Research on Learning Software Organizations – Past, Present, and Future

Harald Holz[1] and Grigori Melnik[2]

[1] DFKI GmbH, Knowledge Management Department,
PO Box 2080, 67608 Kaiserslautern, Germany
Harald.Holz@dfki.de
[2] Department of Computer Science, University of Calgary,
Calgary, Canada
melnik@cpsc.ucalgary.ca

Knowledge itself is power, not mere argument or ornament.
Francis Bacon (Meditations Sacrae, 1597)

1 Introduction

In order for a software organization to stay competitive, its software development needs to be part of organizational change. The organization's ability to change and to adapt quickly to environmental changes provides a foundation for growth and power [7]. For such changes to happen, the learning capabilities of the organization have to be enhanced, being an essential part of producing more effective and efficient work practices. Moreover, continuous learning is essential for surviving – let alone prospering – in dynamic and competitive environments [15]. The Learning Software Organization (LSO) workshop series has been promoting this vision since 1999, addressing the questions of organizational learning from the software development point of view.

Though the workshop series is relatively young, the ideas it is based on have been circulating for decades. As early as in 1971, Weinberg recognized software development as learning: "writing a program is a process of learning – both for the programmer and the person who commissions the program" [23]. This was superseded by the engineering approach, when software development began to be considered as "software engineering", omitting for a long period the humanistic people-centric aspect of it. The history of LSO workshops reflects this development to a certain degree. In 1999, LSO started with the premise that "with continuous technological change, globalization, business reorganizations, e-migration, etc. there is a continuous shortage of the right knowledge at the right place at the right time. To overcome this shortage is the challenge for the Learning Organization." [20]. In other words, the main challenge considered six years ago was the *availability* of knowledge. As a result, many solutions were built to address this. The proliferation of knowledge bases is a clear indication of this. The knowledge is primarily considered to be an object, and, thus, it can be codified, stored, retrieved and distributed.

Unfortunately, many such solutions suffered from the "build it and they'll come" syndrome which resulted in a lack of user involvement and enthusiasm. Researchers and practitioners in LSO began to realize that knowledge externalization and storage are not automatically equal to knowledge re-use, that building an experience factory for the sake of experience factory will not pay off the investment. Thus, new efforts to enhance the utilization of knowledge/experience bases/repositories along with improving the software development process commenced. Years 2000 and 2001 were greatly influenced by the Software Process Improvement (SPI) initiatives. The challenges addressed went beyond the availability of knowledge – but further into its *understandability*, *re-use*, *relevancy* and *applicability*. The LSO 2001 main theme was the enablement of the members of the learning software organizations "to effectively quarrel situational requirements, taking past experience into account. Besides improving internal communication (group learning), this also includes documenting relevant knowledge and storing it (for reuse) in an organizational, corporate memory" [1].

Nowadays, it is commonly recognized that promotion of a learning culture and fostering of the exchange of experiences are imperative. Increasingly, experts agree that approaches to achieve this must be based on interdisciplinary research, taking into account results from economical, organizational, cultural, psychological and technological areas.

The year 2002 was marked by the symbiosis of organizational learning and *agility*, the problems facing both the LSO community and the agile methods community seeming to be complementary [10]. Specifically, for the LSO community, the issue is how to quickly adapt to new technologies and market pressures, while for the agile community the issues are how not to lose institutional knowledge and how to enable inter-team learning. The participants commonly recognized the need for a balance between knowledge capture/dissemination and flexibility that enhances the ability of an organization to quickly adapt. In the meantime, the debate about the epistemology of knowledge – whether knowledge is an object or a relation (a context-bound one) – continues.

In 2003, the workshop progressed into the aspects of the evolution of learning organizations and the resulting evolution of repositories they use. Essentially, the workshop focused on the issues of *maintainability* and *scalability* of externalized knowledge [17].

This year (2004), we continue the advancement of the concepts, approaches and techniques to help learning software organizations succeed. The papers included in this volume clearly build upon the results of the previous five workshops. Though a good portion of research today is still dedicated to the development of knowledge management methods and tools, there is an increasing trend towards knowledge management approaches that are lightweight, i.e., do not introduce a considerable additional burden on developers and end users, while at the same time ensuring that the hoped for experience factories do not become "experience cemeteries" which no employee uses. Consequently, the focus is on practical knowledge management initiatives that:

- allow for an incremental adoption without a large up-front investment;
- are flexible enough to allow quick and easy improvements;

- encompass not only the structure, the strategies and the systems of the learning organization itself, but also of those who develop, follow and utilize these structure, strategies and systems.

The following section briefly summarizes current work reflecting the state-of-art and state-of-practice in learning software organizations as presented at the Sixth International Workshop on Learning Software Organizations (LSO 2004) in Banff, Canada.

2 Current Topics in LSO Research and Practice

The 13 full papers and 3 short papers in this volume are drawn from an international base of authors (53), including Belgium, Brazil, Canada, Germany, Mexico, New Zealand, Norway, Portugal, Spain, Switzerland, and United Arab Emirates. A consistent message across all these diverse contributions is that, in order to be effective and agile, we should consider organizational learning as a holistic process, taking into account the particularities of the organization under consideration. Most concepts, approaches and tools will be applicable only in a certain context. The contributions are organized into the following chapters:

- Experience-Based Information Systems
- Software Maintenance
- Communities of Practice
- Planning LSOs
- Case Studies and Experience Reports

Experience-Based Information Systems

Software development processes consist of various knowledge-intensive tasks during which software engineers need to make informed decisions. The contributions in this chapter describe information systems that support users in their decision-making in diverse tasks such as risk management and COTS selection.

Falbo et al. present an ontology-based to support organizational learning in risk management [5]. Their tool GeRis supports novice project managers in the identification, evaluation, ranking, and contingency planning of risks for a current project by providing the manager with corresponding experience from similar, stored projects.

Santos et al. present an enterprise ontology that provides the basis for various tools as part of an enterprise-oriented software development environment (EOSDE) [21]. They illustrate their approach by describing two tools that make use of this ontology: Sapiens, a corporate ‘yellow pages’ tool, and RHPlan, a resource allocation planning tool. Their EOSDE is already being used in 18 small and medium-size software companies.

In [9], Gomes et al. describe their tool REBUILDER, a CBR system that supports designers by retrieving former UML designs similar to the current design diagram, and by automatically augmenting the current diagram by missing elements from former designs. Moreover, the system provides functionality to evaluate the resulting diagrams based on various object-oriented metrics.

Mohamed et al. propose a conceptual model to support decision making during COTS selection processes [13]. They outline how this model can be implemented as an agent-based decision-support system that addresses important issues such as changing stakeholder preferences and evaluation process simulation to try out different scenarios.

Ras and Weibelzahl argue that experience packages retrieved from repositories are often inadequate for learning and competence development, e.g., because users might not have sufficient knowledge to understand the package content, or because users might be unsure of the risks involved on applying the packaged experience [16]. Their approach addresses these issues by automatically enriching experience packages with additional learning elements based on didactical considerations.

Software Maintenance

Several studies indicate that the processes needed to correct errors in a software system, or to adapt a system to the ever-changing environment incurs most of the overall expenses during the life-cycle of a software product.

In [22], de Sousa et al. propose to use postmortem analysis (PMA) to help manage the knowledge gained during maintenance projects, both knowledge on the maintenance process itself and on the system maintened. Based on a standardized maintenance process, they detail when to conduct PMA during process execution, what knowledge to look for, and how to perform PMA during maintenance.

Rodríguez et al. outline the architecture of a multi-agent system designed to manage knowledge generated during the software maintenance process [18]. Their web-based system aims at proactively providing maintenance engineers with knowledge sources that could help them in carrying out their current tasks.

In [19], Roth-Berghofer reports on experience gained from setting-up and running an internal CAD/CAM help desk support system for IT-related problems at a large company. He discusses lessons learned from this project, where a systematic maintenance process needed to be established, e.g. in order to enhance the domain model appropriately whenever necessary because of environmental changes.

Communities of Practice

Communities of Practice (CoP) are informal groups of organization members that share common interest, practices, and subjects. Approaches that integrate CoPs into daily work processes by lightweight IT support as well as the advantages of informal knowledge exchange are discussed in this chapter.

Chau and Maurer present the lightweight, Wiki-based knowledge management tools MASE and EB [2]. These tools support agile teams by providing them with a process support systems that enables users to share their experience by a collaborative creation and task-specific retrieval of WIKI pages containing information related to the task type. Moreover, first result from a study on inter-team learning using MASE and EB are reported.

In [14], Montoni et al. present a knowledge management approach for acquiring and preserving knowledge related to specific software processes. Their tool ACKNOWLEDGE supports the capture of different knowledge items such as lessons-learned or ideas, as well as their subsequent evaluation by an evaluation committee, and the packaging by knowledge managers. Knowledge items can be retrieved from a community of practice repository via a web-based system with regard to a given process type, user-specified keywords and knowledge types.

In [12], Melnik and Richter analyze the role of imprecise statements in conversations among software developers. They argue that impreciseness can be very useful in interaction, and describe how finding an optimal level of impreciseness can be interpreted as a learning problem for software organizations.

Planning LSOs

An important characteristic of learning software organizations is that learning processes are systematic – learning should not occur in an ad-hoc, chaotic fashion, but as part of the organization's overall strategy, where continuous learning is identified as an explicit goal and methods are deployed to achieve it.

In order to be agile, integrated and aligned, an organization must be architected accordingly. Therefore, Goethals et al. present their framework FADE for managing the concurrent development of the business and the ICT side of an enterprise [8]. FADE identifies several enterprise life-cycle phases as well as their links to the strategic, tactical, and operational level.

Case Studies and Experience Reports

The contributions in this chapter report on experiences and case studies conducted in an industry context. The discuss successes achieved as well as mistakes made, and outline lessons learned.

In [3], Doran reports on his experience with the implementation of knowledge management techniques in an agile software development department of a start-up company. He outlines the difficulties encountered ant approaches chosen for handling knowledge related to process, problem domain and technology, and discusses the tools introduced into the company to support these approaches.

Based on their experience with an industry partner, Draheim and Weber outline general conditions for an approach to collaborative learning of software organizations and academia [4]. They propose a co-knowledge acquisition and sharing process that is lightweight, peer-to-peer, and demand-driven.

In [6], Folkestad et al. report on a case study on the effect of introducing the Unified Process and object-oriented technologies into a company. The authors demonstrate the application of activity theory in a qualitative approach, and identify the iterative development introduced by the Unified Process to have a large effect on organizational and individual learning, flanked by new roles and more formal communication patterns.

John and Melster report on their experience from building and using a knowledge model for a knowledge network for know-how transfer in the area of software engineering, using a classical approach to model building [11]. Based on this experience, they outline a personal and peer-to-peer knowledge management approach that better takes into account the flexible and social structures of knowledge expert communities.

3 Conclusion

The diversity of topics addressed by the contributions presented at LSO 2004 clearly reflects the interdisciplinary viewpoint required for successful knowledge management approaches for software-intensive organizations. Despite the advances reported on, further effort will need to be spent on a number of outstanding issues and challenges, in particular: Techniques, methods, and tools that allow for a lightweight, incremental phase-in of knowledge management; peer-to-peer knowledge sharing; scalability of proposed knowledge management approaches; measuring the success of these approaches, to name but a few.

Notwithstanding innovations in the domain of learning software organizations, we continue to recognize that human skills, expertise, and relationships will remain the most valuable assets of a software-intensive organization.

References

1. K.D. Althoff, R.L. Feldmann, W. Müller (Eds.): Advances in Learning Software Organizations, Third International Workshop, LSO 2001, Lecture Notes in Computer Science, vol. 2176, Springer Verlag, 2001.
2. T. Chau, F. Maurer: Tool Support for Inter-Team Learning in Agile Software Organizations. LNCS 3096, Springer Verlag, 2004.
3. H.D. Doran: Agile Knowledge Management in Practice. LNCS 3096, Springer Verlag, 2004.
4. D. Draheim, G. Weber: Co-Knowledge Acquisition of Software Organizations and Academia. LNCS 3096, Springer Verlag, 2004.
5. R.A. Falbo, F.B. Ruy, G. Bertollo, D.F. Togneri: Learning How to Manage Risks Using Organizational Knowledge. LNCS 3096, Springer Verlag, 2004.
6. H. Folkestad, E. Pilskog, B. Tessem: Effects of Software Process in Organization Development - A Case Study. LNCS 3096, Springer Verlag, 2004.
7. Gartner UK Ltd.: "The Age of Agility", Report prepared by Gartner for BT, July 2002.
8. F. Goethals, J. Vandenbulcke, W. Lemahieu, M. Snoeck: A framework for managing concurrent business and ICT development. LNCS 3096, Springer Verlag, 2004.
9. P. Gomes, F.C. Pereira, P. Paiva, N. Seco, P. Carreiro, J.L. Ferreira, C. Bento: REBUILDER: A CBR Approach to Knowledge Management in Software Design. LNCS 3096, Springer Verlag, 2004.

10. S. Henninger, F. Maurer (Eds.): Advances in Learning Software Organizations, 4th International Workshop, LSO 2002, Lecture Notes in Computer Science, vol. 2640, Springer Verlag, 2002.
11. M. John, R. Melster: Knowledge networks -- managing collaborative knowledge spaces. LNCS 3096, Springer Verlag, 2004.
12. G. Melnik, M.M. Richter: Impreciseness and Its Value from the Perspective of Software Organizations and Learning. LNCS 3096, Springer Verlag, 2004.
13. A. Mohamed, T. Wanyama, G. Ruhe, A. Eberlein, B. Far: COTS Evaluation Supported By Knowledge Bases. LNCS 3096, Springer Verlag, 2004.
14. M. Montoni, R. Miranda, A.R. Rocha, G.H. Travassos: Knowledge Acquisition and Communities of Practice: an Approach to Convert Individual Knowledge into Multi-Organizational Knowledge. LNCS 3096, Springer Verlag, 2004.
15. M. Popper, R. Lipshitz: Organizational Learning: Mechanisms, Culture, and Feasibility. Management Learning, 31(2), 2000: 181-196.
16. E. Ras, S. Weibelzahl: Embedding Experiences in Micro-Didactical Arrangements. LNCS 3096, Springer Verlag, 2004.
17. U. Reimer, A. Abecker, S. Staab, G. Stumme (Eds.): Proceedings WM 2003: Professionelles Wissensmanagement - Erfahrungen und Visionen. GI-Edition - Lecture Notes in Informatics (LNI), Vol. P-28, Bonner Köllen Verlag (Germany), 2003.
18. O.M. Rodríguez, A. Vizcaíno, A.I. Martínez, M. Piattini, J. Favela: How to Manage Knowledge in the Software Maintenance Process. LNCS 3096, Springer Verlag, 2004.
19. T.R. Roth-Berghofer: Learning from HOMER, a Case-Based Help Desk Support System. LNCS 3096, Springer Verlag, 2004.
20. G. Ruhe, F. Bomarius (Eds.): Learning Software Organizations. Methodology and Applications, Lecture Notes in Computer Science, vol. 1756, Springer Verlag, 1999: 4.
21. G. Santos, K. Villela, L. Schnaider, A.R. Rocha, G.H. Travassos: Building ontology based tools for a software development environment. LNCS 3096, Springer Verlag, 2004.
22. K.D. de Sousa, N. Anquetil, K.M. de Oliveira: Learning Software Maintenance Organizations. LNCS 3096, Springer Verlag, 2004.
23. G.M. Weinberg: The Psychology of Computer Programming, Dorset House Publishing, 1998: 12.