Open Source Software Factory – Step by Step: A Case Report: Alan Kelon Oliveira de Moraes (Centro de Informàtica – UFPE, Brazil)

Alan Kelon Oliveira de Moraes, Silvio Lemos Meira, Jones Oliveira de Albuquerque
Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – 50732-970 – Recife – PE– Brazil
{akom, srlm, joa}@cin.ufpe.br

Abstract

In the last years, it has been realized an increasing movement around open source software development where the team is distributed and usually uses a very light process. The challenge to be overcome is how to enable software engineers to work in a distributed software factory in accordance to best practices applied in the open source model. This work presents the steps in building a distributed software factory together with the results gained in a real experience conceived at Open Experience Environment (OXE) Software Factory.

1. Introduction

In the early years of computer science, software was not seen as a product and software development was not conducted by a process either [14]. Nowadays, it's realized the necessity to apply a development process to gain software with quality, decrease costs and time [15] and also reduce the risks that huge projects involving a big number of companies end up in failure [20]. At the same time, successful open source projects encourage companies to search for the integration of this development model with its own projects [16].

The research question is how to enable software engineers to work in an open source software development environment which is usually distributed and characterized as a virtual organization [3]? On the other hand, real software organizations aim to professionalize their operations working like a factory, where productivity is taken by standardized methodologies and tools that provides coordination, systematization and formalization [1].

The focus of this paper is to show which steps are necessary to create an Open Source Software Factory based on Open Source Software Development (OSSD) best practices and Software Factories principles. We present a real experience conceived at Open Experience Environment (OXE) Software Factory (http://www.oxe.org.br) [24]

The paper proceeds as follows. We start with the necessary background discussion defining what is a Software Factory and defines some background on Open Source Software Development model. Then we present the steps to create an Open Source Software Factory based on OSSD model and finally we consider some conclusions and prospects.

2. Software Factories

The term software factory labels development facilities with formal approaches to software development [28]. Bemer labels a software factory as software development facility aimed to reduce variability in programmer productivity through standardized tools and management controls [29]. A common mistake is to compare the term software factory applied to software development context with the understanding of the term in other industries, which is associated to generally mass-produced products including large-scale centralized operations, standardized and unskilled job tasks, standardized controls, low skilled workers, mechanization and automation. A software factory aims to standardize good practices, gradually improve tools and techniques, and strategically integrate their efforts with rigorous employee training [28]. Indeed the overall concept is to have a more structured approach for software development, emphasized code reuse as well standardization and control over tools and processes

The concept of a factory also implies a particular way of organizing work with considerable job specialization, formalization of behavior and standardization of work processes [1].

Building a software factory means gathering a group of factors. Harvey and colleagues [6]

suggest an approach to do so: to define a detailed software development process; to give extensive training in the new process to staff members; to specify the process specification separated from process execution; to collect quantitative and qualitative data about project execution (interviews, software process assessments, process attributes for each project, configuration management system, project tracking data) and analyze the results to try to find out opportunities for software process improvement.

What is really necessary to build a software factory depends on each situation, but in general there are two essential questions to be considered: a suitable infra-structure environment to work and a defined software process development. The professional competence of factory members must also be taken into account.

3. Open Source Software Development Model

The basic Open Source Software philosophy [8] [10] is extremely simple: when programmers are allowed to work freely on the source code of a program, code improvement is realized because collaboration helps to correct errors and enables adaptation to different needs and hardware platforms [9]. In fact this has happened and Open Source software is well known today because of its high degree of reliability and portability [17][18]. In this new approach, the hierarchically organized structure adopted in all productive processes is put aside in favor of a new kind of bottom up structure, which is non-coercive and largely decentralized [7] [9] [27].

According to Fuggetta[4], many of open sources practices are also applicable in proprietary software development. The open source just leverage some practices to the extreme. His analysis does not cover one of the main points in open source success: the distributed development. Some questions are extremely important: What happens in the OSSD model that works so well? Why do not IT corporations have the same kind of successful results? [11]

The essence of the OSSD model is the rapid creation of solutions within an open, collaborative environment [9] [30]. Despite the many challenges associated with geographically distributed development [19], the OSSD model relies on collaboration and agile development practices. Key practices of the OSSD model include: projects start quickly; requirements are defined as soon as possible; development is distributed among contributors in a collaborative way; development cycles are short, fast and iterative; reuse existing code from similar projects; developer's multi-project flexibility.

By using collaborative development processes and tools for managing projects issues, communication and code, the OSSD model allows virtual teams to produce quality results. To do so, it is necessary tools manage all project activity throughout the application development life cycle, the collaboration on documents and design models, to track project issues, and to conduct structured and traceable project communication.

Some practices should be adopted in traditional software development. There are at least three areas in which open source practices can benefit corporate IT organizations: team communication, user involvement and staffing models [2]. The companies should adopt agile processes being proven on open source projects. Corporate IT teams should also evaluate the following practices that are common to open source and agile communities: release software early and often, formally involve users in development, enable collective code ownership, practice continuous integration and employ automated testing to source code [9] [21].

4. Step by Step

There is no standard way to create a Distributed Software Factory based on OSSD model. Therefore we discuss general issues to consider in designing this kind of software factory. The presented steps are based on OXE Software Factory experience, an open source software factory run last year during a Software Engineering (IN953) graduate course at Federal University of Pernambuco, Brazil [22]. IN953 exposes students to real, team-oriented development in a software development organization staffed and managed by students under the guidance of the faculty. Several students are professional developers, certified programmers and work in industry, too. These courses are hands-on courses that require student participation in one of the factories defined. The projects for each software factory are chosen by professors and software factory managers. The demands are characterized by Request For Proposals (RFP) and have one client per project. These projects are in collaboration with C.E.S.A.R (http://www.cesar.org.br) and its partners which reflect current trends in industry and makes its professionals (which are students in the course) motivated [23].

Step 1. Define the Factory Business Model

The factory intends to be an organization inhabited by developers engaged in the common effort to develop open

source software. But, what which business models are suitable for this configuration/environment? What kind of services should be provided? What about real projects with real clients? The factory must define some strategic way to invite community's developers to collaborate in its projects and, at the same time, the factory must consider clients wills and how to manage their requirements.

OXE Software Factory's strategies include calling community's attention to the software project, as soon as possible, by releasing the first prototype in the open source community, e.g. VENSSO (http://vensso.sourceforge.net), the product developed last year. Open source community developers will be engaged in the product lifecycle, attempting to spread coding activities. We work with GPL license and provide additional services around our products. We aim to work with software product lines to better organize our development and to boost our productivity through reuse [25].

Step 2. Define the Factory Organization

The software factory is supposed to define the responsible members for the organizational management. OXE has Management Committee, Sales, Research and Development, Products, Finance, Component Library, Quality and Project divisions. Each one is composed by PhD and M.Sc. students from Federal University of Pernambuco, Brazil.

Step 3. Define a Lightweight Development Process

A software factory needs a configurable software development process platform based upon proven best practices [1]. The process must be flexible and must enable what is necessary to be constructed for each stage of the project. It is necessary to think about a lightweight process that can be applied without obstructing the natural stages taken by software factory's team and the open source community developers.

OXE has defined the IXI Process which focuses on the best practices of Software Engineering, based on the Unified Process [5] and the agile practices such as Extreme Programming [21], to better coordinate the activities and concentrating on the organization's objectives: deliver the desired product to client. IXI Process is iterative and incremental, followed by five phases with defined milestones and objectives. Each phase is composed by activities that result in some artifacts necessary to go on the process.

Step 4. Enable the Work in a Geographically Distributed Way

A distributed software factory is intended to work as a virtual organization. That means the adjustment of three points: the existence of a shared goal by the team members, geographical distribution and the use of Internet services to manage and coordinate the interdependences [3]. These factors characterize OSS communities and must be considered in open source factories as well.

As factory team members are geographically distributed, the Internet is the main tool to coordinate activities and actions. That means a distributed and collaborative development environment comprising communication, management and peer revision tools. Asynchronous communication through mailing lists is the main communication channel because developers are distributed world wide and we cannot guarantee synchronous availability. Mailing lists provides a way for project members and interested people to communicate with each other any time. This tool allows project discussion, improves knowledge management and sharing among developers and enables the product self-documentation.

An issue tracker helps to manage the project progress, tracking many different kinds of activities in the projects, including bug reports and feature requests. The assets must be under a control revision system [26] to avoid effort duplication, to solve possible conflicts between two different versions of the same document, and to manage and control source code interdependences.

Step 5. Provide a Web Site for the Factory

The software factory is intended to provide its own web site where clients and community may surf. The factory site must aggregate information about its business mission, development process, news, team members, projects and solutions. The factory web site is also a point to centralize definitions and coordination.

OXE Factory publishes its site at http://www.oxe.org.br. OXE Software Factory is itself open which means that collaborators are welcome to the factory. These collaborators are invited to contribute on both software products and in the IXE process improvement.

Step 6. Provide an Exclusive Web Site for Each New Project

The factory has its own site, but for each new project another site must be built. Therefore, each new project will have its site with its instantiated process, templates, artifacts resulting from each development phase, downloads, news

and other project intrinsic characteristics. The aim of each project website is to allow open source community factory members to follow the development process. One of our projects may be found at http://vensso.sourceforge.net/.

Step 7: Define Roles for Each New Software Project

For each new project, development roles are necessary, but they may change depending on the project domain or software requirements. Thus some roles may be defined according to each project, such as: project manager, software architect, software engineer, configuration engineer, test engineer, requirements analyst, database designer, and others.

Step 8: Team Members Must Work in Harmony

When working in a distributed software factory, team members are supposed to learn some relationship practices. The first one is to trust on other people' works - while we can't stay face to face and check our partners progress, we must trust in their reports. The second lesson is to code in a collective way. It is possible using the tools described on step 4. That means joining efforts to get result with quality. OXE Software Factory members try to work in accordance with these best practices.

6. Conclusions

Corporations who allow developers to bring their open source experiences to bear in the corporate environment increase both software quality and employee satisfaction.

In this work, we have described the initial guide to build professional open source software factories. We have listed some steps addressing the integration of the two worlds: software factories and open source software development model. However, after following the suggestions, one of the most important things to figure out is the necessity of a good remote structured environment where team members may communicate and come up with solutions.

Team leadership is usually welcome but the most important thing is to have all the team members actually engaged in a common effort. We can assess the quality of our guide only by using it in real experiences – that's what Open Experience Environment (OXE) Factory has been doing [24].

7. Acknowledgements

One important factor of a open source software factory success is the team members themselves. We are in luck to have a team that really works together. Our special thanks to Carlos Eduardo, Clélio Feitosa, Euclides Neto, Lucas Schimitz, and Severino Andrade, OXE members.

8. References

[1] 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.

[2] Barnett L. (2004) "Forrester Report: Applying Open Source Processes in Corporate Development Organizations." Available at http://vasoftware.com/

[3] Crowston K., Scozzi B. (2002) "Open Source Software Projects as Virtual Organizations competence rallying for Software Development". In IEEE Proc-softw Vol 149, No1, February.

[4] Fuggetta, A. (2003) "Open source software—an evaluation". Journal of Systems and Software, Volume 66, Issue 1, Pages 77-90.

[5] Jacobson I., Booch G., Rumbaugh J. (1999) "The Unified Software Development Process ". Addison-Wesley Professional; 1st edition.

[6] Harvey S., Herbsleb J., Mockus A., Krisghnam M., Tucker G. (1999) "Making the Software Factory Work: Lessons from a Decade of Experience". Available at: www.research.avayalabs.com/ user/audris/papers/factory.pdf, last acess: 29/07/2005.

[7] Mockus, A., Fielding, R. T., Herbsleb, J. D. (2002) "Two case studies of open source software development: Apache and Mozilla". ACM Transactions on Software Engineering and Methodology. Volume 11, Issue3, Pages 309-346.

[8] 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.

[9] Raymond, E. S. (1999) "The Cathedral and the Bazaar". 1st. O'Reilly & Associates, Inc.

[10] Stallman, R. (1999) “The GNU Operating System and the Free Software Movement” in Open Sources: Voices from the Open Source Revolution, C. Dibona, S. Ockman, and M. Stone, Eds., O’Reilly, Sebastopol, Calif., 53–70.

[11] VA Software (2005) Survey Report: Application Development and Open Source Process Trends, available at http://vasoftware.com/

[12] VA Software (2005) Leveraging Open Source Processes and Techniques in the Enterprise”, available at http://vasoftware.com/

[13] Sommerville, I. 1996. Software process models. ACM Computing Surveys 28(1), 269-271.

[14] Naur and B. Randall (Eds), Software Engineering: Report on a conference by the NATO Science Committee. Petrocelli/Charter, NY: NATO Scientific Affairs Division.

[15] Fayad, M. E. 1997. Software development process: a necessary evil. Communications of ACM, 40(9), 101-103

[16] Goth, G. 2005. Open Source Meets Venture Capital. IEEE Distributed Systems Online, 6(6), 2.

[17] Zhao, L. and Elbaum, S. (2003) Quality assurance under the Open Source development model. Journal of Systems and Software. 66 (1), 65-75.

[18] Norris, J. S., Kamp, P. (2004) Mission-Critical Development with Open Source Software: Lessons Learned. IEEE Software, 21 (1), 42-49.

[19] Mockus, A. and Herbsleb, J., Challenges of global software development. In Proceedings of IEEE METRICS (pp. 182-184).

[30] Perens, B. (2005) The Emerging Economic Paradigm of Open Source. http://perens.com/Articles/Economic.html

[21] Beck, K. 2000 Extreme Programming Explained: Embrace Change. Addison-Wesley Longman Publishing Co., Inc.

[22] Jones Albuquerque and Silvio Meira. Software Engineering in Practice: Building Software Factories. ESELAW04 - 1st Experimental Software Engineering Latin American Workshop. SBC - Brazilian Computer Society and IEEE/TCSE-Technical Council on Software Engineering. October, 18. Brasília - DF, 2004.

[23] Wiegers, K. E. 1996 Creating a Software Engineering Culture. Dorset House Publishing Co., Inc.

[24] Cavalcanti, A. P. C., Lucena, L. R., Lucena, M. J. N. R., Moraes, A. K. O. de, Fernandes, D. Y. S., Pereira, S. C., Albuquerque, J. O. and Meira, S. R. L. 2005. Towards an Open Source Software Factory. In: 2nd Experimental Software Engineering Latin American Workshop, Uberlândia, MG, 2005.

[25] Weiss, D. M. and Lai, C. T. 1999 Software Product-Line Engineering: a Family-Based Software Development Process. Addison-Wesley Longman Publishing Co., Inc.

[26] Pilato, M. 2004 Version Control with Subversion. O'Reilly & Associates, Inc.

[27] Johnson, K. A. (2001) Descriptive Process Model for Open-Source Software Development. Master Thesis, Univ. Calgary, Alberta.

[28] Cusumano, M. A. (1989) The Software Factory: A Historical Interpretation. IEEE Software. 6 (2), 23-30.

[29] Bemer, R. W. (1969) Position Papers for Panel Discussion: The Economics of Program Production. In Information Processing 68 (pp. 1626-1627). North-Holand, Amsterdam.

[30] Goldman, R. and Gabriel, R. 2004 Innovation Happens Elsewhere: How and Why a Company Should Participate in Open Source. Morgan Kaufmann Publishers Inc.