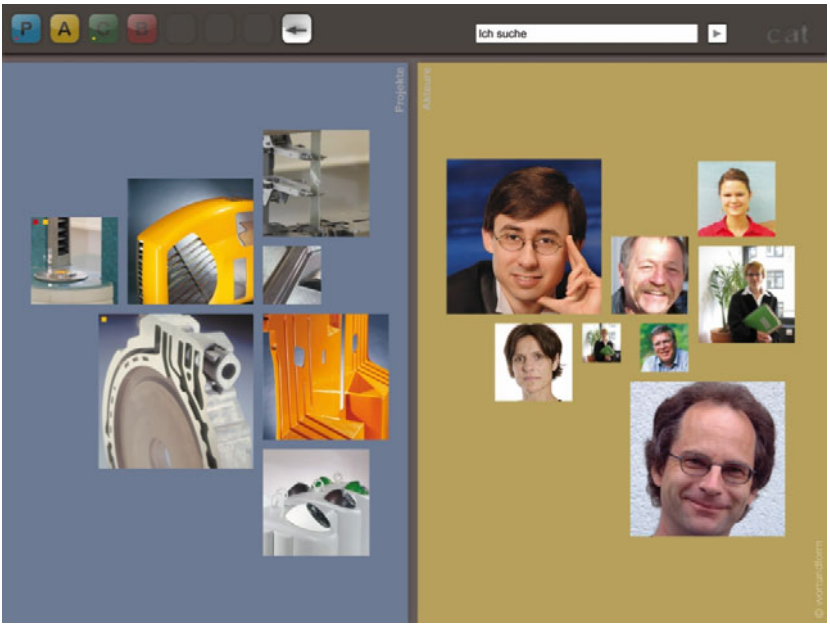

Fig. 2. p.i.t.c.h. orientation environment

From these the following lead-scenarios were selected for the prototype and as the basis for extracting reusable abstract, semi-abstract and concrete HCI patterns:

- Profile Board
- Infolog
- Collaborative Reporting

The scenarios apply the same basic organizational scheme that uses the *login* pattern to individually access initial *orientation environments* that allow for a global overview and further orientation within the collaborative knowledge sharing system. The starting point is represented by three abstract structural patterns: *orientation environment, protagonists* and *projects*. These patterns provide the infrastructure for accessing all relevant information and serve as the pivotal structures for all collaborative and communicative actions of the users.

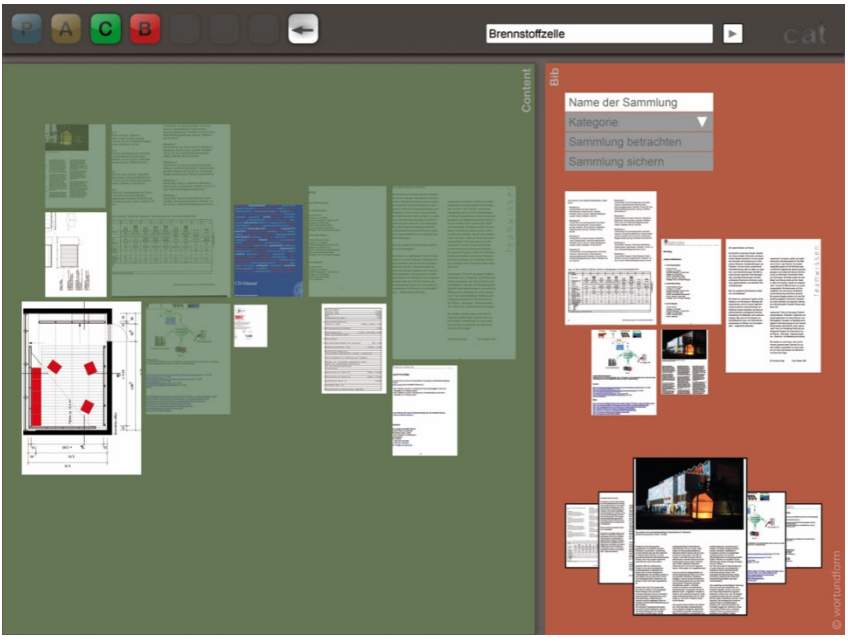

Fig. 3. User interface generated from document library pattern, domain and HCI patterns

The orientation environments are individually personalized (fig. 2). They are characterized by a specific visual ranking of the relevant content data (their size, color, position depending on up-to-dateness, scope, deadline etc.). Patterns were developed to represent these mappings. In the figure the individual network of protagonists and projects is visualized in the form of an information cloud or map. All functionality, internal, external links, documentation and communication tools are accessible from these orientation environments. Figure 3 shows the document library for a specific project. HCI patterns [7], containing the mappings from the XML content models to semi-abstract UI representations, were modeled for various types of knowledge sharing and communication tasks.

3 HCI Patterns and Content Models

Patterns in our environment are specified interactively using the PLass editor [12]. They are modeled with similar attributes, e.g. *name, problem, forces, context, solution, examples,* as can typically be found in design and HCI pattern languages (e.g. [2], [3], [9], see [13] for a comprehensive overview on pattern-based development of interactive systems). Internally they are represented in XML notation.

In addition to classic design patterns, our patterns provide the powerful attribute *automation* that either provides structural, content, and contextual information for refining more abstract into more concrete patterns or provides linking information from a concrete pattern to a single or a group of concrete interaction objects that can directly be exploited by automated user interface generator plug-ins.

XML is also used for content and domain modeling of the chosen application types. The *Alfresco* CMS [10] in its version 3.4 is used as our underlying model repository and knowledge server.

Starting from the structural, functional and workflow requirements that were defined during analysis and led to the resulting prototypical scenarios, both HCI and domain patterns could be derived from the scenarios. The domain patterns mainly define the domain classes, their attributes, functional and communication behavior. In order to allow an easy mapping to the target environment, the content of the *solution* attributes of the domain classes was directly mapped to XML representations of each domain class in the content model residing in the Alfresco model repository.

Alfresco offers useful open interfaces (e.g. RESTful Services (REST), Alfresco Web Scripts) for coupling and synchronizing data-oriented servers with visual clients and supports both Javascript and Java for developing the interactive client behavior. Alfresco´s REST provides an URL-based interface for resource access and state transfer between server and clients. It is used to access content data stored in the knowledge repository and map them to the final UI structure that was derived from the specified scenarios and HCI patterns.

Alfresco Web Scripts apply the MVC pattern with the XML based content model residing in the Alfresco repository, Java or Javascript as the Web Script controller language and the Freemarker Template Language as the toolkit for providing the visual elements (view). In order to support different target environments plug-ins for HTML 5, Javascript, Java, Adobe Flash, and Ajax were used by the interactive clients. The HCI patterns that were identified during analysis and scenario definition were linked to the representations of the domain patterns, kept in the model repository.

Figure 4 shows the structure and relationships of the various patterns and model components defined for the prototypical application based on the three example scenarios. The figure focuses on one of the scenarios, the *profile board.*

A scenario can itself be seen as a pattern that specifies the use-cases, possible tasks, workflow, domain knowledge as well as the structural and behavioral requirements of the user interface. Resulting from initial requirements analysis and an extensive contextual inquiry phase, the scenario-pattern attributes were specified as structured texts. Scenario patterns serve as the most abstract elements in the project pattern language. To link them to the more formal patterns of the later development stages the scenario-patterns are entered directly below the root element of the pattern language. The pattern attribute *automation* is filled with links to all structural patterns that are directly referenced by the scenario pattern.

From the beginning the project team included domain experts, designers, and computer scientists. Therefore, whenever possible, the most important structural patterns were already identified at this stage.

For the profile board scenario the structural patterns *orientation environment, protagonists, and projects* were identified (dashed lines indicate that a scenario uses these patterns). These patterns are reusable for model specifications throughout the entire knowledge sharing and knowledge management domain. The structural patterns are specified using the regular (HCI) pattern attributes provided by the PLass editor. As can be seen in fig. 4, structural patterns may reference other structural patterns, e.g. the *projects* and *protagonists* patterns are used by *orientation environment.* In the resulting pattern language *orientation environment* appears at a higher level in the pattern hierarchy.

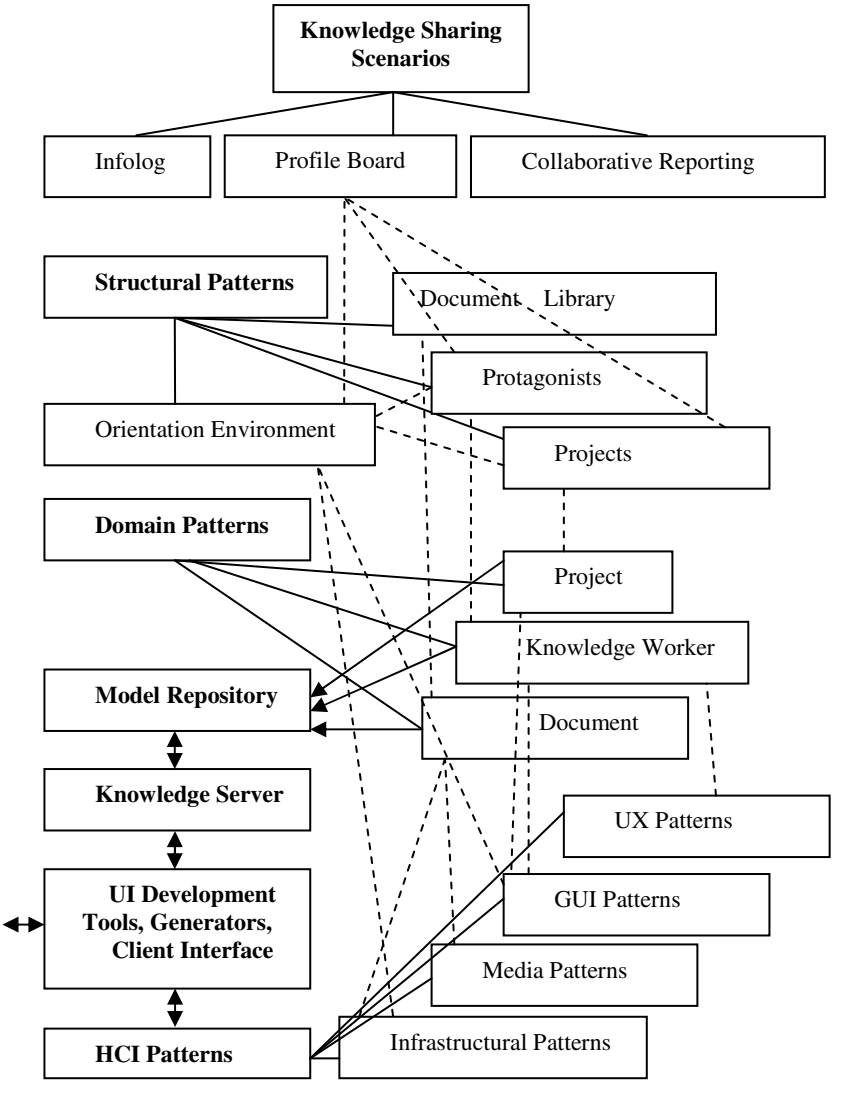

Fig. 4. Excerpt from the p.i.t.c.h. model structure

At the center of the knowledge sharing applications developed by the project a model of the major domain classes and their features is maintained in the model repository.

However, the domain classes are directly derived from the domain patterns identified during the analysis process. Domain patterns like e.g. *knowledge worker* or *project* specify the data structure of single or composed domain classes, their functionality and inter-class communication requirements. Domain pattern attributes are specified as structured text. Their *automation* attribute contains the XML

specification data and functionality that is stored by the Alfresco-based model repository during runtime together with some glue code. Thus the mapping of domain patterns to the underlying CMS (see the arrowed lines) can be automated. For the final application, the respective enterprise data can be filled in using the CMS templates structured after the domain pattern specifications.

As can be seen from fig. 4 domain patterns are separated from, but interact with the HCI patterns that specify all aspects of GUI representation, media requirements, contextual mapping to varying target environments as well as user experience. Using the automation mechanisms and plug-ins provided by the *PAGui* tools [12], parts of the GUI and interactive behavior of the target application can be generated, other parts are implemented with the HTML 5, Java and Javascript-based Web-development tools supported by Alfresco.

Table 1. User experience pattern for visually arranging project contributors

Attribute	Description
ID	Arrange protagonists
Problem	How to visually arrange protagonists that contribute to a project
Context	Find the most relevant communication partners when working on the project details
Solution	1. Compare the following attributes of the protagonists: last contact, type of contact, number of contacts 2. Evaluate the results in a layout algorithm that returns a position, size and coordinate-pair for each protagonist image. The algorithm is running each time the detailed project view is activated.
Illustration	Refer to figure 5.

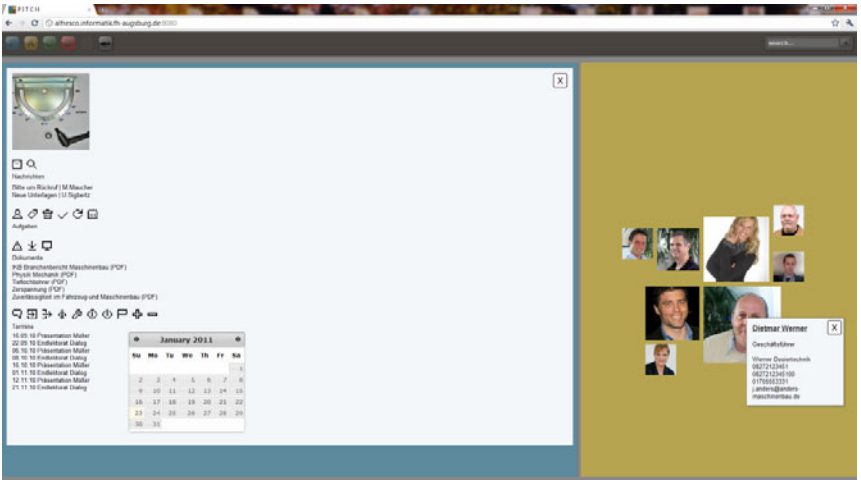

Fig. 5. User interface generated from *project*, *protagonist*, GUI and UX patterns

In figure 5 a partly generated user interface is shown that was derived from the structural patterns *project* and *protagonist*. It shows the detailed view of a project with documents, a to-do-list and a calendar. These properties are modeled in the *project* pattern. On the right-hand side, the project contributors are visualized by their images. A UX pattern (see table 1) is exploited for generating the layout of the contributors´ images.

4 Conclusion

The development workflow and the design strategy defined within the p.i.t.c.h. project were used to design and implement several prototypical knowledge sharing scenarios that serve as the basis for future, more comprehensive knowledge sharing environments for small and medium sized enterprises. The pattern language developed by the project is reusable and can be extended for additional domains.

The *profile board* scenario introduces protagonists with all their projects, fields of interest and their individual business network, whereas the *infolog* scenario presents an aggregation of media-dependent personal or project-specific content data in the form of an individualized, interactive kiosk. The *collaborative reporting* scenario establishes separated rooms for communicating and exchanging information with external business partners. These scenarios currently serve as an experimental platform for evaluating the quality and user experience possible with our underlying design strategy and the pattern-driven development process.

Future development stages will focus on mobile access to knowledge sharing applications, advanced visualization concepts for project-related real-time workflows and high-level automation tools for the development process.

Acknowledgements. Part of this research was supported by the Innovation Fund of the IHK Schwaben (Augsburg Chamber of Commerce), Augsburg, Germany. Our thanks also go to Christian Herdin and the student members of the M.Sc. CS project in the winter semester 2010/11 who were responsible for implementing the concrete UI patterns, the Alfresco content models and the domain part of the sample scenarios.

References

1. Beyer, H., Holtzblatt, K.: Contextual Design. Interactions, 32–42 (January-February 1999)
2. Buschmann, F., et al.: Pattern-Oriented Software Architecture – A System of Patterns. Wiley, New York (1996)
3. Designinginterfaces.com, Jennifer Tidwell, Patterns for Effective Interaction Design (March 10, 2010) , http://www.designinginterfaces.com/
4. Roumois, H.: Ursula: Studienbuch Wissensmanagement, Zürich, p. 34 (2007)
5. Lehner, F.: Wissensmanagement, Munich, p. 46 (2008)
6. Märtin, C., Engel, J., Kaelber, C., Werner, I.: Using HCI-patterns for modeling and design of knowledge sharing systems. In: Forbrig, P., Günther, H. (eds.) BIR 2010. LNBIP, vol. 64, pp. 1–13. Springer, Heidelberg (2010)
7. Marcus, A.: Patterns within Patterns. Interactions, 28–34, (March-April 2004)

8. Trondsen, E., Vickery, K.: Learning on Demand. Journal of Knowledge, Management, 169 (March 1998)
9. van Welie, M.: Welie.com, Patterns in Interaction Design (March 10, 2010), http://www.welie.com/patterns/
10. Alfresco ECM System product homepage (March 10, 2010), http://www.alfresco.com
11. Obrist, M., et al.: User Experience (UX) Patterns for Audio-Visual Networked Applications: Inspiration for Design. In: Proc. NordiCHI 2010, October 16-20, pp. 343–352 (2010)
12. Engel, J., Märtin, C., Forbrig, P.: Tool-support for Pattern-based Generation of User Interfaces. In: PEICS 2010, Berlin, Germany, June 20. ACM Int. Conf. Proc. Series, pp. 24–27 (2010)
13. Breiner, K., et al. (eds.) Proc. of the 1st Int. Workshop on Pattern-Driven Engineering of Interactive Computing Systems (PEICS 2010), Berlin, Germany, June 20. ACM Int. Conf. Proc. Series (2010)