Towards an Open Source Software Factory

Ana Paula Carvalho Cavalcanti, Leonardo Reis Lucena, Márcia J. N. Rodrigues Lucena, Alan Kelon Oliveira de Moraes, Damires Fernandes, Silvia Cássia Pereira, Jones Albuquerque, Silvio Romero Lemos Meira

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE– Brazil
{apcc2, lrl, mjnrl, akom, dysf, scp, joa, srlm}@cin.ufpe.br

Abstract. *Facing the reality of today's software development within the Open Source community, it is a substantial fact that an undefined methodology and practices are the core discipline that guides the community members. Around these assumptions and thus the great amount of good efforts available world wide, this work is an attempt to define a software factory structure to better comprise development tasks and developers abilities. In accordance to this, it is presented the evaluation of the joint of these two paradigms, Open Source Development and Software Factory, applied in the development of an open source software factory.*

Keywords: *Open Source Software, Software Factories*

1. Introduction

The Open Source Development is a movement that has been gathering a significant amount of interest around the software engineering society, specific by its way of organization and development of solutions.

The concept of software factory appears as an intended organization with defined methodology grouped by collaborators that target the development of software according to a specific set of requirements. These requirements are defined by clients or stakeholders, which are done in order to define the scope of a project.

According to these assumptions, a challenge is launched with the intention to supply the demand of software: how to link the efforts available in the open source community to the reality of a software factory? Not only this, but the evaluation whether this approach is applicable to the reality of the software market and how to define a business model to an organization.

Therefore, this paper presents an overview on Open Source and Software Factories to better illustrate the experience based on the OXE Factory Project [OXE 2005], Open Experience Environment. This is an endeavor to perform an experimental study to join these two paradigms and validate if these distinct areas can go along together.

In the next section, Open Source characteristics will be discussed, followed by an overview on Software Factories. Then we present the Open Source Software Factory Paradigm illustrated by OXE's experiences and community interaction issues. Finally, we present prospects and conclusions to this work.

2. Open Source

The expression *Open Source Software (OSS)* was coined in 1998 [Perens, 1999] to define software that may be freely distributed, has the source code included, and allows the redistribution of derived works. The definition of OSS is quite the same of Free Software Movement's definition of Free Software. The difference is more philosophic than pragmatic. Free Software sees source code sharing as a moral and ethical aspect while Open Source Movement sees source code sharing as a means to achieve technical excellence. Recently the expression *Free/Libre/ Open Source* Software (FLOSS) came out to refer to both Free and Open Source Software.

In his classical essay, *The Cathedral and the Bazaar* [Raymond, 1999], Eric Raymond explains the model of software development used by OSS. The Bazaar model states that OSS, in contrast with the traditional ones (Cathedral), must be developed in a decentralized environment without predefined roles.

Some patterns are usually present on software development using Bazaar [Raymond, 1999]: users are treated as co-developers, the availability of early releases and several versions of the software products, frequent integration to avoid work overhead, high level of modularization and dynamic decision-making structure.

In the scope of OXE Factory's definition, OS community behavior plays an important part on the business. Community interaction may lead to Software's technical excellence but may also have a strong impact on traditional software factories structure.

3. Software Factories

A software factory is an organization or a department that has software development as its end. The mature practices of Software Engineering disciplines made possible the arising of Software Factories, where their concepts remote to 1969 and Bemer recommended the adoption towards the increase of programmers' productivity [Bemer, 1969].

Hence, software factories provide services of software development with quality, low cost, and in a fast way. In addition, they use a defined development process together with the market's technologies, besides recognizing and dealing with process improvement opportunities [Siy, 1999].

A typical Software Factory has the following characteristics: an organizational structure where people plays roles related to the organization's activities; use of process, tools and methodologies; customers that define the scope and sponsor the software development; metrics and reuse policies to improve the development process and quality. Moreover, there are software product lines that define reusable artifacts in a specific domain, with the purpose to be used in different projects to achieve a desirable level of productivity.

It is seen that a software factory has a established structure that is continuously being enhanced over the years [Cusumano, 1989][Aaen, 1997][Greenfield, 2003]. Such principles are the foundation for the experimental study of OXE Factory.

4. Paradigm Shift

A paradigm is usually characterized as a set of practices that must consider what is observed and how to solve specific ideas in a determined field of study. In the course of the study, unexpected discoveries may prove inconsistency with prevailing paradigms, what would cause raise of a new scientific activity around a certain domain [Sametinger, 1997].

In software engineering, a paradigm shift involves the application of new concepts, ideas, methods, and problems. Upon this belief, an experimental study brought to OXE Factory was whether the joint of the OSS Development and Software Factory would cause a design on a new paradigm of software development.

The experimental method [Travasso, 2002] suggests a model, develops a qualitative and/or quantitative model, applies an experiment, measure and analyzes, evaluates the model and repeats the success. In consequence, the steps followed in the context of this work began with the proposition to develop an open source software factory, along with the establishment of a structure and the application of an experiment based on real project [Vensso, 2005] so that the analysis and measurement of the results could be possible.

The organization of the structure included the definition and evaluation of a business model to comprise the distributed development and open source community together with a software development process.

Upon the related experiment, paradigm shift can be seen by two viewpoints: OSS development with a client and Software Factory developing OSS. Both sides can take advantages of the union. Open Source community can establish a factory to develop services and software factories can develop open source software as strategies of its business model.

In principle, it is hard to believe that a software factory that develops property (closed) software will open its source code. The paradigm shift occurs when the software factory sees the development of OSS as part of its business model and the possibility to make money developing OSS. There are clients that pay for OSS development.

The focus in not to have the community developing software for the factory. Community members cooperation is seen as a mean to enhance software's quality. Their typical role is to test the software, submit patches and report bugs. When a collaborator does more than these activities, she/he is a strong candidate to be hired by the factory.

The development of OSS can also be a client's requirement. Government departments, for instance, commonly have as goal make their acts transparent. This can be achieved by keeping their source code open.

Therefore, software factories developing OSS does not mean a cathedral inside a bazaar. It considers using OSS practices in a factory with the motivation to achieve technical excellence of factories' products.

5. Community Interaction

In accordance with these perspectives, apparently antagonist concepts of software development, the challenge to join both worlds is seen as an attempt to be evaluated. As shown in [Deng, 2003], it is necessary to validate the commonalties between these paradigms to propose a model of integrations.

In the scope of OXE Factory, the necessity raised from a client's proposal to develop open source software. From this point, OXE started as a conventional Software Factory that had to be adapted to best fit the business requirements and therefore, evaluate how to apply software factory's paradigm in the open source community world.

The practice followed by the establishment of a software development process, along with roles definition, for both factory and project, business model and specific guidelines and policies to integrate the open source community into the factory's structure.

In the designation of the factory's business model, a key area, and probably the most obscure to be solved, was the interaction of the factory with community collaborators. The challenge was how to make business out of a factory inserted in the open source world. This consideration lead OXE Factory to make strict assessment of the external and internal environment, critical success factors, market vision, services and products, social and economic impacts and how human resources will be considered in the factory's scope. The idea of how community members would engage in the project is showed n in figure 1.

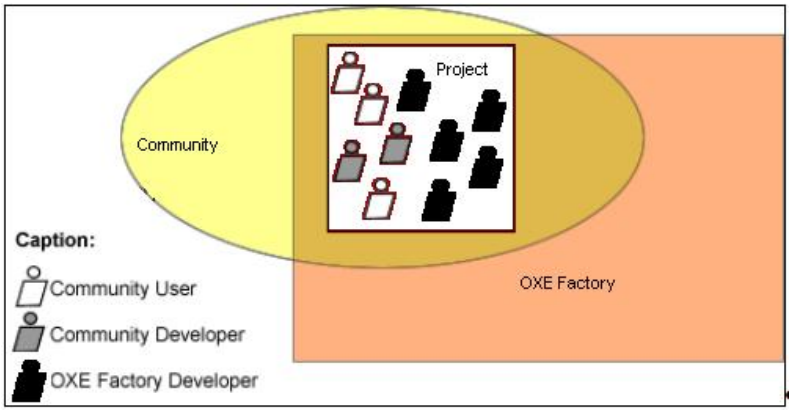

Figure 1. Community Interaction Model

Community interaction is done in a way that the project belongs to the community and becomes an intersection point with OXE Factory. Collaborators are welcome to interact directly with the developers, responsible for the execution of a pre-defined task. The factory follows the process and community can randomly collaborate, and the responsible for a task is the mean to receive contributions from the Open Source Community.

OXE factory's policies propose interaction procedures that include the definition of communications channels, new member's acceptance, and general guidelines, along with all relevant political information with respect to the interaction of the open source community and its projects.

As an immediate advantage of this model we see that the software will be broadly used by the volunteers that will provide us constant feedback and suggested improvements to be made. Consequently, by the end of the process, mature and high quality software will be delivered to the client and the Open Source Community will continue its development.

6. Conclusions and Prospects

According to the facts presented, the main goal of OXE Factory is to become an open model of software factory. It is necessary to shape both ideas and concepts in a common effort to best define a structure of software development.

This article discusses how can OSS and Software Factories be integrated, a necessity that may arise from clients or as part of the development model. We focus on how the open source paradigm can be used, showing the model defined and advantages of this approach: distributed work, community integration, and quality on the products delivered to the community.

The validation of this belief was conducted by the establishment of an open experience environment (OXE) which is being applied together with a real project to a specific client [Vensso, 2005]. The intention is to validate and evaluate the effects of a new paradigm application, in order to interpret the results of this scientific investigation.

After the first cycle of evaluation, marked by the end of the project, we could state the difficulties and lessons learned of the proposed model, which mainly focus on the difficulty to manage a project on the open source community and maintain the communication between members. Nevertheless, the deal of intellectual property on the work produced was also an odd occurrence that caused a great amount of effort to solve the problem along with the client. We could also see the necessity of a software product line definition. OXE Factory established an infra-structure that support Web applications using Rapadura Framework [OXE 2005].

Another point that must be considered is the assessments of the conflicts management between both paradigms. Nevertheless, the practice of an Open Process is also being applied, where IXI Process (The Process defined to OXE Factory) is considered a project itself and community contributions and interactions are being experimented and evaluated in the experimental scope.

As a planning for the consequent steps, one of the objectives of OXE is to evaluate the establishment of a software product line in the next generation of projects development, so that a more precise analysis can be achieved along with reuse metrics collections.

Acknowledgments

Our special thanks to Carlos Eduardo Lima, Clélio Feitosa, Euclides Arcoverde Neto, Lucas Schmitz, and Severino Andrade, OXE members.

References

Aaen I., Botcher P., Mathiassem L. (1997) “The Software Factory: Contributions and Illusions”. In proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo.

Bemer, R.W. 1969. "Position Papers for Panel Discussion--The Economics of Program Production". Information Processing 68, Amsterdam, North-Holland.

Cusumano, M. A. (1989). “The Software Factory: A Historical Interpretation”. *IEEE Softw.* 6, 2 (Mar. 1989), 23-30.

Deng, J, Seifert, T, Vogel, S.(2003) “Towards a Product Model of Open Source Software in a Commercial Environment” In 3rd International Workshop on Open Source Software Engineering. ICSE, May 2003.

Greenfield, J. and Short, K. (2003). “Software factories: assembling applications with patterns, models, frameworks and tools”. OOPSLA '03. ACM Press, New York, NY.

OXE (2005) “OXE Open eXperience Environment Factory”, http://oxe.sourceforge.net, 2005.

Perens, B. (1999) “The open source definition. in Open Sources: Voices from the Open Source Revolution” C. Dibona, S. Ockman, and M. Stone, Eds., O’Reilly, Sebastopol, Calif., 171–188.

Raymond, E. S. (1999) “The Cathedral and the Bazaar”. 1st. O'Reilly & Associates, Inc.

Sametinger, J. (1997) “Software Engineering with Reusable Components”. Springer-Verlag.

Siy, H. P., Herbsleb, J. D., Mockus, A., Tucker, G. T., and Krishnan, M. 2001. “Making the Software Factory Work: Lessons from a Decade of Experience”. In Proceedings of the 7th international Symposium on Software Metrics (April 04 - 06, 2001). METRICS. IEEE Computer Society, Washington, DC, 317.

Stallman, R. (1999) “The GNU Operating System and the Free Software Movement” in Open Sources: Voices from the Open Source Revolution, O’Reilly, Sebastopol, Calif., 53–70.

Travasso, G., Gurov, D., Amaral, E. (2002) “Introduction to Experimental Software Engineering”, Technical Report – RT-ES-590/02, COPPE, UFRJ, Brazil (in Portuguese).

VENSSO (2005) “Selling Service and Software”. www.vensso.sourceforge.net