KNOWLEDGE BASED OPEN SOURCE PROJECT MANAGEMENT

PhD. Student **Radu BUCEA-MANEA**
PhD. Student **Rocsana ȚONIȘ**
The Bucharest Academy of Economic Studies

ABSTRACT

This article focuses on the Open source phenomena, as a result of the activity of some top laboratories(Bell Labs, MIT AI Lab, UC Berkeley), and than by the quasi-illegal know-how dissemination through hackers and Internet. We refer the first work known by the community "The cathedral and bazaar" by Eric S. Raymond, which makes a synthesis of the open source development process. On the other side the paper presents the scientific principles of agile programming, trying the efficient corroboration of these methodologies in the development of a new paradigm in the Open source projects development process. The synergic effect of the "many eyeballs" principle is similar to the goals of KM that breaks the formal boundaries of the corporatist model in order to reach new levels of efficiency on the basis of good team work and mutual understanding.

KEYWORDS: *Open source, UNIX, Linux, Agile*

1. Introduction to KM

According to classical theory of the firm, it mobilizes resources to achieve collective goals, tools to meet personal objectives. Promoters of the new paradigm of knowledge-based organization tend to minimize the influence of opportunistic behavior or niche markets in developing organizations. They say applied knowledge is the main factor responsible for giving the competitive advantage. This factor is materialized in those custom organizational and language skills that facilitate the progress of the organization.

The work on knowledge, combined capabilities and Kogut's replication technology, abolish theory of transition costs by emphasizing the primacy of knowledge in the creation process and use them. The new theory, firms exist because the market fails to coordinate the skills of individual specialists who hold knowledge only. From this it follows that the first role of business is to use knowledge provided by experts in their capacity as employees.

Theorists like Nonaka [3] and Toyama tend to describe the firm as a dialectical entity divided between organizational structures and individuals in the continuing process of accumulating and creating knowledge through directed use. If the classical view, employees were motivated from the outside as prescribed contractual arrangements the company today, individuals must consent to the use of inner motivation and distribution of knowledge we have in the organization. They are differentiated two main forms of knowledge, tacit and explicit, and the process through which it passes from one form to another is socialization. Transition process of knowledge is described by the following steps:

S_1. outsourcing, the process by which new knowledge is created based on tacit knowledge;

S_2. combination, which allows the accumulation of tacit knowledge based on explicit knowledge older;

S_3. internalization, a process that enables the creation of new knowledge based on explicit knowledge created.

Defined by [2] that all collection methods, verification, transmission, storage and data processing, information systems enable the use of knowledge in all hierarchical levels within an organization and accumulation of knowledge in databases or file servers throughout life company. After the same author, the computer system was in a relationship part to the whole information system and is defined as all hardware and software resources, communication networks, users and information.

2. The cathedral and bazaar or vice-versa?

The first modern operating system realized on scientific bases by official researchers was UNIX. Its success led to the invention of the strong programming language C in the Richie-Kernigan release and a tradition in programming that became culture with all its myths and exaggerations.

The lessons of this effort in development were synthesized by the author of bazaar (http://www.catb.org/~esr/writings/cathedral-bazaar/) as follows:

- ✓ small utilities but with precise functionality;
- ✓ using prototypes;
- ✓ iterative development cycle.

UNIX is holding also on other principle such as the trend of programs to behave like filters (STDIN, STDOUT and STDERR), pipeline communication, accessing the core in system gateway through primitive and user libraries, these are some percepts that are the base of almost all modern systems. Eric Raymond underline those characteristics of the UNIX development process that decisively influenced the evolution of Linux bazaar community, with task agenda and different approaches. Linus Torvald style is also characterized by launching frequently new release, delegating all the responsibilities that merge from the development cycle and the opening to chaos, being a surprise for that time.

In (Raymond, 2001), the author refers at the waste of young talent in participation to the cathedral closed type projects for reasons of necessity, an important feature of the very best programmers is focus on results, always more easily obtained otherwise than from scratch. Linus, for example, did not try to write Linux from the beginning. He began by re-code and ideas from Minix, a Unix-like operating small, written by Andrew Tanenbaum for teaching purposes at the Department of Computer Science, Faculty of Sciences in Amsterdam (http://www.cs.vu.nl/~ast/).

Another strong point of the Unix tradition is that many users are hackers, too. This can be extremely useful to shorten the time error correction. With little encouragement, users will find problems and if they would gain access to the Internet, they would have suggested solutions and would have helped improve source code much faster.

Linus maintain his employees through permanent motivation and reward - stimulated by the fact that their work counts and that satisfies their ego, rewarded by following enhancements result of their work. With a large number of beta testers and co-developed, almost every problem will be identified quickly and will have an obvious solution for anyone. Informally, "with enough eyes, all bugs are reduced". This is "Linus's Law."

In the cathedral style, bugs and development problems are deceptive, hidden and profound. It takes months of searching for the few developers to say that all bugs were eliminated, hence the delays in launching new versions and the inevitable disappointment when they do not appear perfect.

Sociologists have discovered years ago that the masses average opinion as expert observers (or ignorant observers) is a better predictive factor than one observer randomly chosen. They called it "Delphi effect". It seems that Linus has shown that *Delphi effect* is also applied for debugging an operation system, namely the effect explain the complexity of development even in the complexity of a core (kernel).

Jeff Dutky, developer with experience at Blackstone Technology Group, San Francisco, California (http://www.dutky.info/jeff/resume.html) observed that the total cost of maintenance of a widely used program is typically 40% or more of the development cost. Surprisingly, this cost is strongly affected by the number of users in the sense that many users find more bugs. Each person approaches the task of finding defects with slightly different conceptual and analytical tools, under a different angle of problem approach.

The author considers that the best programs begin as a solution to personal every day developer's problems and had spread to the issues proved to be typical for a large class of users.

Fred Brooks, a researcher in computer science and Turing Award winner in 1999 (http://en.wikipedia.org/wiki/Fred_Brooks), noted that the time of the programmer is not replaceable and adding developers to a late project makes it delays more. He argued the fact that complexity and communication cost in a project increases with the square of the number of developers, while the amount of work grows only linearly. Since then, this idea became "Brooks's Law" and is generally considered a truth. The same scholar said in his work "The Mythical Man-Month: Essays on Software Engineering", that there is no panacea against errors occurring accidentally in software development projects or that "It is a very humbling experience to make the multi-million-dollar mistake, but it is also very memorable."

Open source developer communities manage to refute Brooks's law through an unselfish participation in this kind of projects and implementing the "principle of understanding", quoting the 'Memoirs of a Revolutionary', Kropotkin's autobiography, geographer, zoologist and one of the foremost Russian anarchists in the nineteenth century (http://dwardmac.pitzer.edu/Anarchist_Archives/kropotkin/kropotkinarchive.html). Linux was the first project that made a continuous and conscious effort to use the World Wide Web as a source of talented programmers. While programming remains an essentially solitary activity, really important programs engage the attention and the power thinking of entire communities.

Linux community behaves in many ways, as a free market or an organic/ecologic system, a collection of selfish agents trying to maximize utility, which produces a self-corrected spontaneous order, more elaborated and efficient than could be done by any central planning. World trade cannot beat a growing open source community that mobilizes in time more talented groups to solve a problem.

Our conclusion is that development evolutionary model has precedence over so-called classical proprietary model and at present the exception rule is closed development model. In the second part of the article it is presented the agile development model, implemented by different methodologies (XP, Scrum), that is similar with "bazaar" model, summarized above.

3. Agile methodologies in developing ICT projects

CMMI - Capability Maturity Model ® Integration (CMMI ®) - is an approach used to improve both work processes and resulting products and services. CMMI for Development is at version 1.2 (August 2006) and consists of 22 process areas with 5 levels

of maturity. Each area (e.g. area: Project Monitoring and Control) is a series of requirements that can be covered by:

- ✓ the company internal working method;
- ✓ the method inspired from traditional project management;
- ✓ Agile methods class.

CMMI should be adapted by each company to help achieve business tasks. CMMI was created and is maintained by a team of members from industry, the U.S. government and the Software Engineering Institute (http://www.sei.cmu.edu/).

According to (Schuh, 2005), Agile development is a method of building software by empowering and trusting people, recognizing change as a rule, and to promote constant feedback .(...) Processes, tools, plans, documentation and contracts must be adapted and applied as needed to achieve this objective.

After Parvulescu in http://business-edu.ro/Business/Software-Development/ viziune-strategie-si-succes-in-software-development.html, adopting Agile tries to minimize risk and maximize productivity. Software is developed in short iterations (maximum 4 weeks) to obtain interim versions fully functional (working software). Thus application grows, every iteration adding new features or refining some existing ones.

Functions are implemented depending on the value of their business and depending on current needs of the client. Working method adapts to the needs and the project team; learning from the past iterations mistakes and the exposure of the existing problems are two mechanisms that help enhance team performance. Transparency, communication and collaboration participants are behind the success of Agile projects.

The advantage of Agile is given by obtaining a software work very early in the production process. Working in short iterations, correction can be done depending on contextual changes and depending on the feedback obtained. Good results are obtained both in dynamic and where requirements are stable (http://business-edu.ro/Business/Software-Development/viziune-strategie-si-succes-in-software-development.html).

The main agile methodologies are:

- ✓ Adaptive Software Development (ASD);
- ✓ The Crystal Methodologies;
- ✓ Dynamic Systems Development Method (DSDM);
- ✓ Extreme Programming (XP);
- ✓ Feature-Driven Development (FDD);
- ✓ Lean Software Development;
- ✓ SCRUM.

According to Diaconu in http://business-edu.ro/Business/Software-Development/provocarea-unei- schimbari-adoptarea-agile-i.html, software companies are facing performance problems and the economic environment is not favorable. Analyzing a company's study 'The Standish Group' made in 2000 on a sample of 35,000 software applications, we see that 49% of software projects did not meet the time and on budget, and only 23% felt completely, 28% met the time and budget requirements.

There are many factors leading to failure of software projects, but some of the major challenges are:

• most functionality in a software application are not used at all - Standish study shows that approximately 75% of features are used rarely, very rarely or not at all;

• good ideas that come throughout the project, and 'late' features required by customer, leads to a 're-work' too big to be included in the project, which is actually an obvious loss in value added product,

• can not provide from the beginning of the project all problems or all the opportunities, so planning and design in excess not necessarily lead to a successful project; in addition there is the risk to remain stuck in the analysis phase if nothing is materialized after the early period;

• an enormous amount of time spent to correct the 'bug' and other types of problems after the product was finished. (http://business-edu.ro/Business/Software-Development/provocarea-unei-schimbari-adoptarea-agile-i.html)

Agile values after (Schuh, 2005), are:

V_1. Encouraging personality and inter-personal relationships;

Agile sets the focus on the professionals' integration in project teams as team based on skill, individual effort in their synergy to obtain a successful product. To get a successful project, a team must select and implement a set of processes and tools that fit the environment, the technology and the purpose.

V_2. Set the focus on code in detriment of project documentation;

Documentation may encourage continuous software delivery, but excessive documentation too early in the development cycle, may kill a project, i.e. when updating documentation becomes more important than writing code.

V_3. Collaboration with customers is more important than contract negotiation;

Agile Development presumes that IT professionals work in teams to write software to customers. Tools and documentation implemented both within and outside the development team can remove a project from its purpose. Often things go bad because the project team is too busy to work on the contract letter. Agile teams follow practices that keep them focused on their customers needs. Agile Development supports only the actual contractual relationships that encourage the project team to work with and for its clients.

V_4. Reacting to changes and less to the initial plan.

Agile Development emphasizes practices that enable teams to adapt to the requirements and environment changes. Short and iterative feedback is an essential component of many of these practices. They help provide the team personnel (whether members of the development, management, or business) information necessary to make decisions in time. Flexible and high quality code is an essential goal, among many agile practices. A rapid response to change is only possible when developing project may be subject to change. Finally, when the requirements and environment change, plans must follow.

4. Conclusions

Romania has more to do concerning knowledge based project management, but the increasing number of hackers raises the potential of the Open source community so the synergic effect of the "many eyeballs". Also many companies applies Agile methods even without knowing it, admitting working from home, quick version release and iterative developing based on the feedback received from users. Accordingly to (Nicolescu, 2009), over 80% of SMEs are equipped with computers, 81.44% of companies use the Internet and over 70% of firms use e-mail. The majority of successful companies are managed by entrepreneurs with university education, thus increasing the potential of applied knowledge management in developing process.

References

1. Ovidiu Nicolescu, Alexandru Isaic Maniu, Florin Anghel – *Cartea alba a IMM-urilor din Romania*, Lidana Publishing House, Bucharest, 2009, ISBN 9789737679819, 414 pages
2. Eric S. Raymond - *The cathedral and the bazaar: musings on Linux and Open Source by an accidental revolutionary*, O'Reilly, 2001, ISBN 0596001088, 9780596001087, 241 pages
3. Ken Schwaber, Mike Beedle - *Agile software development with Scrum Series in agile software development*, Pearson Education International, 2008, ISBN 0132074893 9780132074896, 158 pages
4. Peter Schuh - *Integrating agile development in the real world*, Cengage Learning, 2005, ISBN 1584503645, 9781584503644, 346 pages
5. http://www.catb.org/~esr/writings/cathedral-bazaar/
6. http://www.cs.vu.nl/~ast/
7. http://www.dutky.info/jeff/resume.html
8. http://en.wikipedia.org/wiki/Fred_Brooks
9. http://dwardmac.pitzer.edu/Anarchist_Archives/kropotkin/kropotkinarchive.html
10. http://www.sei.cmu.edu/
11. http://business-edu.ro/Business/Software-Development/viziune-strategie-si-succes-in-software-development.html
12. http://business-edu.ro/Business/Software-Development/provocarea-unei-schimbari-adoptarea-agile-i.html