

Proactive Knowledge Distribution for Agile Processes

Rosina Weberⁱ

College of Information Science & Technology, Drexel University, Rosina.Weber@drexel.edu

Abstract

Monitored distribution (MD) is a case-based approach for proactive knowledge distribution. MD allows the dissemination of knowledge artifacts in a just-in-time fashion in the context of its applicable targeted processes. In MD, knowledge artifacts are retrieved when they are applicable to the task in which a user is currently engaged. We define MD's requirements and argue that it can be applied to agile processes because the targeted processes are collected as an attribute of knowledge artifacts.

1. Monitored Distribution

Monitored Distribution (MD) [1] is an approach for proactive distribution of knowledge artifacts [2]. It has been designed for the distribution of *lessons-learned*, which are knowledge artifacts that embed a validated strategy that positively impacts organizational results when reused [3]. Because of the strong association between lessons-learned and organizational results, MD is integrated with organizational processes.

MD addresses problems associated with other distribution methods that are divorced from targeted organizational processes and require users to have the initiative and skills to access, manipulate and interpret knowledge artifacts. Most importantly, MD motivates the reuse of a knowledge artifact by bringing it to the attention of the user *when* and *where* it is applicable and by including a rationale for its reuse.[4]

The MD approach shifts the burden of knowledge dissemination from the user to the software. MD can accomplish this by relying on an intelligent module that monitors when a lesson-learned should be disseminated to the user by matching the lesson to the user's context.

MD's intelligent module relies on case-based reasoning (CBR). CBR is often recommended for knowledge management (KM) tasks [5] possibly due its flexible knowledge representation and because it uses different techniques to manage a set of knowledge containers [6].

In the case-based module, each case is represented by a lesson-learned that is applicable to a task within an organizational process. Table 1 presents an example of a lesson-learned (from the Navy Lessons Learned System [7]) and its representation structure, which combines indexing elements (i.e., applicable task, preconditions)

and reuse elements (i.e., lesson suggestion, rationale). These elements can be subdivided into more parts, as Table 1 shows.

Table 1. Lesson-learned example

Applicable task	Action: Assign air traffic controllers. Mission type: NEO Task: Provide for Movement Services in Theater of Operations
Pre-conditions	A civilian airport is used for military air traffic.
Lesson suggestion	Assign military air traffic controllers.
Rationale	Type: Failure What? Military traffic overloaded civilian controllers. Why? The rapid build-up of military flight operations at Mactan Intl Airport, Cebu quickly overloaded the civilian host nation controllers.

2. Requirements for Implementing MD

First of all, to benefit from MD, targeted users have to deliver their actions or decisions using a computerized system (e.g., enterprise resource planning). This system has to be flexible enough to allow the integration of the MD approach, which monitors the user's actions. MD is constantly trying to match the user's context with previously recorded lessons-learned.

Accordingly, the second requirement is that lessons-learned are stored in the MD's case base using the representation elements exemplified in Table 1. Because MD has a case representation structure that assesses the similarity between user's contexts and lessons-learned, MD does not require that all tasks be defined *a priori*. As lessons with new tasks are captured, they will match the contexts when these same tasks become current.

Without meeting those conditions, MD's applicability-oriented distribution fails. In this case, MD may attempt to distribute knowledge that is not relevant; potentially preventing the positive impact on organizational processes that lessons-learned are meant to provide.

3. MD in Agile Processes

The ability to adapt seems to be one of the key features to achieve *business agility* [8], and so should be learning from experience and incorporating new knowledge into organizational processes. Sometimes, incorporating

knowledge means changing old or creating new tasks. Therefore the advantages of using MD in support of agile KM processes are two-fold. First it supports agile incorporation of experiential knowledge into organizational processes, allowing an organization to adapt while steered towards its goals. Second, MD represents a process-oriented KM approach that does not rely on static and previously determined processes to support knowledge sharing, making it amenable to implement in agile organizations with dynamic processes.

Agile processes do not pose a difficult obstacle to the MD approach because new tasks and processes are potentially identified before or simultaneous to the identification and learning of knowledge that will eventually be reused in such tasks. Given the flexible framework in the MD approach, new tasks can be included at any point. MD will be able to identify a new task once there is a lesson-learned that is applicable to this task.

4. Direct and Indirect MD

MD has been integrated with a plan authoring tool [4]. MD would display applicable lessons-learned on the screen so the user could decide whether to reuse it or not. This is a direct use of MD.

An indirect use of MD does not deliver lessons directly to the end-user but to an intermediary system. This intermediary system may reuse lessons while producing an outcome for the end-user.

It is reasonable to expect that agile businesses rely on intelligent systems that can adapt to their dynamic context. In this case, MD can deliver lessons to these systems, and incorporate knowledge to be used as an additional source of change and evolution to organizational processes.

An example of this indirect use of MD is a computational intelligence (CI) tool for software testing being currently developed at the National Institute for System Test and Productivity (NISTP) at the University of South Florida. The CI-tool designs a software testing strategy. Because this task is very sensitive to experiential knowledge, it is desirable that lessons are distributed while strategies are being designed. In this case, the CI-tool is designed to execute a dynamic process. As with human software testers, the system has to adapt depending on the type of software to be tested, causing it to evolve as more experiences are gained. The MD approach can contribute to the agility of this process by disseminating lessons-learned directly to this system and incorporating experiential knowledge to improve and change the testing design process.

5. Discussion

One major concern in choosing KM approaches originates from the recognition that successful KM should be accompanied by a correspondent change in culture [9]. On the other hand, today's large organizations require technological KM solutions [10] because most of their processes are automated in enterprise-wise information systems. For example, some of the features that include Cisco in the selective set of agile organizations are its *rapid, highly automated and virtually paperless* processes [8]. Interestingly, the more automated the organization, the easier it is to implement an approach like MD, because these organizations tend to be highly automated and their systems tend to be flexible enough to allow the integration of new models. It is our intention to investigate the extent of the difficulties and challenges of the integration of MD when processes are agile.

References

- [1] D.W. Aha, R. Weber, H. Muñoz-Avila, L.A. Breslow, K.M. Gupta, Lesson Distribution Gap, in: Proceedings of IJCAI, Seattle, WA, 2001, 2, pp. 987-992, AAAI Press, Menlo Park, CA.
- [2] C.W. Holsapple and K.D. Joshi, Organizational knowledge resources, *Decision Support Systems* 31, 2001, pp. 39-54.
- [3] R. Weber, D.W. Aha and I. Becerra-Fernandez, Intelligent Lessons Learned Systems, *International Journal of Expert Systems Research and Applications* 20, No. 1, 2001, pp. 17-34.
- [4] R. Weber and D.W. Aha, Intelligent Delivery of Military Lessons Learned, *Decision Support Systems* 34, 3, 2003, pp. 287-304
- [5] D.W. Aha, I. Becerra-Fernandez, F. Maurer and H. Muñoz-Avila eds., *Exploring Synergies of Knowledge Management and Case-Based Reasoning: Papers from the AAAI 1999 Workshop* (Tech. Rep. WS-99-10). Menlo Park, CA: AAAI Press, 1999.
- [6] M.M. Richter, The Knowledge Contained in Similarity Measures: Some remarks on the invited talk given at ICCBR'95 in Sesimbra, Portugal, October 25, 1995. Online: <http://www.cbr-web.org/documents/Richtericcbr95remarks.html>
- [7] Navy Lessons Learned System, Navy Warfare Development Command, Navy Lessons Learned Database (Feb. 2001) [<http://www.nwdc.navy.mil/Command/NavyLessonsLearned/default.asp>]
- [8] Gartner UK Ltd.: The Age of Agility, Report prepared by Gartner for BT, July 2002.
- [9] W.J. Orlikowski and J.D. Hofman, An Improvisational Model for Change Management: The Case of Groupware Technologies, *Sloan Management Review/Winter*, 1997, 38, 2, pp. 11-21.
- [10] R. Weber and R. Kaplan, Knowledge-based knowledge management. *Innovations in Knowledge Engineering*, (eds). C. Faucher, L. Jain, and N. Ichalkaranje. Physica-Verlag, forthcoming.

¹ The author is currently supported in part by the National Institute for Systems Test and Productivity at USF under the USA Space and Naval Warfare Systems Command grant no. N00039-02-C-3244, for 2130 032 L0, 2002.