Position paper for CHI'2003 Workshop: "Perspectives on HCI Patterns: Concepts & Tools"

Just-UI: Using patterns as concepts for IU specification and code generation

Pedro J. Molina

CARE Technologies S.A.
Research & Development Department
Partida Madrigueres, 44.
03700 Denia, (Alicante), SPAIN

pjmolina@care-t.com

Javier Hernández

CARE Technologies S.A.
Research & Development Department
Partida Madrigueres, 44.
03700 Denia, (Alicante), SPAIN

jhernandez@care-t.com

Keywords: HCI Patterns, user interface specification, pattern supported tools, pattern use, conceptual patterns, patterns driven code generation.

1 INTRODUCTION

This position paper describes the approach taken by the authors related to HCI patterns considering both a theorical approach complemented with our experience in the field. Patterns are used in this approach as concepts or building blocks in a user interface specification model from a conceptual point of view. In such a model, each pattern provides a precise semantic for a problem identified in the problem space and possible solutions in the solution space. Some editors and code generator tools have been developed to implement the pattern supported method.

We will start providing a pattern definition. After that, we will discuss about how we search patterns, how we used them and, the tools implemented to support them. Eventually, conclusions and references are given.

2 PATTERN DEFINITION USED

A pattern can be seen as a *bean of experience*. It describes a well known problem with clues for detecting the problem in a given context, possible solutions to such a problem and related forces. In this way, classic and well known definitions of pattern like Alexander [1] or Coplien [2] are useful. Coplien [2] writes:

I could tell you how to make a dress by specifying the route of a scissors through a piece of cloth in terms of angles and lengths of cut. Or, I could give you a pattern. Reading the specification, you would have no idea what was being built or if you had built the right thing when you were finished. The pattern foreshadows the product: it is the rule for making the thing, but it is also, in many respects, the thing itself. J. Coplien, 1996

We think that this property *"predictability"* is essential for a good pattern. If a pattern is predictable, practitioners will realize very quickly if a pattern is suitable for a given context or not.

Furthermore, the frontier dividing the problem and solution spaces can be crossed using patterns. Used as *bridges,* patterns connect both worlds allowing to cross the former to reach the latter. Patterns connect different abstractions levels and can be used as reification mechanisms to cross from an abstract level to a more concrete one.

Finally, Vlissides [9] discuss about the necessity of using standard templates for describing patterns, recognizing that different context has different needs, and therefore, specialized description formats could be possible.

3 USE OF PATTERNS

In our daily job, we build tools for supporting the development of business applications following a software engineering point of view in order to improve the productivity, the quality and to save resources. In particular, we are concerned about the development of user interfaces involving classical phases as requirement elicitation, analysis, design, implementation and, maintenance.

Converse to design patterns [7, 10], we have developed a pattern language based on Conceptual User Interface Patterns [5, 6], increasing in this way, the abstraction level. Figure 1 shows the pattern language used and its relationships. These patterns are used in the conceptual or analysis phase before the design phase. However, design patterns can also be used in design models as a refinement of the abstract model to describe design decisions previously to code generation. Therefore, we are using a pattern supported development similar to the presented in PSA [4].

Based on the pattern language, we have build a model to describe abstract user interfaces, i.e., independent from design details, and therefore

useful for different target platforms like Desktop, Web or PDA UIs.

4 PATTERNS HUNTING

During the last five years, we have developed the pattern language to describe the problems we have found in our domain or context.

For us, the pattern hunting process is a slow searching task done after developing many UIs, identifying and abstracting recurrent problems and comparing with other existing and related patterns. After that, the pattern can be considered to be added to the pattern language or not. The main criterion used by us to solve such a question is to consider it as a cost/profit analysis: *How important is the problem solved by the pattern? How frequent is the occurrence of the pattern in a given domain? How the expressiveness power of the model is increased?*

Sometimes, the pattern identity is still unclear: *How to be sure if is this a pattern or not?* Some excellent hints for answering this question are provided by Winn & Calder [11].

Once a pattern is added into the pattern language, the full language must be reconsidered. Some patterns could be opposite to others or complementary, i.e., the relationships among patterns could be affected. Once added, the extended pattern language is used in real cases in order to test its usefulness. Iteration by iteration, and empirically verified, the patterns distilled in this way can be finally validated as useful for the given context (or domain).

5 THE DELTA (Δ) EFFECT

Seen as bridges, patterns allow us to cross abstraction levels: going down (*reification*) an also coming back (*reverse engineering*) supported by pattern matching. As reported in [5], we have experienced the following effect in the usage of patterns: A conceptual pattern helps to cross form requirements to analysis, a design pattern helps to cross form analysis to design, etc. To choose implies discarding other alternatives and therefore, constraining the specification/design/solution. Whenever a decision is taking, e.g. selecting a pattern, no matter the level we are, we have constrained the set of applicable solutions in the next levels. Such constraint has as a conical scope or *Δ effect* (see Figure 2): The question marks in Figure 2 represent questions made in the requirements phase. Such questions with its correspondent answers can instantiate a conceptual pattern in the analysis phase. The scope of such decision has a Δ-form triangle where parts of the subsequent phases are constrained or guided by such decision.

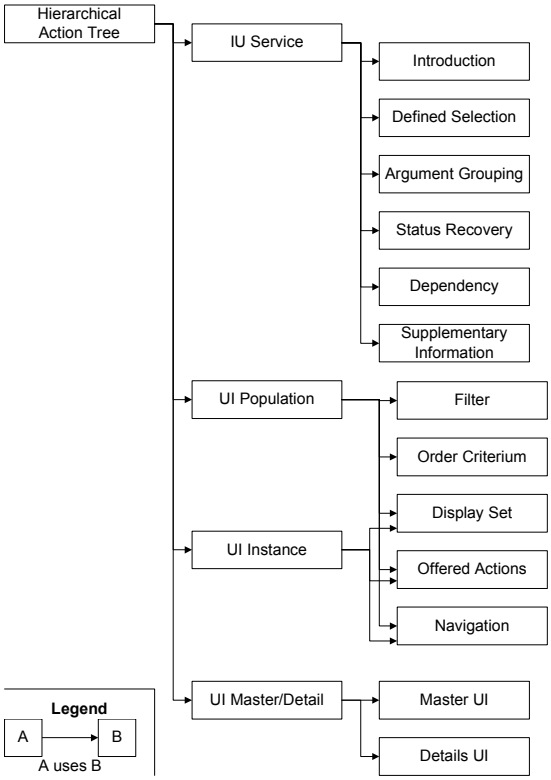

Figure 1. Pattern Language.

In this sense, decisions taken in upper levels (conceptual patterns selected at analysis level) can *guide or help* to take decisions in design levels where design patterns could be selected in order to delegate the implementation. A wizard can be provided to select patterns in the design phase guided from the information captured in previous phases. For example, a wizard could suggest the reification of the Conceptual Pattern *Master/Detail* [6] at the design level with the design pattern *Container Navigation* [7] or, alternatively, using the *Navigation Spaces* [10].

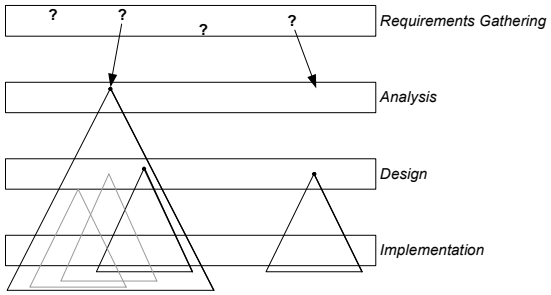

Figure 2. Delta (Δ) Effect.

6 PATTERNS SUPPORTED TOOLS

The described process has been tested in an industrial environment with success. The process of developing IUs has been improved in CARE Technologies S.A. using a pattern and model based approach. An editor for building such a model (OlivaNova Modeler®), a validator for checking

the correctness of models (embedded in the modeler) and a set code generators (OlivaNova Transformation Engines®) that translates the specifications into implementations for different target platforms such Desktop, Web or, PDA environments.

In the modeler tool, the analyst uses the patterns as building blocks to build the specification. Later on, the validator can check if the specification is valid, e.g., respect to a set of constrains imposed by the pattern composition language or detecting missing information. Finally, a Pattern Driven-Code Generator translates the patterns to a UI implemented in a particular environment. Each translator implements different solutions for each pattern due to the implementation and the design considerations are different from platform to platform.

To achieve the goal: multiple user interfaces from a unique specification, the patterns take into account structural, behavioral and semantic aspects at the same time. A pattern contains a family of possible parametric solutions, at least one solution for each target implementation environment. The parametric solution will be instantiated in a given context of use.

7 CONCLUSIONS

Based on our experience in the domain of developing UIs for business applications, patterns can be successfully exploited for increasing the abstraction level and providing reuse. Patterns are an effective way of communication of ideas shared by the developers and users as stated by Erikson (*lingua franca*) [3].

On the other hand, in our domain, pattern-driven code generation has been proven as a useful technique to implement UIs from an unambiguous pattern based specification and implementing reliable **Model Execution** tools.

However, the use of patterns for building systems is a clear empirical approach. For a well-defined domain, it is possible to specify and generate applications. Nevertheless, in open domains a new pattern or a new type of problem not yet classified could appear. This consideration justifies the search for extensibility features to properly deal with these new requirements.

As desired property in a pattern tool, extensibility could be excellent. However, in the domain described, it is difficult to achieve due to the need of having a precise frame-work to describe the patterns in a formal language (natural language it is not enough) in terms of structure, behavior, consequences, trade-offs, etc. at the same time.

Interoperability can be achieved in general purpose patterns builders/explorers tools using standardized XML DTD or Schemas following a given pattern format description. And it will be very valuable to have tools for describing patterns in a normalized way [8].

Easy of use is another desired property. Patterns languages must be comprehensible for users; otherwise they will not use the pattern language at all.

To sum up, IU patterns can improve the common knowledge of the concepts involved in a domain. They can be used as a common language for users, analysts and developers, and used for learning about the domain. Also, as shown in this position paper, the can be used for formal specification and code generation of user interfaces.

8 REFERENCES

[1] Alexander C. *The Timeless Way of Building*. Oxford University Press, New York, 2000.

[2] Coplein J.O., *Software Patterns*, SIGS Books & Multimedia, New York, USA, 1996.

[3] Erickson T. *Patterns Languages as Languages*. CHI'2000 Workshop: Pattern Languages for Interaction Design, 2000.

[4] Granlund Å., Lafrenière D. *A Pattern-Supported Approach to the User Interface*, Design, In Proceedings of HCI International 2001, 9th, International Conference on Human-Computer Interaction, pages, 282-286, New Orleans, USA, August, 2001. Also available at http://www.sm.luth.se/csee/csn/publications/HCIInt2001Final.pdf

[5] Molina P.J., Meliá S., Pastor O. *User Interface Conceptual Patterns.* In Proceedings of Design, Specification Verification of Information Systems, DSV-IS 2002, June, 2002. Also in Lecture Notes in Computer Sciences, vol. 2545, Springer Verlag, 2002.

[6] Molina P.J., Meliá S., Pastor O. *Just-UI: A Model for User Interface Specification.* In Proceedings of CADUI'2002, Valenciennes, France, Kluwer Academics, May, 2002.

[7] Hallvard Trætteberg, *Model based design patterns*, (position paper) Workshop on User Interface Design Patterns, CHI'2000, The Netherlands, 2000.

[8] Sharon Greene, Paul M. Matchen, Lauretta Jones. (position paper) CHI'2002 Workshop: Patterns in Practice: A Workshop for UI Designers, 2000.

[9] Vlissides J., Seven Habits of Successful Pattern Writers, *C++ Report*, November/December, 1996. Also available at http://www.research.ibm.com/designpatterns/pubs/7habits.html

[10] Weile M. van, Veer G. van der, Eliëns, A. *Patterns as Tools for User Interface Design.* http://www.cs.vu.nl/~martijn/gta/docs/TWG200.pdf. 2000

[11] Winn T., Calder P. Is This a Pattern?, *IEEE Software*, pages 59-65, January/february, 2002.