Pattern-based Modeling and Development of Interactive Information Systems

Jürgen Engel, Christian Herdin & Christian Märtin

Augsburg University of Applied Sciences,
Faculty of Computer Science

An der Hochschule 1, 86161 Augsburg, Germany

{Juergen.Engel, Christian.Herdin, Christian.Maertin}@hs-augsburg.de

1 Introduction

For more than thirty years the field of model-based user interface development (MBUID) has produced many approaches, development systems, and specific tools for creating and tailoring individual and highly usable interactive systems. There are reasons, however, why MBUID after such a long time period, still has not made it to the typical software developer's workplace (Meixner et al., 2011). We feel that the two major reasons are the missing standardization of MBUID environments including model specification languages and tool support, and, the poor integration of the interactive development approach into the whole software engineering life cycle.

In order to target these two shortcomings of MBUID, we have focused our work on expanding the model-based development process for real-world application domains by including well-known practices from software engineering, e.g. scenarios and patterns into our framework, and thus making a more refined level-of detail and individualization exploitable for our modeling and generation tools.

As one target domain we have chosen the field of knowledge sharing systems. Requirements of the workflows, functionality, data access, and user interfaces for individual knowledge sharing and knowledge management applications may differ in details from enterprise to enterprise, but can often be modeled and refined on the basis of reusable scenarios and patterns. The p.i.t.c.h. project (Kaelber & Märtin, 2011) has led to a new software engineering approach for knowledge sharing systems. It proposes a pattern-based hierarchical method for developing real world knowledge sharing applications for medium-sized enterprises, starting from requirements analysis and scenarios, exploiting a hierarchical knowledge-sharing pattern language and arriving at high-quality, user-experience-driven target systems.

Areas of major interest for the user interface aspects of the p.i.t.c.h. project were the modeling of the various levels of a priori knowledge of the potential target users, the diversity of target platforms and media options, and the extraction of reusable scenarios from the results of a requirements analysis process that was driven by contextual design practices (Beyer & Holtzblatt, 1999). HCI patterns for various modeling purposes and on different abstraction levels were chosen as the major modeling paradigm for p.i.t.c.h. Important domain areas examined by the project were video-based interactive knowledge-sharing applications, e.g. guided tours (Märtin et al., 2010), and a peer-to-peer intra- and inter-organization communication platform with advanced support for accessing structured and unstructured company and expert knowledge residing in the companies' network file systems, databases, and content management systems. This paper concentrates on the pattern-based modeling aspects applied in p.i.t.c.h. and the benefits it can yield for MBUID approaches in general.

2 Related Work

Originally patterns were introduced by Christopher Alexander for solving problems in architecture and urban planning of buildings (Alexander et al., 1977). In the year 1995 the pattern concepts have been translated to the domains of software architecture and software engineering (Gamma et al., 1995). In the meantime patterns are also applied to the fields of human-computer interaction (Fincher et al., 2003), user experience (UX) (Tiedtke et al., 2005), usability engineering (Marcus, 2004), task modeling (Gaffar et al., 2004), and application security (Yoder & Barcalow, 1997).

3 Pattern-based Modeling and Development Approach

In order to use a structured approach for pattern definition, pattern access, user interface modeling and transformation to various abstraction levels of the p.i.t.c.h. target platforms, tools of the PaMGIS framework were used (Engel & Märtin, 2009). PaMGIS is an integrated framework developed by the Automation in Usability Engineering group (AUE) at Augsburg University of Applied Sciences. It supports developers and software designers with a combination of tools and model- and pattern-based methods. The crucial component of the framework is the pattern repository which contains hierarchical pattern languages and patterns for various problem domains consisting of diverse pattern types on different abstraction levels.

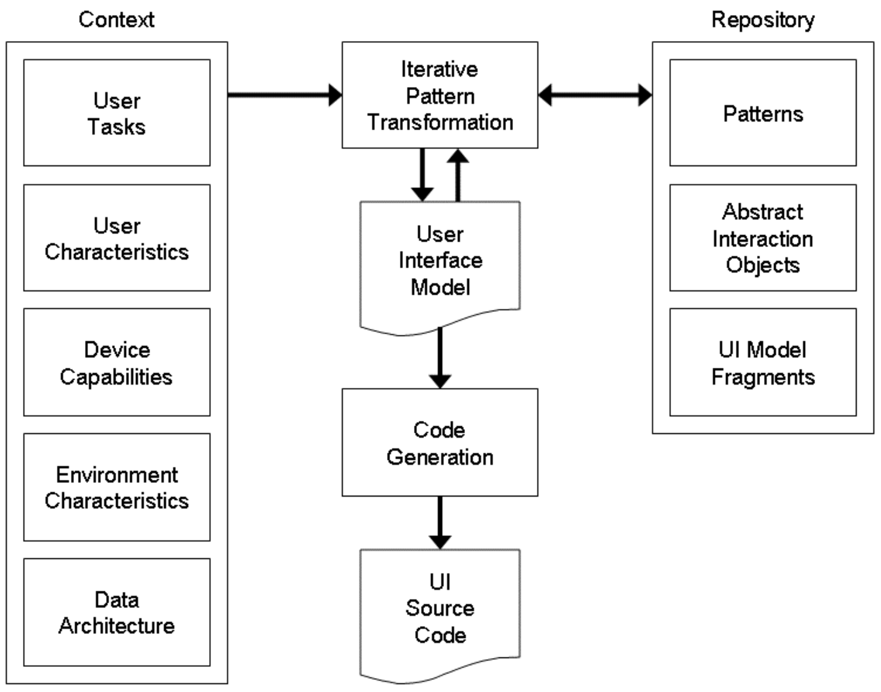


Figure 1 PaMGIS Functional Overview

As outlined in Figure 1 individual contexts of use can be defined through a task model, user model, device model, environment model and data architecture model. From this origin an abstract user interface model is compiled which is substantiated step by step via pattern transformation techniques until UI source code can be generated automatically. The current iterative pattern transformation process of the PaMGIS framework is solely engaged during design-time and does not allow for pattern-based UI modification during run-time.

In the p.i.t.c.h. project we did not exclusively use our own pattern languages for knowledge sharing systems and portable mobile applications (Engel et al., 2011; Kaelber & Märtin, 2011), but we made also use of foreign patterns. Predominantly we utilized patterns from Jenifer Tidwell (Tidwell, 2011), Douglas van Duyne (van Duyne et al., 2006), and Martijn van Welie (van Welie, 2012).

Table 1: Pattern description elements of the PaMGIS framework

PaMGIS pattern element	Brief description

UID
  PatternID
  InstanceID
	Unique identifier
	  Pattern identifier
	  Pattern instance identifier

Name	Name of the pattern
Alias	Alternative names; also known as
Illustration	Good example of instantiation of the pattern
Problem	Description of the problem to be solved
Context	Situations and circumstances in which the pattern can be applied
Forces	Description of forces in the environment that the use of the pattern will resolve
Solution	Description of how to resolve the problem
Synopsis	Summary of the pattern description
Diagram	Schematic visualization of the pattern

Evidence
  Example
  Rationale
	Verification that it is in fact a pattern by
	  at least three known uses of the pattern
	  discussion and any principled reasons

Confidence	Rating of how likely the pattern provides an invariant solution for the given problem
Literature	References to related documents or papers
Implementation	Code fragments or details of technical realization

Related-patterns
  LinkID
  Link-type
  Relationship-type
 	
  PatternID
  Revision
  InstanceID
  CollectionID
  Label
	Relationship to other patterns or pattern instances
	  Unique link identifier
	  Type of link (i.e. PERMANENT or TEMPORARY)
	  Type of relation (i.e. CONTAINS, INVOKES, EMPLOYS, IS-SIMILAR-TO)
	  Identifier of the referenced pattern
	  Revision of referenced pattern
	  Instance ID of referenced pattern
	  Identifier of the related pattern collection
	  Name of the pattern link

Management
  Authors
  Credits
  Creation-Date
  Last-modified
  Revision-number
	Authorship and change management
	  Name of the pattern author
	  Merits
	  Date of pattern compilation
	  Date of last change
	  Version of the pattern definition

The patterns stored inside the PaMGIS Pattern Repository are specified in an Extended Markup Language (XML) format. For this purpose we employed the Pattern Language Markup Language (PLML) version 1.1 (Fincher et al., 2003) and made enhancements and changes to achieve better support for semi-automatic pattern processing. Table 1 provides an overview of the PaMGIS pattern definition elements including brief descriptions.

PaMGIS pattern definitions basically differ from PLML 1.1 compliant specifications in terms of the *PatternID* and the *Related-pattern* attributes. For semi-automated processing it must be possible to model extensive pattern hierarchies where relations to multiple patterns can be expressed and particular patterns might occur more than one time, i.e. several pattern instances have to be distinguished. In addition there is a need to distinguish between two general types of pattern relations. On one hand there exist types of permanent links to other patterns which can be regarded as “hard-coded” and normally will not change for a long period of time. If such a permanent link is to be changed it would lead to a new revision of the particular pattern. As soon as a respective parent pattern is applied during the modeling process, all child patterns referenced by permanent links can be also applied automatically. On the other hand it is required to define temporary pattern relationships just when the pattern is being applied respectively the pattern hierarchy is constructed.

In order to meet these requirements we introduced a pattern description element named *UID* which combines the unique pattern identifier *PatternID* and the supplementary *InstanceID*. Within the element *Related-patterns* we hold information about link identifier and label, link and relationship types, and references to target patterns consisting of pattern identifier, revision number, instance identifier and collection identifier. The link type allows for distinguishing permanent and temporary links while the relationship type specifies the nature of the dependency between the particular patterns. As yet we have identified four different basic relation types including “contains”, “invokes”, “employs”, and “is-similar-to” relationships. Certainly there implicitly exist respective inverse relations, i.e. “is-contained-by”, “is-invoked-by”, and “is-employed-by”.

4 User Interface Modeling

The UI model of the knowledge sharing application consists of a hierarchy of components and patterns of various abstraction levels as illustrated in Table 2.

Table 2: Hierarchical structure of model components

Model component	Examples
Scenario patterns	Profile board, Info log, Collaborative reporting
Structural patterns	Orientation, Projects, Protagonists, Document library
HCI patterns	UX patterns, GUI patterns, media patterns, infrastructural patterns
Abstract interaction objects	Icons, Buttons, Links, List boxes, Combo boxes, Check boxes, etc.

Scenarios can be interpreted as patterns of the highest abstraction level, i.e. scenario patterns, which establish the root-level of the knowledge sharing pattern language. Such patterns can be refined by more concrete but still very abstract structural patterns. The structural patterns in turn are refined by HCI patterns with the help of UX patterns, GUI patterns and media patterns and linked to the domain content model by infrastructural patterns. On the lowest abstraction level interaction objects specify, how potential users of the application will interact with the knowledge sharing system. Within the UI model all relations between the various patterns of the different abstraction levels are managed.

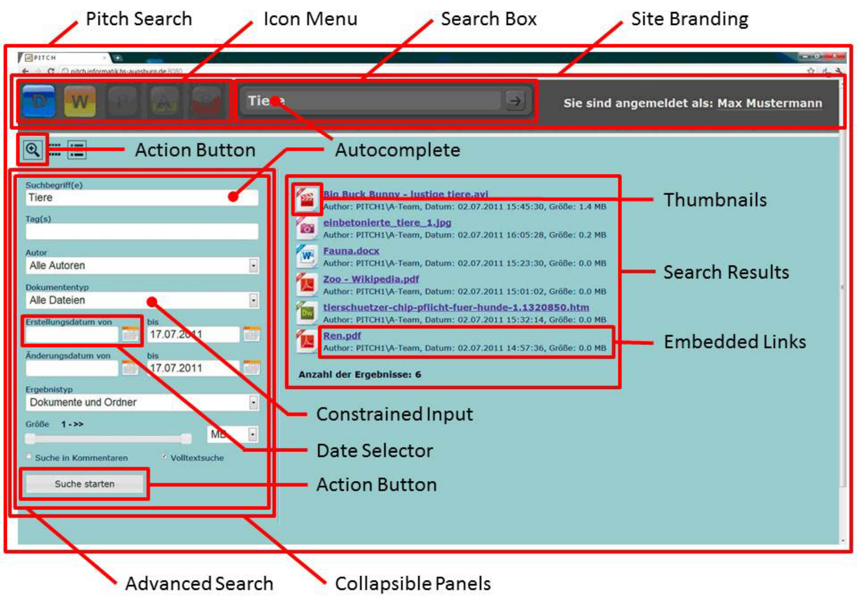
5 User Interface Model Transformation

The structure of the UI model and therefore the appearance of the user interface can be influenced and controlled by the definitions inside the various other models of the PaMGIS framework (Engel & Märtin, 2009), i.e. task model, user model, device model, environment model and data architecture model. Thus we have the option to create and adapt user interfaces for different contexts of use, for instance for specific user categories, e.g. novices, experts or impaired people, for different types of devices, e.g. desktop PC, tablet computers, mobile phones, and so forth. The settings in the models activate different sets of Pattern Transformation Patterns (PTP) that control which patterns on lower abstraction levels are used and how they are related to each other.

PTPs can be regarded as documentation of necessary information about the pattern transformations to be carried out. They are structured in a pattern-like manner. The most relevant description elements are *ID*, *Problem*, *Context*, *Solution* and *Illustration*. For examples, please refer to Table 3 and Table 4.

The focus of the p.i.t.c.h. platform is on the users and their work experience in collaborative knowledge sharing environment tools. For knowledge sharing applications we provide design rationales and use cases on the highest level of abstraction. For the creation of more specific task or sub-task models like e.g. search for project documents or check who is related to which project the CTTE editor (Paternò, 2001) is used. CTTE provides a graphical syntax for specifying activities and modeling task trees. By linking activities in the task model to the automation attributes of the semi-abstract and concrete HCI patterns and interaction objects of our pattern language, the mapping from scenarios, use cases and the resulting tasks to concrete user interfaces can be realized. Tasks and sub-tasks as well as the mappings to HCI patterns and interaction objects can later be reused for different knowledge sharing scenarios and use cases.

Figure 2 shows an exemplary screenshot of the search function of the p.i.t.c.h. desktop PC application. The red rectangles indicate the essential patterns which have been used to design the search capabilities.



Tiere

Sie sind angemeldet als: Max Mustermann

Suchbegriff(e): Tiere

Tag(s):

Autor: Alle Autoren

Dokumententyp: Alle Dateien

Erstellungsdatum von: bis 17.07.2011

Änderungsdatum von: bis 17.07.2011

Ergebnistyp: Dokumente und Ordner

Größe 1 - >> MB

Suche in Kommentaren ✓ Volltextsuche

Big Buck Bunny - lustige tiere.avi
Author: PITCH1\A-Team, Datum: 02.07.2011 15:45:30, Größe: 1.4 MB

einbetonierte_tiere_1.jpg
Author: PITCH1\A-Team, Datum: 02.07.2011 16:05:28, Größe: 0.2 MB

Fauna.docx
Author: PITCH1\A-Team, Datum: 02.07.2011 15:23:30, Größe: 0.0 MB

Zoo - Wikipedia.pdf
Author: PITCH1\A-Team, Datum: 02.07.2011 15:01:02, Größe: 0.0 MB

tierschuetzer-chip-pflicht-fuer-hunde-1.1320850.htm
Author: PITCH1\A-Team, Datum: 02.07.2011 15:32:14, Größe: 0.0 MB

Ren.pdf
Author: PITCH1\A-Team, Datum: 02.07.2011 14:57:36, Größe: 0.0 MB

Anzahl der Ergebnisse: 6

Figure 2 Screenshot of p.i.t.c.h. Search Functionality includiung pattern indications

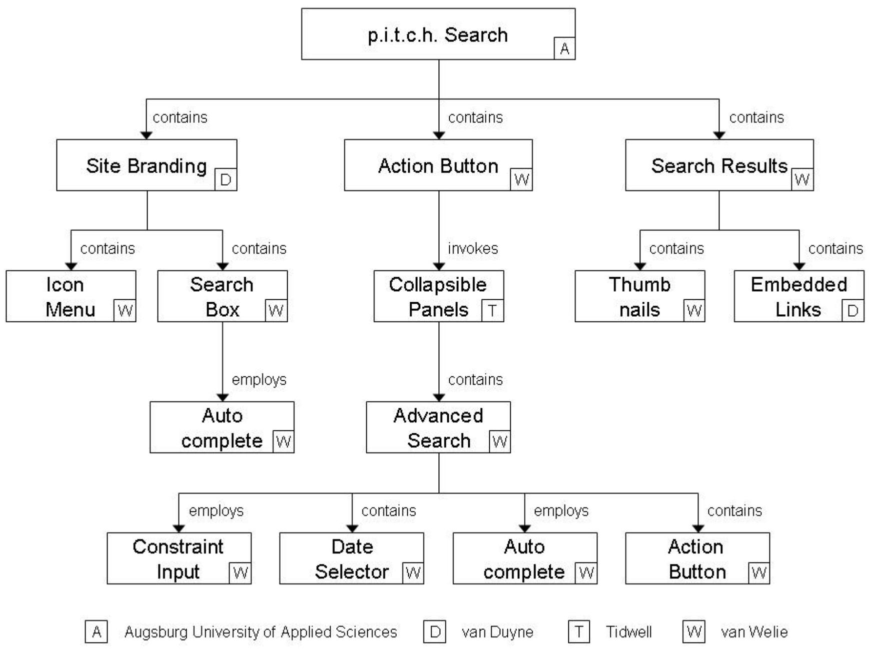


Figure 3 Excerpt of the pattern hierarchy of the p.i.t.c.h. Search Functionality

From a functional point of view the mobile phone user interface model is similar to the desktop PC model. But due to screen size limits the mapping on the HCI pattern level is different. While in the desktop PC version the entire search functionality and even the representation of the search results are displayed in one single window, the mobile variant requires a sequence of diverse small screens each providing a particular portion of the search feature. The initial search dialogue screen more or less consists merely of a "Search Box" pattern for specification of the search arguments and two buttons. The *Search* button triggers the actual search process while the *Filter* button leads to a subsequent screen with a menu consisting of several buttons. Each of these buttons in turn activates a new screen which allows for specifying certain filter values, e.g. to constrict the publishing date to a certain period in time. The *Start Search* button finally again leads to a separate screen which provides the search result in a condensed format. Figure 4 illustrates the screen flow for the mobile device.

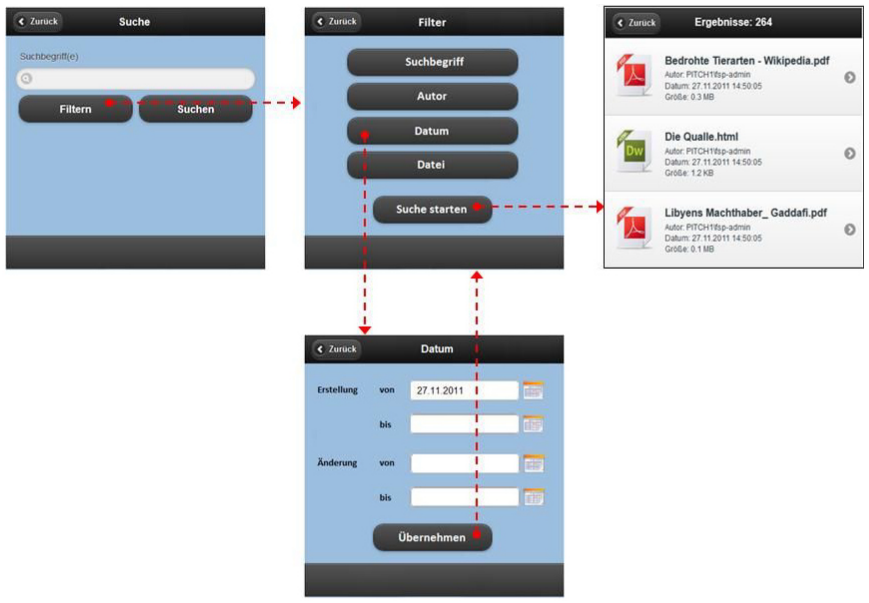


Figure 4 Screenshots of the p.i.t.c.h. Mobile Search functionality including screen sequence indications

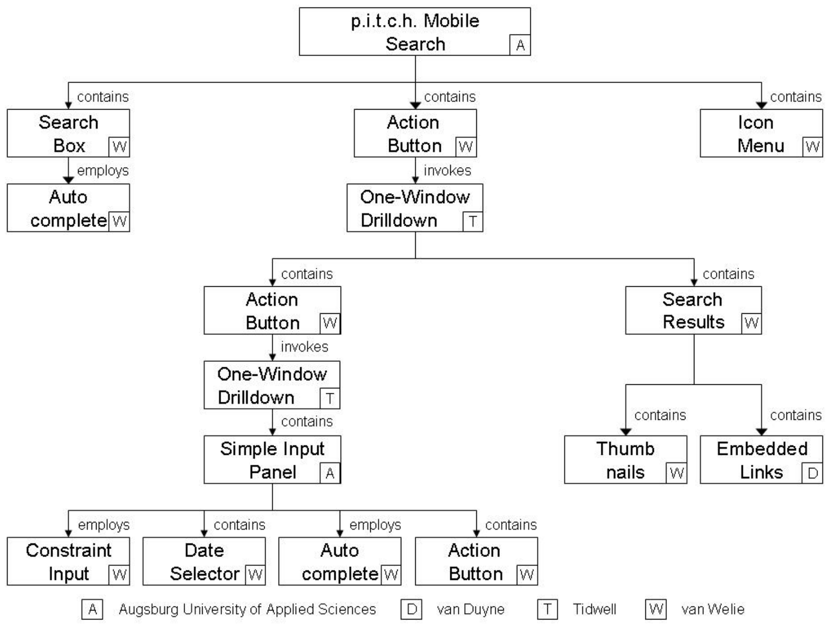


Figure 5
Excerpt of the pattern hierarchy of the “p.i.t.c.h. Mobile Search” functionality

Figure 5 shows an excerpt of the pattern hierarchy of the p.i.t.c.h. mobile search function. The screen transitions are represented by the "One-Window Drilldown" pattern (Tidwell, 2011) and the various screens for setting the filter values are constructed by means of our own "Simple Input Panel" pattern.

Starting with the desktop PC UI model it is possible to derive the mobile device UI model by applying PTP sets which lead to specific transformations and mappings of the various patterns throughout the entire pattern hierarchy. As an example Table 3 provides an excerpt of the PTP definition for the transformation of the "p.i.t.c.h. Search" pattern to be implemented on a mobile phone device. Due to screen size limits merely the "Search Box" pattern and two "Action Button" pattern instances are taken into account.

Table 3: PTP for the "p.i.t.c.h. Search" pattern.

Attribute	Description
ID	T-001
Problem	How to transform the p.i.t.c.h. Search pattern under the given context?

Context	• Source: desktop PC user interface
	  o Display size: 15 in. or bigger
	  o Display resolution: 1024 x 768 or better
	• Target: mobile device UI
	  o Display size: 35 x 41 mm or alike
	  o Display resolution: 176 x 208 px or alike

Solution	1. Apply the "Search Box" pattern. 2. Replace the "Collapsible Panels" pattern instances each by an "Action Button" pattern instance.
Illustration	Please refer to Figure 2 and Figure 4.

Another example is the transformation of the "Advanced Search" pattern. Again we have to find a way to consider that it is not possible to simply adopt the existing structure. Therefore we divide the various elements into groups and represent each group by a button. Clicking on such a button will lead to a new screen where the elements of the group will appear in order to specify the related filter values. An additional button will allow to save the changes and get back to the previous screen.

Table 4: PTP for the "Advanced Search" pattern.

Attribute	Description
ID	T-002
Problem	How to transform "Advanced Search" pattern under the given context?

Context	• Source: desktop PC user interface
	  o Display size: 15 in. or bigger
	  o Display resolution: 1024 x 768 or better
	• Target: mobile device UI
	  o Display size: 35 x 41 mm or alike
	  o Display resolution: 176 x 208 px or alike

Solution	1. Divide the existing patterns into reasonable groups. 2. For each group apply an "Action Button" pattern. 3. For each button apply a "One-Window Drilldown" pattern, a "Simple Input Panel" pattern, the patterns within the particular group and an extra "Action Button" pattern. 4. Apply an "Action Button" pattern (for starting the search process).
Illustration	Please refer to Figure 2 and Figure 4.

The pattern-based transformation can be accomplished step by step by traversing the source pattern hierarchy and applying related PTPs. For documentation and reuse purposes all specified PTPs are stored in the PaMGIS Pattern Repository.

6 Conclusion

In this paper we have summarized our pattern-based method to model and transform the user interface of the knowledge sharing application p.i.t.c.h. in order to accommodate it to different contexts of use. By providing several general categories of patterns for modeling the interactive target system mostly as a top-down process and by controlling the refinement process both, by the models of the PaMGIS framework and the PTPs, our approach combines high target system quality and usability, by applying established patterns, and user-, media- and context-specific individualization. At the same time, our approach is open for automating parts of the refinement process by integrating tools and components for pattern transformation and user interface generation.

As a next step in order to broaden our potential for automating the modeling and UI generation process, our research currently concentrates on the *Implementation* element of the PaMGIS pattern specifications. With it we intend to enrich the pattern definitions with code and model snippets exploitable by tools and user interface generators.

References

Alexander, C., Ishikawa, S. & Silverstein, M. (1977). *A pattern language: Towns, Buildings, Construction*, Oxford University Press.

Beyer, H. & Holtzblatt, K. (1999). Contextual Design, in: *Interactions*, 32–42

Engel, J. & Märtin, C. (2009). PaMGIS: A Framework for Pattern-based Modeling and Generation of Interactive Systems, in: *Proceedings of HCI International '09*. San Diego, USA., 826–835.

Engel, J., Märtin, C. & Forbrig, P. (2011): HCI Patterns as a Means to Transform Interactive User Interfaces to Diverse Contexts of Use, in: J. A. Jacko (Ed.): *Human-Computer Interaction, Part I, HCII 2011*, LNCS 6761, Springer-Verlag, Berlin, Heidelberg, 204–213.

Fincher, S. et al. (2003): Perspectives on HCI patterns: concepts and tools, In *CHI '03 extended abstracts on Human factors in computing systems*, Ft. Lauderdale, Florida, ACM, 1044–1045.

Gaffar, A. et al. (2004): Modeling patterns for task models, in: *TAMODIA '04 Proceedings of the 3rd annual conference on Task models and diagrams*, New York, ACM.

Gamma, E. et al. (1995): *Design Patterns. Elements of Reusable Object-Oriented Software*, Addison-Wesley, Reading, Mass.

Kaelber, C. & Märtin, C. (2011): From Structural Analysis to Scenarios and Patterns for Knowledge Sharing Applications, in: J. A. Jacko (Ed.): *Human-Computer Interaction, Part I, HCII 2011*, LNCS 6761, Springer-Verlag, Berlin, Heidelberg, 258–267.

Marcus, A. (2004): Patterns within Patterns, *Interactions*, vol. 11, Issue 2, 28–34.

Märtin, C., Engel, J. & Kaelber, C. & Werner, I. (2010). Using HCI-Patterns for Modeling and Design of Knowledge Sharing Systems, in: Forbrig, P. & Günther, H. (Eds.): *BIR 2010*, Springer LNBIP 64, 1–13.

Meixner, G., Paternò, F. & Vanderdonckt, J. (2011). Past, Present, and Future of Model-Based User Interface Development, in: *i-com*, 3/2011, 2–11

Paternò, F. (2001). *ConcurTaskTrees: An Engineered Approach to Model-based Design of Interactive Systems*, ISTI-C.N.R., Pisa.

Tidwell, J. (2011). *Designing Interfaces, Patterns for Effective Interaction Design*, 2nd Edition, O'Reilly Media Inc.

Tiedtke, T., Krach, T. & Märtin, C. (2005): Multi-Level Patterns for the Planes of User Experience. *Proc. of HCI International, July 22–27, Las Vegas, Nevada USA, Vol. 4 – Theories Models and Processes in HCI*, Lawrence Erlbaum Associates.

van Duyne, D., Landay, J. & Hong, J. (2006). *The Design of Sites, Patterns for Creating Winning Websites*, 2nd Edition, Prentice Hall International

van Welie, M. (2012). Patterns in Interaction Design, available at www.welie.com, Retrieved on Sept. 30, 2012.

Yoder, J. & Barcalow, J. (1997): Architectural patterns for enabling application security, in: *PLoP 1997*.