Patterns Meta-Specification and Cataloging: Towards Knowledge Management in Software Engineering

León Welicki  
Faculty of Computer Science  
Universidad Pontificia de Salamanca, campus Madrid  
Paseo Juan XXIII, 3  
28040, Madrid Spain  
lwelicki@acm.org

Juan Manuel Cueva Lovelle  
Faculty of Computer Science  
Universidad de Oviedo  
Calvo Sotelo s/n  
33007, Oviedo, Spain  
cueva@lsi.uniovi.es

Luis Joyanes Aguilar  
Faculty of Computer Science  
Universidad Pontificia de Salamanca, campus Madrid  
Paseo Juan XXIII, 3  
28040, Madrid, Spain  
luis.joyanes@upsam.net

Abstract
Writing patterns is a very important task for leveraging knowledge within an organization or in the software engineering community as a whole. Patterns are more than text, diagrams or source code. Patterns are knowledge that comes from experience. Sharing patterns is sharing knowledge. The creation of a language for pattern meta-specification and a catalog of patterns from different pattern languages described using this language is a clear step towards managing software engineering knowledge. The creation of a web-based visualization tool for the catalog makes this knowledge available to the world, allowing using, searching, linking, and discussing the patterns in the catalog.

Categories and Subject Descriptors D.2.2 [Software Engineering]: Miscellaneous – reusable software, D.3.2 Language Classifications – Very high-level languages, H.3.m Information Storage and Retrieval - miscellaneous

General Terms Documentation, Design, Human Factors, Standardization, Languages, Theory.

Keywords Patterns, Knowledge Management, Software Engineering, Meta-Specification, Cataloging, Design Patterns

1. Patterns and Knowledge Management
Patterns are explicit forms of software engineering knowledge. Sharing patterns is sharing knowledge [1]. Patterns are a great way for sharing software engineer’s experience. Patterns also help to enhance the vocabulary of their users [5], allowing shifting the abstraction level of the discussions (for instance, design patterns shift the level of design abstraction from classes to collaborations between classes [5]). There is a very interesting convergence point between applied knowledge management and patterns that can be expressed with the following analogy: in [3] a model for creating value from knowledge management in japanese companies is presented. In [5] a tool for text editing with advanced features (the Lexi text editor) is created composing patterns. In both cases value is retained in it.

2. If We Knew What We Know
There is a very famous phrase in the knowledge management world attributed to Lewis Platt, a former CEO at Hewlett-Packard: “If HP knew what HP knows, it would be three times more profitable” [4]. Patterns help to codify and to formalize knowledge, making it shareable within an organization or the whole industry. Patterns are not restricted to object oriented design or algorithms. They exist in a very wide variety of domains (as an example of this variety, in [7] a pattern language for introducing new ideas in organizations is presented).

Patterns can help an organization to codify its tacit knowledge into concrete forms of explicit knowledge that can be shared and reused within the organization or the whole industry. Pattern languages can do the same but at a larger scale, providing a framework for organizing and classifying groups of patterns.

3. The Patterns Life Cycle
Patterns emerge from experience. They have a life-cycle that starts with tacit knowledge living in the head of a person or a group and ends with a shareable explicit description of this knowledge. It is very important to stress the importance of this process, since tacit knowledge only lives in the head of a person or a group and when these individuals live an organization that knowledge may not be retained in it.

The patterns life-cycle goes through the following steps:

- **Knowledge and Experience:** the pattern is just tacit knowledge that comes from experience stored in the head of an individual or group.
- **Discovering and Formalization:** a pattern starts to emerge. The involved individuals document it. It also can be discovered in existing systems doing pattern mining. The final product of this step is an unrevised description of a pattern.
- **Discussion and Refinement:** patterns are discussed within the software community. PLoP conferences series are a great forum for discussing and refining patterns and pattern languages.
- **Publishing and Sharing:** the refined pattern is published and shared within the software community (it may be published in books, journals, proceedings, web, etc.).

4. Patterns Are Not Enough
A pattern may not be enough to share knowledge. Patterns do not exist in isolation and therefore more than patterns may be needed to effectively express an idea. In the same way that an implementation is not enough for expressing a pattern, a pattern itself may not be enough to convey an idea without any supporting concept. For instance, the “Dependency Inversion
Pattern meta-specification language assists pattern authors to formalize their tacit knowledge in a standardized shareable form. EML (Entity Meta-specification Language): modularly composable DSL [8] based on XML created with the purpose of describing all kinds of software patterns and supporting concepts. It includes language constructs that allow the abstract representation of the description and behavior of an entity. It also allows describing relationships between entities.

Patterns Catalog: catalog of representative patterns, written using EML. It also contains supporting concepts (pattern languages, sources of information, types of patterns, design concepts, etc.), also described using EML. It exposes its contents through a service interface. The catalog provides the entire necessary infrastructure for storing and sharing entities (addressing, versioning, searching, linking, registering, storage, subscription, syndication, etc.).

Catalog Browser: visualization and navigation tool for the catalog. It supports multiple views on a pattern. Includes community features that allow discussing on a pattern.

6. Conclusions
The creation of a meta-specification language for pattern description, a cataloging infrastructure, and a catalog of concepts and patterns (including all the relationships existing between them) described with the meta-specification language is a clear step towards gathering, managing and sharing software engineering knowledge. The creation of a web-based catalog visualization tool makes this knowledge available to the world.

These technologies support and change the patterns life-cycle. The pattern meta-specification language assists pattern authors to formalize their tacit knowledge in a standardized shareable fashion. The catalog helps them to organize and classify their patterns, relate them with other entities, and share them with other users. The pattern browser provides community features that allow discussing on a pattern in order to refine or evolve it. This changes the patterns life-cycle, making it more dynamic and interactive: once a description of pattern exists, it can be shared with the world. Discussion is no longer a fixed step in the middle of the cycle since the visualization tool has mechanisms to discuss any pattern in the catalog at any stage in the cycle. Publishing is neither a static step at the end of the cycle anymore. Authors can refine their patterns using as an input comments of users of the entire world or new experiences. The new cycle is more dynamic, supporting constant pattern refinement and evolution.

Sharing patterns is sharing knowledge, but not any knowledge...Patterns convey applied knowledge that comes from experience. The tools presented in this paper help to create value sharing software engineering knowledge in form of patterns.

References