The Concept of "Ba" Applied to Software Knowledge

Nuno Flores, Ademar Aguiar, Hugo Sereno
Department of Informatics Engineering
University of Porto - Faculty of Engineering
Rua Roberto Frias s/n, Porto, Portugal
{nuno.flores, ademar.aguiar, hugosf}@fe.up.pt

ABSTRACT

Software development is a knowledge-intensive activity. Software products usually start as a simple idea, or a vision, and then progress towards a final deliverable product. Along this evolution, there is a lot of knowledge that is captured, organized, and shared, leading to new knowledge, both as a whole and within specific development activities. The concept of "Ba" provides a foundation to advance individual and collective knowledge, which describes knowledge creation as a spiral involving tacit and explicit knowledge: the Socialization, Externalization, Combination, Internalization model (a.k.a. SECI model). By applying this foundation to software development, we found issues that may hinder the effective knowledge management cycle. In this paper, we present a vision and a set of requirements for tools to overcome such issues and therefore better support the whole process of software knowledge evolution.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirements/Specifications—*tools*

General Terms

Software Development, Knowledge Management, Tools

Keywords

software knowledge, software process, knowledge creation

1. INTRODUCTION

Developing software is a highly creative process. Creativity comes from having an idea and devising ways to materialize that idea into a software product. Nevertheless, it is not just a mental process. It is something that takes place in the "phenomenal" space. This space can emerge from individuals, working groups, project teams, and other clusters within an organization or community.

CHASE'14, June 2 – June 3, 2014, Hyderabad, India
ACM 978-1-4503-2860-9/14/06
http://dx.doi.org/10.1145/2593702.2593713

Knowledge is intangible, boundary-less and dynamic. When it becomes tangible, it turns into information and resides in media and networks. Knowledge-creating teams play a key role in value creation, but if this knowledge isn't used at a specific time and place, it is of no value. "Ba" is where knowledge creation takes place[1].

The concept of "Ba" [13] provides a platform for advancing individual and collective knowledge by unifying the physical space (e.g. offices, open spaces, meeting rooms), the virtual space (e.g. e-mail, chat rooms, task and team planning apps), and the mental space (e.g. shared experiences, ideas, ideals, visions), where knowledge creation takes place. "Ba" defines knowledge creation as a cyclic process of interactions between tacit and explicit knowledge, whether of the self, the team, or the organization. It names it "Socialization Externalization Combination Internalization" (SECI) model.

"Ba" can be mapped into the software development process, recognizing that the SECI model is present throughout the several stages of development, although at different paces in different activities.

The authors research goal is to find out that, despite existing tools providing support for many software development activities, when it comes to knowledge management, there are still gaps that hinder the effective support of the SECI cycle within the software development process, Those gaps should then be analyzed to inspire the creation or evolution of better software development practices and tools.

This paper presents a shared vision and a set of requirements that tools should cover so that the "Ba" concept can be better supported in software development, and therefore to help the evolution of knowledge along the software lifecycle. This work results from a collective effort conducted by the authors and their students along a master's course on Agile Methods in the University of Porto.

2. MANAGING SOFTWARE KNOWLEDGE

Managing knowledge is all about individuals, teams and organizations collectively sharing knowledge. Within the software development lifecycle, all these elements play a role in the huge amount of knowledge created, shared and reused.

Knowledge creation is the incorporation and merging of new and old experiences to provide new knowledge. Consequently, knowledge rarely stays in the same state or is understood in the same context. In a software project, knowledge is created by past experiences of other projects and practical experience of processes from other companies. When

[1]Actually, the word "Ba" roughly translates into the English word "place"

this knowledge is shared with other individuals, it is enhanced, merged and refined to create new knowledge. Due to its intrinsic tacit nature, knowledge at this level becomes a challenge to capture and record outside of the individuals minds.

Effective knowledge transfer depends on both the development methodology, the organization culture, and the support by any existing infrastructure.

2.1 Development Methodology Factor

Looking at a traditional "waterfall" methodology, it is expected that requirements do not change, issues do not arise, and that things go as planned. In this scenario, little knowledge would be lost as there was no need to feed it back to the knowledge management process. But frequently, extra requirements are discussed with the customer, issues prompt in and planning shifts and the knowledge process is not updated. As a result, knowledge is lost, causing the client relationship being damaged, time being spent in redoing requirements and project loses on turn over [11]. A more agile or iterative methodology improves this highly likely change of knowledge, by shortening the creation cycle and minimizing the need for explicit knowledge to be available. Nevertheless, there are still caveats regarding the transfer of tacit knowledge throughout the whole development process, specially between phases and respective people involved, where its iterative nature is not in sync with each other.

2.2 Organization Culture Factor

It is a misconception to assume that all companies sharing knowledge have their people communicating regularly and have a great working environment. A large amount of tacit knowledge is lost due to fear of a work colleague, or a culture where no one talks to each other, or a culture where everyone leaves after a short period, therefore forming no relationships [13]. The key here is communication, whereas if it is not properly promoted and supported, it may become a barrier to effective knowledge transfer.

2.3 Infrastructure Factor

Supporting knowledge transfer should cover both conversion of knowledge into information (explicitly formatted artefacts, e.g. documents, code, models, etc.) and capturing of tacit knowledge into explicit knowledge (promoting communication through the infrastructure, e.g. comments, forums, chat rooms). This infrastructure is usually articulated with the development process by having specific tools covering specific activities (e.g. planning, communicating, coding, testing, documenting, etc.). Despite most of them being integrated in a single environment, they are not seamlessly incorporated with the knowledge they interchange. There are still chunks of knowledge that are lost in the process of interchanging information within the infrastructure and between the process and its actors.

Overall, there are barriers in all knowledge management scenarios originating from people, process and technology. It is therefore important to understand how the knowledge flows within the whole software development process environment and, at the utmost, provide support for collaboration and sharing, managing of tacit knowledge and feedback into the knowledge base. The "Ba" concept presents a knowledge creation idea that can also be found in the process of developing software and that should be accounted for when trying to support software knowledge management.

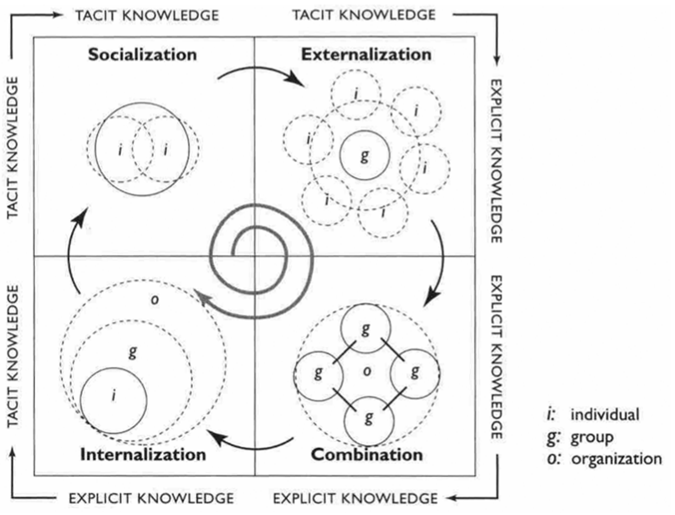

Figure 1: The SECI model of knowledge creation. Adapted from [14]

3. "BA" APPLIED TO SOFTWARE

The "Ba" concept [14] defines knowledge creation as a self-transcendental process. Although quite an abstract idea, it can be practiced by following the "Socialization Externalization Combination Internalization" (SECI) cycle or model. It categorizes knowledge as being *explicit* or *tacit*.

Explicit knowledge can be expressed in words or numbers and shared in various forms and formats (data, documents, specifications, etc.). It can be transmitted between individuals both formally and systematically.

Tacit knowledge is highly personal and hard to formalize (hunches, intuition, subjective insight, etc.), making it difficult to share with others. It has two dimensions: *technical*, encompassing personal skills and know-how, and *cognitive*, consisting of beliefs, ideas, values and mental models.

According to this model, knowledge creation is a spiralling process of interactions between explicit knowledge and tacit knowledge, and is composed of four conversion steps:

Socialization Involves the sharing of knowledge between individuals. Knowledge is exchanged through joint activities and physical proximity. Sometimes just sharing the same working environment allows for dissemination of tacit knowledge.

Externalization Expressing tacit knowledge and its translation into comprehensive forms that can be understood by others. It forces tacit knowledge to be articulated into explicit knowledge through expression techniques (words, concepts, narratives, etc.), using dialogue to support the effective conversion.

Combination Involves the conversion of explicit knowledge into more complex sets of explicit knowledge. Key issues are communication, sharing and systematization of knowledge. This phase goes through three phases: collecting externalized knowledge and combining it; disseminating it over the organization; and improving its usability.

Internalization Individuals convert the explicit knowledge into new tacit knowledge by action and practice. Learn-by-doing and training are common practices that allow internalization of knowledge, updating concepts and methods, and the way individuals (and, thus, the organization) perceives the world.

Thus, the SECI Model (depicted in Figure 1) describes a dynamic process of exchange and transformation between explicit and tacit knowledge, conceptualizing the evolution of knowledge within a social organization through a series of self-transcendental processes.

Software development encompasses several activities, lead by different people, playing different roles, with a wide range of technical skills in several disciplines, globally involving a large set of interdependent artefacts. All of the knowledge involved must be efficiently exchanged, shared, and communicated, sometimes face to face, other times using digital tools. In all these interconnections (person to person, tool to person, tool to tool) may reside possible points of communication inefficiency that must be analyzed and managed, being some of them possibly supported by tools.

4. "BA" TOOL SUPPORT

There are lots of existing tools that provide support to software knowledge management. Usually, their knowledge-view is centered around the process, the artifacts, and the communication [16]. To evaluate the usefulness of a tool in supporting ”Ba”, the analysis should encompass other aspects beyond those of just coping with the crystallization and evolution of explicit knowledge. It should be noted that the SECI cycle regards human interaction and emotions, which have a relevant part in the knowledge creation process. Therefore, the authors propose that tools supporting the ”Ba” concept should take the following requirements into consideration:

- *Team (people) awareness.* The tools should have a way to represent the people that is important and relevant to the knowledge creation and the evolution process. There should be a way to form relationships between community members and notions on who can be more helpful in certain activities. The presence of role-playing and gamification features would allow for a better engagement and commitment of users.

- *Sense of common place.* ”Ba” roughly translates into the English word ”place”. Users should feel comfortable and with a familiar feeling when using the tool. It should gather common and shared knowledge, and the user should feel he/she belongs there, and has a role (or roles) to play. Much like a ”neighbourhood” or a place one goes to ”hang out”.

- *Strong interaction.* Open dialogue, debating, brainstorming, and sharing thoughts and ideas should be at the core of the tool support. P2P communication, chat rooms, forums, comments, blogs, and similar features are examples of supporting tools.

- *Personal tailoring.* Users need to have ”their place”, where they can rearrange their own knowledge and tailor it to their personal needs. It instills a sense of privacy and ownership that brings stability and confidence into the tool.

- *Content-flexible.* The tool should allow a variety of content formats and simple ways of manipulating those contents. Externalizing tacit knowledge into a readily understandable form should come as a natural ability of using the tool. Text, images, drawings, whiteboards, sound bits, videos and other information containers should be available for the users to communicate and share their own knowledge.

- *Allow for practice.* Either integrated or articulated with, the tool should allow the users to materialize their activities into its expected products. Artifacts editors, IDEs for development, mind mappers for jotting down ideas, are just a few examples of tools that promote the ”learn-by-doing” philosophy for incorporating tacit knowledge.

Take notice that such supporting set of tools is expected to be adopted within a larger context, encompassing the organization’s culture and practices that, in conjunction, would promote the knowledge creation process within the software team. The authors aim to validate that only with such a combination can the ”Ba” concept be truly supported within an ecosystem of software development.

5. CASE STUDY

A case study was performed by the authors specifically to experiment the application of the ”Ba” concept to software development. This was done in the context of a contract for the development of a technological platform to support the MOVERCADO ecosystem [4].

The contract consists on the development of a complex mobile-based messaging platform that enables a more efficient aid distribution and impact, by facilitating interactions between all its parts. The platform has two main components: the flow board, responsible for helping to define and orchestrate those interactions, its inputs needed and the outputs produced in specific flows; and a web interface, enabling management of reports, dashboards and campaigns, created according to existing flows, together with their related information (e.g. flows, activities, products, services, participants).

In order to balance effort with effective reusability, the development was done in a fast-paced way, throughout six iterations of two-weeks using an agile process (Scrum) and best practices of evolving reusable software systems [15].

As depicted in Figure 2, a team of 20 elements was engaged in the whole agile development process, playing different roles, and producing diversified artefacts. Activities ranged from producing an initial and further versions of the strategic vision (done by stakeholders), to planning (done by the product owners team with the development team), to all other kind of activities required, such as programming, designing, modeling, system administration, and customer support.

All the elements were organized in sub-teams according to their roles, each having a specific leader, and all performing highly specific work, although intrinsically related and interdependent between teams.

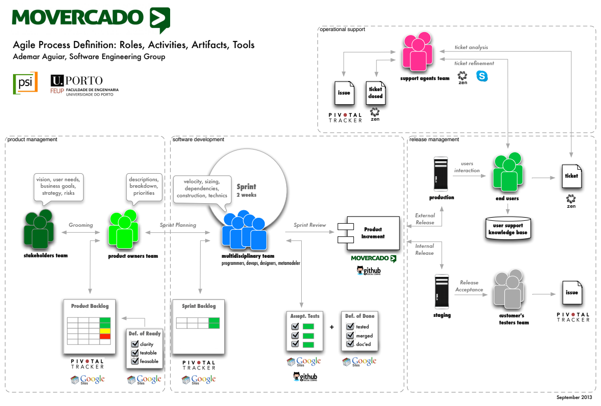

Figure 2: MOVERCADO process: roles, activities, artifacts, tools

To properly support an efficient and effective knowledge creation process (the so called "spiral of knowledge"), a small combination of tools was adopted, which included a wiki, a project management tool, a software forge, and a user support system. Beyond these, cloud services were used to share some documents and video-audio-and-chat messaging rooms and email were used for fast synchronous and asynchronous communication.

Although it proved to be a valuable toolset, there were some barriers regarding contextual knowledge classification and user contribution, specially in the interfaces of the distinct teams, roles, and activities, and also in the interconnection of the individual tools, where some knowledge got lost and became disconnected. The authors suggested, as an improvement, to lean from a traditional wiki towards a semantic wiki, for example, and integrate it into the development environment, planning tool, and content authoring support using more social collaborative features.

6. RELATED WORK

Regarding software knowledge management, candidate tools and techniques may range from evolutionary software forges [7] [3] or weakly-typed wikis [8] to collective knowledge systems [10][9] or collaboration-enhanced IDEs [5] [2] [1]. Their impact on parts or the whole development process is yet to be assessed regarding the SECI knowledge cycle.

7. CONCLUSIONS

As a conclusion, we claim that the "Ba" concept applied to software development may help uncover points of inefficiency in terms of software knowledge evolution and that there are still room to improve its support through proper tool support.

There are still many issues that arise and that should be dealt with in forthcoming research, such as: How does the "Ba" concept fares when compared to other knowledge transfer models like minimalist documentation [6] or SLEs [12]? Are there specific differences regarding software knowledge at different levels of abstraction (e.g., architectural knowledge vs. code knowledge) when applying the SECI cycle?

In future work we aim to envision a toolset, an environment, or set of tools that might cover the still unsupported "holes" in "Ba"'s applied to software.

8. REFERENCES

[1] Collide. https://code.google.com/p/collide/ [Online; accessed March 2014].

[2] Devtable. http://try.devtable.com [Online; accessed March 2014].

[3] Github. http://github.com [Online; accessed March 2014].

[4] Movercado: a mobile-techonology based ecosystem for efficient aid distribution. http://enter.movercado.org [Online; accessed March 2014].

[5] Rational team concert. http://www-01.ibm.com/software/rational/products/rtc/ [Online; accessed March 2014].

[6] A. Aguiar. *Framework Documentation – A Minimalist Approach*. PhD thesis, University of Porto, Faculty of Engineering, 2003.

[7] F. Correia. Supporting the evolution of software knowledge with adaptive software artifacts. In *SPLASH'10, Reno/Tahoe, Nevada, USA*, 2010.

[8] F. Correia, H. Ferreira, N. Flores, and A. Aguiar. Incremental knowledge acquisition in software development using a weakly-typed wiki. In *WikiSym - 5th International Symposium on Wikis. Orlando, Florida, USA*, 2009.

[9] N. Flores. *Patterns and Tools for Improving Framework Understanding: a Collaborative Approach*. PhD thesis, University of Porto, Faculty of Engineering, 2012.

[10] T. Gruber. Collective knowledge systems: Where the social web meets the semantic web. *Journal of Web Semantics*, 2007.

[11] P. Grunbacher and P. Briggs. Surfacing tacit knowledge in requirements negotiation: experiences using easywinwin. In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001.

[12] J. Hart. Intro to social learning environments: A social learning resource. http://janeknight.typepad.com/socialmedia/2009/10/intro-to-social-learning-environments-a-social-learning-resource.html [Online; retrieved June 13, 2013].

[13] I. Nonaka. *Knowledge Management - Critical Perspectives on Business and Management*. Routledge, 2005.

[14] I. Nonaka and N. Konno. The concept of "ba": Building a foundation for knowledge creation. *California Management Revies*, 40(3):40–54, Spring 1998.

[15] D. Roberts and R. Johnson. Evolving frameworks: A pattern language for developing object-oriented frameworks. In *Proceedings of the Third Conference on Pattern Languages and Programming*. Addison-Wesley, 1996.

[16] J. Whitehead. Collaboration in software engineering: A roadmap. In I. C. Society, editor, *Future of Software Engineering within the International Conference on Software Engineering*, pages 214–225, Washington, DC, 2007.