Support for the definition and usage of process patterns

Mariele Hagen (hagen@adesso.de)

adesso AG, Stockholmer Allee 24, 44269 Dortmund, Germany

Introduction

Since the 1980s software patterns have gained rising interest within the software community. While in the last 20 years the focus has been on mining and describing single patterns, recent focus has shifted to describing more complex structures like pattern catalogues [GHJ96], systems [BMR96], languages [AIS77] or handbooks [RZ96]. The variety of pattern types has increased, too, as there are design patterns, process patterns, organizational patterns, pedagogical patterns and so on. Despite this increasing attention patterns of all types bear shortcomings with respect to their description. We will explain these deficiencies with respect to process patterns, but these deficiencies are valid for any pattern type.

Ambiguity because of lacking precision

Patterns – also called a “literary form” [Cop96] - are mostly described in an informal way by natural language. This can be considered as an advantage, since understanding a pattern does not require the knowledge about notation semantics or a certain syntax. However, there is a limitation to precision in natural language. Eden examined the semantic ambiguity of Gamma’s design patterns and revealed vast deficiencies concerning precision [Ede97]. This informal description of a pattern allows for an ambiguous interpretation and execution of a pattern’s process. It is also not known, how a pattern can be composed of other patterns, under which conditions a pattern is a variant of another pattern and in which cases patterns can be executed sequentially. Consequently, in many cases maybe not the most adequate pattern is chosen.

Ambiguity because of non-standard description of pattern interfaces and pattern relationships

It is widely accepted that patterns should not be considered as isolated solutions, but be a part of a more complex structure (like pattern languages, catalogues, handbooks or systems) to “achieve their fullest power” [Cop96]. This requirement is important especially for process patterns. It is necessary to know, which patterns might work together or even depend on each other to build up a software process. We need to know the entry and exit conditions (i.e. the interfaces) of a process pattern to glue it together with other process patterns. Present process pattern descriptions contain textual context definitions, but they are not accurately standardized described.

In addition to a more accurate context definition, pattern relationships have to be defined more precisely. Although several publications bother with pattern relationships, they provide mostly a textual, nonformal and unprecise description like “A variant pattern refines a more well-known pattern” [Nob98]. Relationships defined without precise criteria are questionable as they do not give reliable implications for their usage.

Lacking process pattern management and tool support

Existing literature on patterns does not focus the problem of process pattern management (i.e. pattern writing, problem specification, pattern search, pattern selection and pattern application) in detail (cf. [Czi01] for the microprocess of patterns). But we think that for an effective and productive use of process patterns the process pattern management must get more into focus. First, pattern authors or miners have to be equipped with a standard notation to specify patterns in a unambiguous, precise way. This notation must allow to specify relationships as composition, variance and sequence and to specify interfaces. Users of a pattern structure (pattern language or the like) should be supported in providing mechanisms for pattern search. Secondly, patterns could be retrieved and used more effectively and in a more adequate way if an information system like a pattern workbench would present not only a pattern candidate but also possible variant patterns, successors and component patterns which compose the

candidate pattern. There should also be the possibility to log the application of process patterns to understand a project's history.

Key Ideas

To overcome the mentioned deficiencies, a more precise specification of patterns is needed. With respect to this objective, we defined a process pattern description language (PPDL) based on the UML [Dit02]. We added several new concepts referring to the UML`s syntax and semantics to provide mechanisms to specify the structure of a process pattern, the interfaces (i.e. contexts) and relationships between patterns. To specify the structure of process patterns, we first had to define an adapted pattern form which contains all elements to model a process. For example, we renamed the pattern elements Applicapability with Initial Context and Consequences with Resulting Context and added a new pattern element named roles (cf. [GHJ96] for an overlook of the GOF pattern elements). Secondly, we defined the process pattern relationships Sequence, Usage, Refinement, ProcessVariance, ProcessAlternatives and ProblemVariance. In addition, we defined a notation to present these new language concepts. We extended the UML meta-model with several meta-classes and added several OCL-Constraints to obtained a syntactically and semantically enriched UML derivate, the process pattern description language. Currently, we are working on a process pattern workbench that implements the process pattern description language and that provides all the necessary mechanisms to support the definition, modification, retrieval and application of process patterns.

Conclusion

We have shown that current (process) pattern descriptions are unprecise and therefore ambiguous. This ambiguity prevents an effective and productive use of process patterns and process pattern languages (or systems, handbooks and catalogues respectively). So, our aim is to improve understanding and use of process patterns and process pattern languages by defining a process pattern description language which possesses the required precision and unambiguity. By developing the process pattern workbench we want to implement the introduced concepts and to support the everyday work of authors and readers of process patterns.

References

[AIS77] Alexander, C.; Ishikawa, S.; Silverstein, M.: A Pattern Language. New York: Oxford University Press, 1977.

[BMR96] Buschmann, F.; Meunier, R., Rohnert, H. et. al.: Pattern-Oriented Software Architecture - A System of Patterns. Wiley & Sons Ltd., 1996.

[Cop96] Coplien, J.: Software Patterns. SIGS Book & Multimedia, 1996.

[Czi01] Czichy, T.: Pattern-based Software Development: An Empirical Study - Summary of Results, University of Technology, Dresden, Department of Systems Engineering, 2001.

[Dit02] Dittmann, T.: PPDL – Eine Beschreibungssprache für Process Patterns, 2002, Universität Dortmund

[Ede97] Eden, A.: Giving The Quality a Name: Precise Specification of Design Patterns: A Second Look at the Manuscripts. In: Journal of Object Oriented Programming, SIGS Publications, http://www.math.tau.ac.il/~eden/bibliography.html#giving_the_quality_a_name, May 1997.

[GHJ96] Gamma, E.; Helm, R.; Johnson, R. et. al.: Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software, Addison-Wesley, 1996.

[Nob98] Noble, J.: Classifying Relationships Between Object-Oriented Patterns, Microsoft Research Institute, 1998

[RZ96] Riehle, D.; Züllighoven, H.: Understanding and Using Patterns in Software Development, Theory and Practice of Object Systems, Vol. 2(1), pp. 3-13.