When Knowledge and Experience Repositories grow
new Challenges Arise

Raimund L. Feldmann¹, Ioana Rus²

¹Software Engineering Research Group,
Kaiserslautern University of Technology,
P.O. Box 3049, D-67653 Kaiserslautern, Germany
r.feldmann@computer.org

²Fraunhofer USA Center for Experimental Software Engineering Maryland
4321 Hartwick Road, Suite 500, College Park, MD 20740, USA
irus@fc-md.umd.edu

1. Motivation
Knowledge and experience management is a key factor for learning software organizations. Consequently, many organizations are operating their own repositories to systematically store their knowledge and experience, and to provide them to the right users, at the right time. However, once a repository is successfully installed (which is a non-trivial task to start with) new challenges arise when we have to keep it continuously up-to-date. But how do we define “up-to-date”? In a business like software development, where product life cycle becomes shorter and technologies more and more dynamic, it seems to be crucial to always offer developers the latest (i.e., “freshest”) knowledge to support their daily work. In addition, more stable knowledge such as recommended practices, lessons learned, and patterns should be preserved. Such kinds of entries usually become mature over time (i.e., they are “aged” – not fresh).

2. Challenges
There is always the danger of having too many entries in our repositories. If we simply continue to add entries, our repositories might eventually turn into data cemeteries, and, therefore, prevent efficient utilization of information that cannot be easily accessed. But how should dated entries in repositories be handled? Should they be deleted or kept? Do we really want (and need) repository entries to be always that fresh? Or could it sometimes be useful to hold the aged entries? Old entries, for instance, can be used to trace the history of our current experience, to detect new heuristics, or to help correcting wrong assumptions. They also might be of use for maintaining or reengineering our old SW systems.

There are multiple and diverse challenges to maintaining experience bases. Some of them are due to the evolution of the organization in which the repository is running (e.g., a change in competences, as for instance, the switch to, or inclusion of another application domain). Other issues arise because of the evolution of the repository entries (e.g., a new version of a programming language or a more precise process model introduced as a result of gaining experience). Let’s take a look at the following two scenarios:

Scenario 1:
  - Situation: The company faces high employee turnover;
- Need: Keep all employees informed with organizational knowledge, policies, and processes (especially the new employees);
- Goal: The repository should contain organizational knowledge that is pretty stable and that must be accessible to all employees.

In this case of an organizational evolution (i.e., it's changing employees) the content of the repository is not very dynamic, "so old (as age) does not mean bad", unless is it not accurate or some things got updated independently from the change of employees.

Scenario 2:
- Situation: Software development technology changes fast;
- Need: Be on the leading edge with introduction of new technologies, thus the need to quickly bring employees up to speed with the new technologies;
- Goal: The repository should contain dynamic knowledge (always the "fresh-est" knowledge).

In this case of an evolution of entries, the repository might be needed to contain only knowledge related to the newest technologies ("old is not good"). If we can be sure that the old entries will not be needed later (e.g., for maintaining and reengineering the old systems) we could remove them.

To what extent can we generalize these statements? What are the factors that drive decisions regarding the evolution of repositories? Is economics the overall principle that should guide all operations upon the repository (addition, update and delete)?

What is the return on investment (ROI) of an experience repository? ROI is determined by the investments in an activity and the benefits (or returns) for those expenses. In general, the cost of a maintaining a repository consists of expenses for:
- Updating and maintaining the repository – this cost increases with size
- Storing - but is this really an issue with today's technology?
- Extracting new knowledge and evolving the content by different operations such as generalization, abstraction, and specialization – how much more costly this becomes with size increase?
- Retrieval time - might increase with size; also the number of retrieved items might increase while their relevance might decrease.

But what are the benefits of a knowledge repository? How can they be measured?

Concluding, we can state that many factors must be taken into account if we want to guarantee the success of a running repository. So far, it can not exactly be explained how these factors interact and influence each other, nor do we have empirical evidence that could guide us.

3. Outlook

We expect for this panel to foster a lively discussion between practitioners and researchers based on this short introduction to the challenges of running growing knowledge and experience repositories in learning software organizations. New perspectives on these and other related topics, as well as the discussions and the outcome of the LSO panel, might guide us in defining a set of heuristics for managing and using knowledge and experience repositories in the future.