THE USE OF CASE-BASED REASONING IN A KNOWLEDGE-BASED (LEARNING) SOFTWARE DEVELOPMENT ORGANIZATIONS

ABDELRAHMAN ELSHARIF KARRAR

Associate Professor, College of Computer Science and Engineering,
Taibah University, Saudi Arabia

Abstract— This paper presents the strategic use of case-based reasoning (CBR) in software industry, particularly in software development organizations that use the knowledge management to improve software quality and productivity.
It describes how knowledge-based organization for the software domain can use both approaches from Software Engineering, and Case- Based Reasoning . and illustrates the requisites for a domain to facilitate learning in the software engineering sphere.
Paper is organized as follows. Section II describes methods in the software engineering field , like Quality Improvement Paradigm (QIP) and Experience Factory model (EF) , Section III presents the requisites for a domain to facilitate learning in the software engineering sphere and Section IV illustrates the motivation for using the Case – Based Reasoning (CBR) technology. Finally, Section V presents conclusion.

Keywords— Software Engineering (SE), Case-Based Reasoning (CBR) , Knowledge-Based Organization (KBO).

I. INTRODUCTION

In the present day software sector, there are several complex demands. These includes short lead-time, increased level of quality demands or requirements, persistent infiltration of new sophisticated technologies, as well as, increased software intricacy. The conventional production-oriented systems for meeting needs such as quality assurance have increasingly become obsolete in the development-oriented software sphere, which requires ongoing fast learning. Indeed, such learning is capable of not only acquiring but also maintaining leading-edge competencies. The conventional way of learning has been deemed as too slow and at the same time, ineffective especially when applied as a means through which new methods, techniques or demands are adopted and adapted.
The issue of the conventional group or individual learning is made to look even more inappropriate where it is not pursued in a goal-oriented manner, if not supported by methodical, organizational and technical means, and if not managed as a project that is integral to the corporate success. In this light, one can rightly argue that learning on an organizational level, as well as, capitalizing on organizational knowledge assets is of central importance if the present day software-dependent sectors are to succeed. The case is aggravated by the notion that SE domain, which abbreviates software engineering, has not transitioned into a real engineering field, with its short history being one of the contributory factors. As Althoff, Bomarius and Tautz (1998) revealed, more research is required in applied software engineering.
The research and the resultant knowledge have to be validated, in a thorough manner, in industrial settings. This is important since it allows for the development of universally accepted processes and technique, which in turn form the core of an advanced software engineeringas a discipline. It is only through an appropriate interaction between software engineering research with a practice that the maturation or the advancement of the field. As a result, the gap between software engineering research and practice will be bridged. At the same time, the growing software sector's demands will be met more efficiently.
The present study focuses on the work carried out at Fraunhofer IESE. At the heart of Fraunhofer IESE processes are the development and transfer of learning (Knowledge – based) organizations for software development enterprises into industrial practices. To fulfill this purpose, the paper will define the manner in which Los for the software domain could be established upon not only advanced or mature software engineering approaches but also industrial strength from AI.

II. METHODS IN THE SOFTWARE ENGINEERING FIELD

According to Recio-García, González-Calero and Díaz-Agudo (2014), experimental software engineering entails several fundamental premises. While this is the case, according to the researcher, the need for understanding and improving the quality and the productivity of software seems to be the most important. The researcher stressed that, just as any other engineering domain, any improvements or understanding ought to be founded on empirical evidence, as well as, on every kind of specific project experiences. According to the researcher, even for smaller software enterprises, accumulating information, for it to be meaningful, has to be modeled, generalized, structured and stored in such a manner that it can be reapplied. While this is the case, in the software engineering practice, reuse of information is not common in projects involving

software development. This applies in particular given that the reuse can potentially conflict with the local objectives of a project. Additional efforts, on the part of an individual, to allow for futur4e reuse of results are not usually rewarded or penalized.

According to Tawfik and Kolodner (2016), to introduce the mechanisms of a Knowledge - based organization into such a culture, it requires the integration of an ongoing build-up, as well as, efficient reuse of information into the organizational learning project goals. These goals, according to Althoff, Bomarius and Tautz (1998), have to be separate from the development projects. Recio-García, González-Calero and Díaz-Agudo (2014) revealed that goals related to learning and reuse ought to be defined in a clear manner. Enough resources should also be assigned. Further, the OL projects have to be managed in the same way that other developmental projects are. After the reuse and learning of information have been adopted as the standard software development project practices, the major portion of the responsibilities and tasks of the support system should be assimilated using the general development organization.

Patil and Bamnote (2015) revealed that an organizational structure that is formed by a remote support and project development organizations has been proposed by the Experience Factory Organization (EF), often classified under Romback and Basili. EF represents an organization, which provides support to the software development projects especially those conducted through the analysis and synthesis of all forms of experiences drawn from the initiatives, serving as the repository for such forms of experiences. The following figure represents the six phases of QIP (Quality Improvement Paradigm) Fig(1). QIP, according to Althoff, Bomarius and Tautz (1998), explains the organizational learning tasks of the EF along with project organization Fig(2).

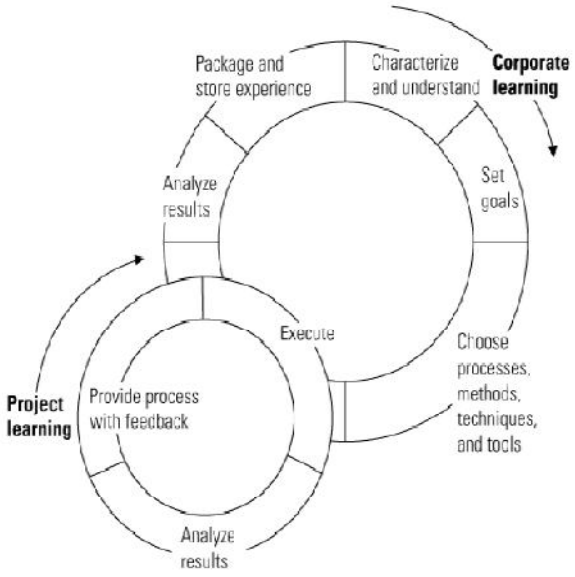


Figure (1) The Quality Improvement Paradigm (QIP)

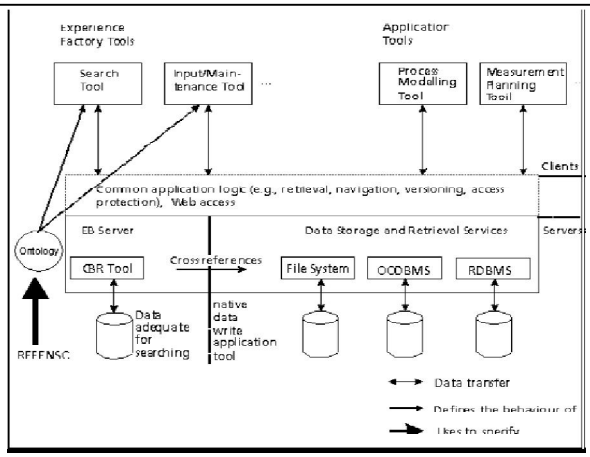


Figure (2) Architecture of the Software Engineering Experience Environment

III. THE REQUISITES FOR A DOMAIN TO FACILITATE LEARNING IN THE SOFTWARE ENGINEERING SPHERE

For the purpose of this paper, learning primarily defines OL as opposed to a team or individual learning. According to Mansouri, Mille and Hamdi-Cherif (2014), a growing level of diverse experiences is usually required, on the part of an expert, if knowledge is to be established regarding a specified area of interest. The expert has to participate in different projects to accumulate sufficient experience. Especially in a dynamic environment, gaining such experience might take essentially too long. More time, according to Althoff, Bomarius and Tautz (1998), is usually taken up to the practice of coaching staff who do not have sufficient experience in the standard situation. What this means is that no real expert work can be conducted. Sharing knowledge with the junior staff is of central importance. It can assist in ensuring that the knowledge held by experts do not become obsolete. With sharing, it means that the expert knowledge becomes more valuable for the organization. Recio-García, González-Calero and Díaz-Agudo (2014) revealed that a well-structured, as well as, an easy to use knowledge representation makes people smarter. Where the applicability of the knowledge is tested, it tends to stabilize even further, usually evolving into what Tawfik and Kolodner (2016) acknowledge as best practices. In the long term, these practices are, however gradually, factored into the standard practices and eventually, they are laid out in the manuals and handbooks.

One of the most critical requirements for effective reuse of software engineering knowledge is support for the mechanisms of continuous, incremental form of learning. Patil and Bamnote (2015) indicated that the elicitation of experiences from projects along with the feedback of experiences to the organization of a project usually necessitates functionality in order to support recording, packaging, and effective nature of reuse. The second requirement is the storage of every kind of artifact associated with software knowledge.

As it was revealed by Tawfik and Kolodner (2016), considering the development and improvement initiatives, a series of artifacts should be stored. Also, lessons learned about these powerful artifacts should and the observations should be documented. To fulfill this, characterizing own schemata is requisite. As a result, the Organizational Memories OM is integrated Fig(3), usually from what He (2014) recognizes as heterogeneous informational sources.

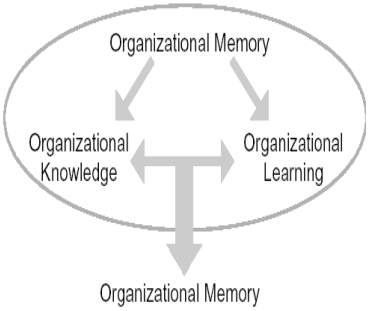


Figure (3) Organizational Memory

Bringing different artifacts together to form a coherent model necessitates flexible modular overall schema for the OM, which can potentially allow interfacing with diverse informational sources and tools. The third requirement relates to retrieval that is founded on incomplete information. Patil and Bamnote (2015) supposed that not all information required in order to characterize, in a comprehensive manner, an artifact is constituted within the OM. Despite the case, there is a possibility of carrying out an effective search of the OM.

A fourth requirement is support for maintenance of experience. According to Mansouri, Mille and Hamdi-Cherif (2014), as an enterprise learns more regarding the processes of software development, it is central to reorganize knowledge. What this means is that OM has to be maintained in a continuous manner. Thus, it is ideal that the experience base provides the necessary support to the procedure of restructuring software engineering knowledge. The ultimate requirement is learning from examples. He (2014) revealed that, where software engineers are confronted with a problem, they usually think of the solutions to the past projects in which similar challenges took place. In the light of this, it is important to capture concrete models as what Mansouri, Mille and Hamdi-Cherif (2014) refers to as problem/solution pairs.

IV. THE MOTIVATION FOR USING THE CASE-BASED REASONING TECHNOLOGY

According to He (2014), two important arguments lead to the belief that the CBR technology is the most promising among the AI technologies. The first argument is a technical one. It is often related that learning from examples resembles a machine learning model. He (2014) revealed that technology, which is founded on both fields, is a CBR one. It can also point out to a natural solution to the retrieval requisites. The second argument is a business-oriented one. According to the argument, it is necessary to support the QIP, including incremental, sustained learning. CBR technology can play this supportive role quite effectively.

CONCLUSION

This paper has evidently exhibited that the most important operations that surround Organizational Memories OM have to be organized in separate organizational units. As it has been shown, this corresponds to the separation in which has been reported as project organization as well as, the experience factory or the EF. An EF has its roles and tends to carry its software engineering projects. It is administered by goals, which share close ties with business objectives. While this is the case, EF can potentially create a conflict. Thus, it is important to establish the best way to which the objectives of the EF and those of the overall organization can be facilitated more efficiently. It has also been learned that a system for supporting knowledge – based organizations should be open. It has been revealed that closed organization cannot solve the issues and challenges in industries that feature a diversity of tools and methods such as those found in the software engineering domain.

REFERENCES

[1] Althoff, K.D., Bomarius, F. and Tautz, C., 1998, August. Using Case-Based Reasoning Technology to Build Learning Software Organizations. In OM.

[2] He, W., 2014. A framework of combining case□based reasoning with a work breakdown structure for estimating the cost of online course production projects. British Journal of Educational Technology, 45(4), pp.595-605.

[3] Mansouri, D., Mille, A. and Hamdi-Cherif, A., 2014. Adaptive delivery of trainings using ontologies and case-based reasoning. Arabian Journal for Science and Engineering, 39(3), pp.1849-1861.

[4] Patil, G., & Bamnote, G. R., 2015. A Survey on Knowledge Management in Small-Sized Software Organizations. Routledge, London: United Kingdom.

[5] Recio-García, J.A., González-Calero, P.A. and Díaz-Agudo, B., 2014. jcolibri2: A framework for building Case-based reasoning systems. Science of Computer Programming, 79, pp.126-145.

[6] Tawfik, A.A. and Kolodner, J.L., 2016. Systematizing Scaffolding for Problem-Based Learning: A View from Case-Based Reasoning. Interdisciplinary Journal of Problem-Based Learning, 10(1), p.6.

★★★