Knowledge Patterns

Jörg Rech
Fraunhofer Institute for Experimental Software Engineering, Germany

Raimund L. Feldmann
Fraunhofer Center for Experimental Software Engineering, USA

Eric Ras
Fraunhofer Institute for Experimental Software Engineering, Germany

INTRODUCTION

Knowledge is one of the most important assets for any kind of organization, and for all areas of science. While *experience* describes events in one specific context that can only be reused carefully, *knowledge* is usually applicable in previously unknown contexts with a fair amount of certainty. To support practitioners and researchers in their knowledge management (KM) activities, the concept of knowledge patterns can be used. Knowledge patterns are one way to formalize and describe lessons learned and best practices (i.e., proven experiences) about structuring knowledge, the design of KM systems, or the development of underlying ontologies. Such patterns capture aspects that positively or negatively influence the KM activities. In the later case, where negative influences are described, such patterns are denoted as anti-patterns. Knowledge patterns and anti patterns can help in developing KM systems and improve the quality of the systems themselves as well as that of the knowledge within (i.e., the quality of the knowledge). Thereby, patterns in KM represent a way of structuring knowledge as well as a form of language that helps knowledge engineers to communicate about knowledge and KM systems.

BACKGROUND

Knowledge is one of the most important assets for any kind of organization, and for all areas of science. While *experiences* describe events in one specific context that can only be reused carefully, *knowledge* is usually applicable in previously unknown contexts with a fair amount of certainty. Unfortunately, a small number of experts who have acquired knowledge through their experiences in day-to-day work hold major parts of the knowledge in an organization. Surprisingly, this is equally true for researchers in KM. Experiences gained regarding knowledge itself and KM systems, either technical, social, or socio-technical ones, are typically recorded in the form of models or process models only. Fine-grained knowledge about the structuring, interconnection, or classification of knowledge is rarely documented, and common and recurring patterns are hardly available – while best practices regarding the KM systems as a whole and initiatives are often shared (Davenport & Probst, 2000; Mertins, 2003).

Such knowledge about KM systems is documented in the form of success factors (Mathi, 2004) (Thomas, 2006) (Morisio et al., 2002), success models (Jennex & Olfman, 2004, 2006), success measures (Jen & Yu, 2006), reference architectures for KM systems (Davenport & Probst, 2000; Mertins, 2003), worst practices (Fahey & Prusak, 1998), barriers (Eberle, 2003), facilitators (Damodaran & Olphert, 2000), and incentives (Feurstein et al., 2001), which are often described in an unstructured and informal way. They typically preserve knowledge about a whole KM system or initiative. Barriers, facilitators, or incentives represent types of patterns that describe common and recurring incidents, practices, or behavioral structures in KM. There are many different types of barriers, such as knowledge barriers in general (Riege, 2005), barriers in knowledge transfer (Sun & Scott, 2005) and distribution (Bick et al., 2003), barriers based on culture (Wolf & Wunram, 2003), as well as barriers based on roles and activities (Awazu, 2004).

In software reuse, several barriers were described by Judicibus and classified into the two classes “individual factors” and “collective factors” (Judicibus, 1996), such as the “Feudal Lord’s Syndrome” or the “Egghead’s Syndrome” (Rech, Decker et al., 2007).

In software engineering, design patterns are a relatively new concept, which was transferred from architecture to represent typical and recurring patterns of good and bad software architectures. These design patterns (Gamma et al., 1994) and anti-patterns (Brown et al., 1998) were the starting point for the description of many patterns in diverse software phases and products. Today, we have thousands of patterns (Rising, 2000) for topics such as software reuse (Long, 2001), agile software projects (Andrea et al., 2002) or pedagogical science

(http://www.pedagogicalpatterns.org/) (Abreu, 1997; Fincher & Utting, 2002). Many other patterns are stored in pattern repositories such as the Portland pattern repository (PPR, 2005) or the Hillside pattern library (HPL, 2005) and are continuously expanded by conferences such as PLOP (Pattern Languages of Programming; see http://hillside.net/conferences/).

While there are similar concepts, the idea of patterns is relatively new in KM. Nevertheless, the concept of patterns and anti-patterns helps in documenting knowledge and experiences.

PATTERNS AND ANTI-PATTERNS IN KNOWLEDGE MANAGEMENT

The quality of the knowledge gained, the technical KM system used, or the social KM method applied is neither easy to be evaluated, nor is it easy to be improved. This is partly due to the fact, that there exists no universal KM system, which is suitable for all kinds of organizations, or universal knowledge, which is suitable in all situations. In practice, each system, as well as the knowledge within, has to be adapted and tailored to the individual needs of an organization and the people within.

In order to tailor knowledge and KM systems to the specific needs at hand, we can resort to the concept software design patterns, which are used to structure and adapt software systems to the needs of the customers. By transferring the concept of software design patterns to knowledge management, we obtain the following definition for knowledge and knowledge management patterns:

> **Knowledge Pattern**: A knowledge pattern is a general, proven, and beneficial solution to a common, recurring problem in knowledge design, i.e., the structuring and composition of the knowledge (e.g., on or via Wiki pages) or the ontology defining metadata and potential relationships between knowledge components.

In general, anti-patterns are the opposite of patterns and represent worst practices that should not be applied. This leads to the following definition:

> **Knowledge Anti-Pattern**: A knowledge anti-pattern is a general, proven, and non-beneficial problem (i.e., bad solution) in a knowledge product, system, or process, i.e., the structuring and composition of the knowledge (e.g., on or via Wiki pages) or the ontology defining metadata and potential relationships between knowledge components.

Knowledge Pattern Classification

In order to group and delimitate the patterns, one can classify them in the following seven groups ranging from patterns for knowledge content via patterns for KM systems to patterns about social KM-oriented systems.

- *Knowledge Content Patterns & Anti-Patterns* apply to the content of a knowledge component as well as the semantic relations between components. Typically, they are perceived from the viewpoint of the reader or writer.
- *Knowledge Usage Patterns & Anti-Patterns* apply to the use and maintenance of knowledge components (i.e., the whole lifecycle) and are typically perceived from the view of the knowledge maintainer or gardener.
- *Knowledge Ontology Patterns & Anti-Patterns* apply to the ontologies used to structure knowledge components and are, typically, perceived from the viewpoint of the ontology developer.
- *Knowledge Presentation Patterns & Anti-Patterns* apply to the presentation of knowledge components or elements in a KM system and are typically perceived from the viewpoint of the reader or writer.
- *Knowledge Transfer Patterns & Anti-Patterns* apply to the transfer (i.e., reading, understanding, and application) of knowledge components or elements and are perceived from the viewpoint of the reader or writer.
- *KM Systems Organization Patterns & Anti-Patterns* apply to organizational aspects of the technical infrastructure of KM systems such as Wikis and are typically perceived from the viewpoint of a KM system (e.g., Wiki) developer or administrator.
- *Social KM Patterns & Anti-Patterns* apply to the social system or purely human-based part of KM systems and are perceived from the viewpoint of a KM system (e.g., a Wiki) developer or administrator.

Knowledge Pattern Template

For describing (anti-)patterns, one can use the following short template, which was derived from more elaborate templates. It consists of the following sections:

- *Name*: What is the (anti-)pattern called?
- *Issue*: What is the issue (e.g., problem) addressed by this (anti-)pattern?
- *Q-Effect*: What "knowledge quality aspects" are affected the most by this (anti-) pattern? This section clearly states if there is a positive (+), negative (-), or neutral (0) effect.
- *Solution*: What are the principal solutions underlying this pattern? Multiple alternative solutions might be given to remove an anti-pattern or build a pattern. In this section, "knowledge refactorings" are cited, which are described in more detail in (Rech, Decker et al., 2007).
- *Causes*: What are the basic causes of this (anti-) pattern?

The full template format for describing these patterns consists of additional information entities such as structure, dynamics, anecdotal evidence, example, or exceptions.

Example of Knowledge Patterns

Knowledge patterns should be based upon experiences from multiple KM initiatives. Several identified patterns and anti-patterns for knowledge and KM systems are listed in (Rech, Decker et al., 2007). An example of a Knowledge Transfer Pattern is described in the following table:

	Notification Pattern	
Issue	In a static knowledge base, the user is not informed about changes to knowledge components he has written, used before, or needs on a daily basis. The user is not informed about updates to his contributions or to knowledge components he is currently using (e.g., in a project).	
Q-Effect	Functionality	+
	Reliability	+
	Usability	+
	Efficiency	0
	Maintainability	+
	Portability	0
Solution	Monitor Knowledge: A user should be able to monitor pages and be notified if changes are made to a knowledge component (especially if he is the author). Monitor Ontology: A user should be able to monitor part of the ontology and be notified if changes are made to it (e.g., to reclassify his own pages).	
Causes	The observation by automated notification helps to keep up to date with a knowledge component, especially if the knowledge is currently still used in a project.	

While patterns describe one solution to multiple or unspecified problems, antipatterns describe one problem with multiple solutions. An example of a Knowledge Content antipattern is described in the following table:

	Knowledge Blob Anti-Pattern	
Issue	The description of an experience or knowledge component gets larger and larger over time and subsumes more and more information. The search for an arbitrary knowledge component will often include the knowledge blob. The knowledge blob can be used for different problems, has multiple solutions, or contact data.	
Q-Effect	Functionality	-
	Reliability	0
	Usability	-
	Efficiency	-
	Maintainability	-
	Portability	-
Solution	*Compact Knowledge*: Summarize and rewrite the knowledge in a shorter form (e.g., on one page). *Extract Elements*: Apply divide & conquer to create several mutually exclusive pages with parts of the original page. *Extract Commonalities*: Find elements in other pages with overlapping knowledge and extract this overlapping element from both (or all) pages into a new page.	
Causes	The KM system makes it easy to find and change (e.g., extend) a knowledge component, the users are not sensitized to create individual experiences, or there is no maintenance or change process established for the knowledge in the KM system.	

FUTURE TRENDS

While knowledge patterns enable new ways of capturing, processing, presenting, sharing, and distributing knowledge, many challenges and possibilities for further research exist. Currently, we have identified the following challenges in the context of knowledge patterns:

Collection of Knowledge Patterns: Design patterns are well-known throughout the software engineering community; however, the use and collection of patterns regarding knowledge, KM, or KM systems is rare. Still, the collection of these knowledge patterns (Rech, Decker et al., 2007) (Rech, Feldmann et al., 2007) and knowledge refactorings has the potential of systematizing LSOs and improving the collected experiences.

Development of a Knowledge Pattern Ontology: As more and more patterns and antipatterns are described, a systematic classification and formalization of these is needed. Ontologies can be used to structure patterns and capture the interrelationships of these patterns, similar to the "Web of Patterns" (Dietrich & Elgar, 2005), the "Object-oriented Design Knowledge Ontology" (Garzás & Piattini, 2005), or ontology-based pattern languages (Henninger & Ashokkumar, 2006). Furthermore, in order to support the dissemination of patterns and their use in education, information is required regarding their use and acceptance by and their usefulness for practitioners.

Aggregation of Experiences and Observations to Knowledge Patterns: One common approach to aggregating quantitative experiences (such as empirical studies) is the meta-analysis technique (Brooks, 1997). For the aggregation of qualitative experiences (such as problem solution descriptions), ad-hoc techniques – if any at all – are used. While some approaches have been described for aggregating "knowledge dust to pearls" (Basili et al., 2001) or for aggregating observations into reusable experiences, patterns, and finally laws (Rech & Ras, 2007), more systematic and automated approaches are required in order to aggregate experiences into more generally applicable patterns and laws.

Collaborative authoring of knowledge patterns: The documentation of experiences and knowledge by an individual always holds the risk that errors are made or that important information is missing. Collaborative authoring or rework approaches for experiences using Wikis (Ras et al., 2008) or writer's workshops (Gabriel, 2002) have the benefit of many eyes checking the documented experience (similar to pair programming). However, motivating and to convincing users to contribute and to make use of the EB is not an easy task. Sharing knowledge can be perceived not entailing many advantages for oneself and even dangerous. Additionally, contributing to the EB is time- and effort-consuming. There is a certain resistance that has to be coped with.

CONCLUSION

Knowledge management or software reuse in learning (software) organizations is often accompanied by poor quality of the knowledge, experiences, or decisions within a KM system as well as the quality of the KM system itself.

Knowledge patterns represent an approach to structuring knowledge in KM systems. These patterns and anti-patterns can be used to develop KM systems and improve the quality of the systems themselves as well as that of the knowledge within (i.e., the quality of the knowledge).

Knowledge patterns will, hopefully, stimulate discussion about the meaning of quality in the context of KM, about how knowledge should (and should not) be described in a KM system, and what is needed to generate a fruitful socio-technical KM system. Readers are encouraged to start writing their own patterns and share them with other users (e.g., by using the following website http://www.knowledgepatterns.eu). Such active knowledge sharing helps to make the knowledge stored in the semi-formal patterns become applicable in other contexts. Applications in a non-SE context might help to increase the experience regarding a specific topic and enable researchers and practitioners alike to further generalize the pattern concept.

REFERENCES

Abreu, F. B. E. (1997). Pedagogical patterns: picking up the design patterns approach. *Object Expert, UK * vol 2 (March April 1997), no 3, p 37, 41, 3 refs.*

Andrea, J., Meszaros, G., & Smith, S. (2002). *Catalog of XP Project 'Smells'*. Paper presented at the 3rd International Conference on XP and Agile Processes in Software Engineering (XP 2002), Alghero, Sardinia, Italy.

Awazu, Y. (2004). *Knowledge management in distributed environments: roles of informal network players.* Paper presented at the 37th Annual Hawaii International Conference on System Sciences.

Basili, V. R., Costa, P., Lindvall, M., Mendonca, M., Seaman, C., Tesoriero, R., & Zelkowitz, M. (2001). *An experience management system for a software engineering research organization.* Paper presented at the 26th Annual NASA Goddard Software Engineering Workshop.

Bick, M., Hanke, T., & Adelsberger, H. H. (2003). A process-oriented analysis of knowledge distribution barriers. *Industrie Management, 19*(3), 37-40.

Brooks, A. (1997). Meta analysis: A Silver Bullet for Meta-Analysts. *Journal of Empirical Software Engineering, 2*(4), 333 - 338.

Brown, W. J., Malveau, R. C., McCormick, H. W., & Mowbray, T. J. (1998). *AntiPatterns: refactoring software, architectures, and projects in crisis*. New York: John Wiley & Sons, Inc.

Damodaran, L., & Olphert, W. (2000). Barriers and facilitators to the use of knowledge management systems. *Behaviour and Information Technology, 19*(6), 405-413.

Davenport, T. H., & Probst, G. (Eds.). (2000). *Knowledge Management Case book: Siemens Best Practices* (2nd ed.). Erlangen: Publicis MCD.

Dietrich, J., & Elgar, C. (2005). *A Formal Description of Design Patterns Using OWL*. Paper presented at the Australian Software Engineering Conference (ASWEC), Brisbane, Australia.

Eberle, M. A. (2003). *Barrieren und Anreizsysteme im Wissensmanagement und der Software-Wiederverwendung.* Student Thesis, University of Kaiserslautern, Kaiserslautern.

Fahey, L., & Prusak, L. (1998). The Eleven Deadliest Sins of Knowledge Management. *California Management Review, 40*(3), 265-276.

Feurstein, M., Natter, M., Mild, A., & Taudes, A. (2001). *Incentives to share knowledge*. Paper presented at the Conference Name|. Retrieved Access Date|. from URL|.

Fincher, S., & Utting, I. (2002). Pedagogical patterns: their place in the genre. *SIGCSE Bulletin, 34*(3), 199-202.

Gabriel, R. P. (2002). *Writer's Workshops and the Work of Making Things*: Addison-Wesley Longman Publishing Co., Inc.

Gamma, E., Richard, H., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software* (3rd printing ed. Vol. 5): Addison-Wesley.

Garzás, J., & Piattini, M. (2005). An Ontology for Microarchitectural Design Knowledge. *IEEE Software, 22*(1), 28-33.

Henninger, S., & Ashokkumar, P. (2006, July 5-7, 2006). *An Ontology-Based Metamodel for Software Patterns.* Paper presented at the 18th International Conference on Software Engineering (SEKE), San Francisco, CA, USA.

HPL. (2005). Hillside Pattern Library. Retrieved 10 October, 2005, from http://hillside.net/patterns/

Jen, H. W., & Yu, M. W. (2006). Measuring KMS success: A respecification of the DeLone and McLean's. *Information and Management, 43*(6), 728-739

Jennex, M. E., & Olfman, L. (2004). *Assessing knowledge management success/effectiveness models.* Paper presented at the SO: Proceedings of the 37th Annual Hawaii International Conference on System.

Jennex, M. E., & Olfman, L. (2006). A model of knowledge management success. *International Journal of Knowledge Management, USA * vol, 2*, July-Sept.

Judicibus, D. d. (1996, 8-9 Jan. 1996). *Reuse: A cultural change.* Paper presented at the Proceedings of the International Workshop on Systematic Reuse, Liverpool, UK.

Kari, L. (1996). Estimating understandability of software documents. *SIGSOFT Softw. Eng. Notes, 21*(4), 81-92.

Long, J. (2001). Software reuse antipatterns. *Software Engineering Notes, 26*(4), 68-76.

Marwick, A. D. (2001). Knowledge management technology. *IBM Systems Journal, 40*(4), 814-830.

Mathi, K. (2004). *Key Success Factors for Knowledge Management.* Master Thesis, University Of Applied Sciences/ Fh Kempten, Kempten, Germany.

Mertins, K. E. H., Peter (Ed.); Vorbeck, Jens (Ed.) (Ed.). (2003). *Knowledge Management. Concepts and Best Practices* (2nd ed.). Berlin: Springer-Verlag.

Morisio, M., Ezran, M., & Tully, C. (2002). Success and Failure Factors in Software Reuse (Vol. 28, pp. 340-357): IEEE Press.

PPR. (2005). Portland Pattern Repository. Retrieved 10 Oct., 2005, from http://c2.com/ppr/, http://en.wikipedia.org/wiki/Portland_Pattern_Repository

Ras, E., Rech, J., & Weber, S. (2008). *Collaborative Authoring of Learning Elements for Adaptive Learning Spaces*. Paper presented at the Conference Name|. Retrieved Access Date|. from URL|.

Rech, J., Decker, B., Ras, E., Jedlitschka, A., & Feldmann, R. L. (2007). The Quality of Knowledge: Knowledge Patterns and Knowledge Refactorings. *International Journal on Knowledge Management (IJKM), 3*(3), 74-103.

Rech, J., Feldmann, R. L., Ras, E., Jedlitschka, A., & Decker, B. (2007). Knowledge Patterns and Knowledge Refactorings for increasing the Quality of Knowledge. In M. E. Jennex (Ed.), *Knowledge Management, Organizational Memory and Transfer Behavior: Global Approaches and Advancements* (pp. 30).

Rech, J., & Ras, E. (2007). Aggregation von Erfahrungen in Erfahrungsdatenbanken. *Künstliche Intelligenz*, 6.

Riege, A. (2005). Three-dozen knowledge-sharing barriers managers must consider. *Journal of Knowledge Management, 9*(3), 18-35.

Rising, L. (2000). *The pattern almanac 2000.* Boston: Addison-Wesley.

Sun, P. Y. T., & Scott, J. L. (2005). An investigation of barriers to knowledge transfer. *Journal of Knowledge Management, 9*(2), 75-90.

Thomas, B. D. (2006). *An Empirical Investigation Of Factors Promoting Knowledge Management System Success.* PhD Thesis, Texas Tech University.

Wolf, P., & Wunram, M. (2003). Barriers to KM between organisational cultures in the face of concurrent enterprising: how to overcome them? *Processes and Foundations for Virtual Organizations. IFIP TC5/ WG5.5 Fourth Working Conference on Virtual Enterprises (PRO VE'03), Lugano, Switzerland, 29 31 Oct. 2003 * Norwell, MA, USA: Kluwer Academic Publishers, 2003, p 333 40.*

TERMS AND DEFINITIONS

Anti-pattern: An anti-pattern is a general, proven, and non-beneficial problem (i.e., bad solution) in a software product or process. It strongly classifies the problem that exhibits negative consequences and provides a solution. Built upon similar experiences, anti-patterns represent "worst-practices" about how to structure or build a software architecture. An example is the "lava flow" anti-pattern that warns about developing a software system without stopping sometimes and reengineering the system. The larger and older such a software system gets, the more dead code and solidified (bad) decisions it carries along.

Best practice: A best practice is commonly understood to be a well-proven, repeatable, and established technique, method, tool, process, or activity that is more certain in delivering the desired results. This indicates that a best practice typically has been used by a large number of people or organizations and / or over a long time, with significant results that are clearly superior over other practices. Knowledge patterns can be used to formalize the description of a best practice.

Design pattern: A design pattern is a general, proven, and beneficial solution to a common, reoccurring problem in software design. Built upon similar experiences, design patterns represent "best-practices" about how to structure or build a software architecture. An example is the façade pattern, which recommends encapsulating a complex subsystem and only allows the connection via a single interface (or "façade") class. This enables the easy exchange and modification of the subsystem.

Explicit Knowledge: This term refers to documented knowledge (e.g., in the form of publications, or webpages) rather then tacit knowledge which is only available through people (i.e., this kind of knowledge only exists in the human brain).

Knowledge Dissemination: This term refers to the process of distribution knowledge to users in demand of this knowledge. Thereby, the distribution can take place within an organization (e.g., across departments or projects) or even across companies or domains. The distribution process can be directly initiated by the user (i.e., in form of a search query) or it is indirectly initiated (e.g., in form of an automated message whenever new knowledge of interest is available, or through a general way of distribution such as a newsletters).

Knowledge refactoring: A (knowledge) refactoring is an explicit, replicable, and beneficial activity that transforms the structure or representation of a knowledge element or component without changing its meaning (i.e., semantics). The goal of knowledge refactoring is the improvement of the quality (e.g., understandability) of the documented knowledge.

Knowledge Quality: The quality of knowledge is important (Marwick, 2001) because knowledge is, typically, reused by people who do not authored it and will have a lifetime of several years. Quality aspects for knowledge include usability-related aspects such a s readability, understandability (Kari, 1996), or learnability, as well as reliability, efficiency, maintainability, portability, etc.

Lessons Learned: A lessons learned (LL) commonly refers to an experience (in most cases a negative one) which manifests in a better understanding and knowledge about a situation and will eventually help in reaching a desirable result in a future situation with a similar context. A frequently used lessons learned eventually becomes a best practice and is used as a foundation for a knowledge pattern.

Software refactoring: A (software) refactoring is an explicit, replicable, and beneficial activity that transforms the structure or representation of a software component without changing its meaning (i.e., behavior). The goal of software refactoring is the improvement of the quality (e.g., maintainability) of the software system.