OSS Factory: Development Model based at OSS Practices

Ana Isabella Muniz, José Augusto de Oliveira Neto

Universidade Federal de Campina Grande – UFCG

Av. Aprigio Veloso, 882- Cx Postal: 10.106, Paraíba, Brasil

{isabella,zedeguga}@dsc.ufcg.edu.br

Abstract. *In this paper we present OSS Factory (Open Source Software Factory), an ecosystem aligning software demands, undergraduate Computing students qualification and Open Software practices in a collaborating relation, dedicated to produce open software applications to cope with market demands, using students coding potential. A contest among students attending software engineering courses (or volunteers), guided by professors and coordinated by a central entity, is the force to move OSS Factory. To validate elements and interaction proposed, experiments applying the structure described in the paper have been performed and positive results were achieved.*

1. Introduction

Since its first moves, Open Software (OS) phenomena has gained credibility as a concrete alternative to adress world wide TI demands, no matter if in simple home based activities or, mainly, corporations´ needs. It is rare TI department which does not use or has considered adopting an OS based technology.

A research recently developed in Brazil among small municipalities [SOFTEX,2004] has pointed OSS software as a promising alternative to their needs of business processes automation. Technological limitations and budget shortage were main factors to suggest OS software as an alternative direction. In addition, software houses have not showed interest in exploiting this small municipalities market.

A third issue concerning OS software it is how its popularity has not reach academy bounds. There is no remarkable reordering in Software Engineering curriculum towards accommodate OS community practices, or any orchestrated movement to drive students closer to OS world.

The objective of this paper is to present OSS Factory (figure 1), an iteration model conceived from practical experiments, where the three factors mentioned before were adrressed.

Figure 1 - OSS Factory architecture.

OSS Factory engine, detailed in section 2, will integrate the three vertices of figure 1, associating ones needs to others abilities, in continuous interactions that will bring positive outcomes to all involved.

Open Software world can expand its limits of influence and actuation, so far mostly restricted back office technologies, such operating systems, or web, e-mail and application servers. In this new scenario, OS can expand today communities to include less specialized personnel, different from high or medium expertise profile typically found in present communities [Lima, 2005]. In present days, taking part as effective member, with frequent contributions accepted, of an OSS community is a privilege of high level programmers, because the complexity of low level software they hostage.

Observing OS Software popularization, we can notice that main steps in their success walk have been made by their militant's enthusiasm. Today, OS software reach academy simply through

communities and the products they offer freely to students access [MySQL, 2006][Hibernate, 2006][Eclipse, 2006]. Except by few and eventual conferences, the very first contact between students and OS world comes from student individual attitude, downloading most popular OS tools.

Fallowing a predefined flow of activities, combined to OS practices of software engineering, students attending undergraduate Computing courses are able to establish a contact with Free Software culture under a new and different perspective. Instead of mere OS technologies users, under-qualification professional take part of OS world as member of a development community, contributing to applications engineering. In this process students are lead by Software Engineering best practices, in a way that they can improve their analysis, design and coding, and extend those abilities with OS related aspects, giving them a more extensive and distinct qualification.

Software manufactured by OSS Factory will be able to support deficient areas, with low/none technological capacities do develop their own solution or with funding limitations. In the experiments we relate further, software applications were dedicated to automate Brazilian small municipalities business process. However, as described in section 2, OSS Factory can be easily reconfigured do address other business process demands.

The rest of paper is organized according to the following structure: section 2 presents OSS overview, along with detailed description of each of its elements. Section 3 relates contest dynamic. Section 4 shows a real experiment applying OSS Factory production dynamics, followed by outcomes obtained from the experiment Section 5 points some future work. Final concerns are related on section 6, titled Conclusions, brings impressions and overall results obtained from OSS Factory experiments.

2. OSS Factory Overview

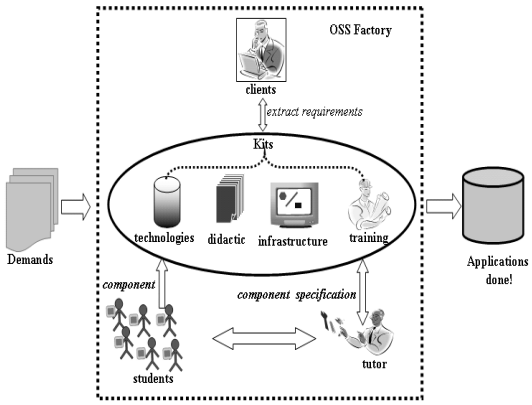

Figure 2 - OSS Factory overview

2.1 Demands

Demands are the source of OSS Factory motivation. As already menssioned, the tipical demands filled out by OSS Factory operation come from segments with TI needs and no technical skills to produce, or fund capacity to acquire them.

In case study presented in Section 4, small municipalities in Brazil are responsible for OSS Factory demands. According to diagnosis resulted from SOFTEX researches [Softex, 2004], 96% of these municipalities are habited in maximum by 100 thousand people [IBGE, 2001]. They are similar in administrative structure, as well, they have few or none capacity to make IT infra-structure investments. From these two characteristics (low budget and similar business process), comes the insight that OS applications, modelled and built according to one of these municipalities business processes, could be applied to the most part or all of them, because of their process compatibility and no costs of license, natural feature of OSS.

In the experiments where OSS Factory was put in operation, demands were presented through direct contact with small municipalities. One of their main demands comes from business process strongly tied to legal constraints, which the administrator is obligated to be compliant with. Frequent changes in legislation result in considerable efforts to keep compliance. Other common class of demands come from social programs, supported by Federal Government. These programs impose some pattern procedures to the municipalities, impacting directly in their business processes and, in most cases, requiring a minimum level of automation. In both case, each demand of automation or support originate in municipalities fulfils Demands box, to be dealt by OSS Factory motion.

2.2 Contest(Competition)

Once demands are identified, a contest involving professors and students from undergraduate Computing courses is carried out. Contest is managed by an entity that will be mainly responsible supervision, with predefined activities towards software application development that will address demands found in first phase of OSS Factory. From now on, that entity will be referred as Coordinator Entity (CE). One single contest can address either one demand or a group of them, depending on professors/CE convenience and applications complexity.

In each edition, application functional requirements elicited from demands are divided through contest time window. How long is this period also must be set up among professors and CE? Following an iterative process, students submit the code they produce for evaluation, made by professors and CE consultants. Grades are assigned to the code, ranking participants in a partial result for that phase. After that, a new phase is started, considering requirements not addressed yet. When all phases are accomplished, partial results are joined and those students who have summed higher scores are the contest winners.

2.3 Tutor

Typical undergraduate Computing curriculum include courses dedicated to Software Engineering practices. In this context, "practices" mean laboratory activities, going beyond theorical teaching which demand analysis, design, codification, test and deployment activities from students. The whole cycle is supported by classical processes and oriented by professor in charge of the course.

Every semester, professors are obligated to repeat efforts to find adequate applications to be delivered as practical project to his courses. "Adequate" means in a complexity and time consumption compliant to students background and scope bounds of his course. After not much long, this procedure extinguishes professor possibilities, and the same applications are repeated time after time, or professor is forced to specify application associated to imaginary projects, with no real clients and requirements.

When inserted in the contest, professor will be provided periodically with projects specifications, arisen from real demands, identified from real clients, ready to assign among students during his course.

2.4 Students

Students attending Software Engineering courses are commonly submitted to unreal requirements, elicited from not real clients, leading them to go through software engineering processes away from client needs. This distance causes low motivation, mainly for being dedicated to build softwares that probably will never be used. When signed in contest, students will be dedicated to real applications and, depending on outcomes, they may be rewarded by their efforts, gain good exposition to software houses market, besides personal satisfaction and pride.

Students that participate to the contest can improve their Software Engineering abilities, going beyond traditional courses, mainly for the following reasons:

Motivation : they will work on real requirements, elucidated from clients that will truly adopt the application they will build, instead of usual mock applications assigned by professors every year; Besides, according to their performance, they may be monetarily rewarded by his efforts, gain good exposition to software houses market, personal satisfaction and pride

Good practices: having contact with tutorials, being pressured to obey recommended design, architecture, documentation and interface patterns delivered by CE, because they are considered in evaluations, students will leave the process with Software Engineering best practices naturally incorporated.

Focussing the big picture, in the former scenario we can find productive capacity of hundreds or thousands of students being allocated in to-be-never-used applications. Signing in this great amount of students to contests, CE will have a small software development army, obtaining the same benefits that Open Software communities currently get from bazaar coding style [RAYMOND,2000]. In addiction, client demands are solved with no cost in software production phase.

2.5 “done” Applications

Every time OSS Factory repeats the production process (a contest edition) the set of application ready to use is increased, offered with no licensing restriction, except those typical for OS software. The mechanism used to deploy applications is a cooperative web portal [FLOAPP, 2006], [MUNIZ, 2007], where the community composed by CE, Software Engineering professors, students signed in to OSS Factory development activities and clients who have applications installed (or interested in adopting any of them) may freely interact, following the same dynamics found in OS software communities.

Clients can access applications through different manners. The most simple of them is just downloading directly from web portal. More complex applications may require presencial support for installation and training. Such support may be accomplished by specialized small companies, inserted in a regional economic development context [Moura et. all, 2006].

There are few details hid in OSS Factory structure described above that must be addressed to garantee smooth interaction among factory’s parts. We mention some bellow.

a) How to present demands in a clear language, enough to guarantee that the application will correctly address requirements?
b) How professor will hear about the contest? What support material will he count with to deal with his students, besides demands description?
c) Which criteria will be used to classify and rank contest participants? Different paradigms and technologies can be evaluated under horizontal, no-distinction criteria?
d) Who coordinates the contest? How coordination is carried out?

Next section details OSS Factory components, answering those questions.

2.6 Technology kits

An Architectural model, followed by technological patterns, was proposed to be adopted by all participants of OSS Factory. This model helps to maintain compatibility among applications conceived during successinding contest editions. Hibernate, Struts, ADO design pattern and Java© web programing support are some of technologies defined as patterns by OSS Factory.

Along with compatibility issues, architecture is expected to be easy to learn and work with. Students with basic development abilities must be able to adopt it. In the other hand, the same architecture must be up to current development patterns, to avoid resistance from more advanced developers. As a final requirement, the architecture must allow students to produce either plain application (1-tier, browser-servlet, for example) or more scalable designs (2, 3, n-tiers). Architecture suggested by OOS Factory meets those requirements, and can be found detailed in [MUNIZ, 2007]

2.7 Training Kit

Technological Patterns established to OSS Factory’s applications intend to contribute to students software engineering qualification. Students can access information kits, built based on software engineering best practices and technologies adequate to market current expectations.

The training process is coordinated through pedagogical support items, available on contest web portal, freely accessed by all participants, students and professors as well. Training kit is composed by the following items:

a) Architectural model description.

This document presents the architecture overview. The document is detailed enough to be comprehensive and followed by beginners, who had never built applications under that type of architecture;

b) Tutorials comprehending each of technologies defined as OSS Factory pattern;

Since architecture definitions include a series of technologies constraints, it is fundamental to provide OSS Factory’s players with pedagogical support to domain those technologies;

c) Complete applications example to download.

Small applications, fully implemented, where all technological patterns defined by OSS Factory pattern can be found and analyzed, with source code and documentation.

2.8 Didactic Kit

This kit is mostly dedicated to professors. Before starting a new course, it ais part of professor's work to plain carefully the activities to be developed. Define book and reference material, to prepare didactic material, presentation slides, exercises, to schadule and topics through course period are example of such activities. Didactic kits provided by OSS Factory hold, in resume, the items listed bellow:

a) A web page template;
b) Class scheduling, proposing a division of discipline content along course period;
c) Class content, offered in presentation slides;
d) Examination proposal and exercise lists, per chapter;
e) Spreadsheet models to grades registration;
f) Bibliography recommended;
g) Free e-books.

The kit motivates professors to join OSS Factory, gaining qualification to him and his students, once Didactic Kit items are based on well succeeded experiences and classical authors;

2.9 Infrastructure Kit

OSS Factory is supported by a web portal used to give publicity to activities and, mainly, to provide and coordinate communication flow among participants. Among most important features, students, volunteers and professors subscriptions, Kit's hostage and distribution, information concerning appraisals, schedules and description of contest dynamics. This web portal [MUNIZ, 2007] is also OSS application repository, where students will upload source code and documentation they produce in every phase of contest.

3. A Free Software Contest Model

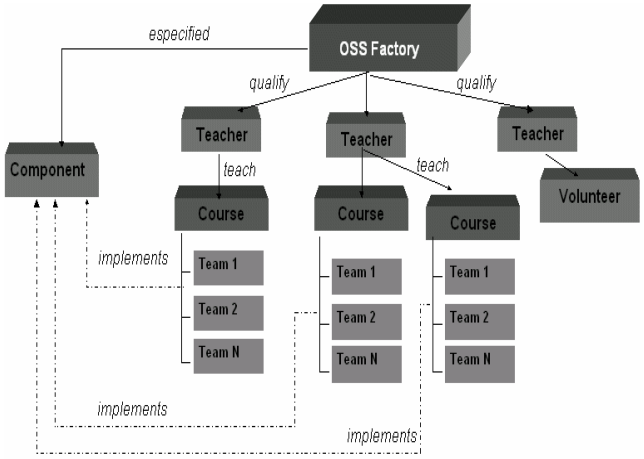

Figure 3 - OSS Contest model.

The contest is managed by Entity Coordinator (CE), in our case study represented by the nucleus of Via Digital project [Via Digital, 2006], located at the Federal University of Campina Grande (UFCG - www.dsc.ufcg.edu.br), whose role can be undertaken by an institution or an involved OS community. This entity has as fundamental attributions: a) select and authorize professors to perform during the contest; b) define the standard architecture of the applications; c) define software requirements; d) construct and maintain the contest portal and d) awards.

3.1 Contest dynamics

The contest begins when CE publishes the list of (or a single) applications to be developed on its web portal.

After this event, comes enrolment of the competing teams and volunteers. The difference between the two types of participants is that first ones are attending course from a professor with credentials by CE, while the others have personal interest in the contest and seek insertion in the competition from the professor. It is up to the professor with credentials, according to his/her

possibilities and management, to limit the number of volunteers enrolled under his/her responsibility – the professor with credentials himself/herself does this.

After enrolment, the period of contributions by the enrolled groups and volunteers begins. Depending on the modality of contest, the contributions can be offered or not. Figure 4 shows the dynamics of the contest. Notice that the dashed line elements report to activities concerning only to interactive modality.

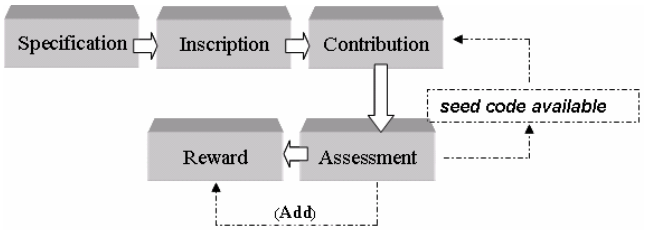

Figure 4 - Contest dynamics.

In the first modality, the phase after the specification and enrolment is a long period of implementation, which will end with the submission of contributions completely constructed by the students. CE will define which applications each credentialed professor will evaluate and these evaluations will compose the rank of the ones who will be awarded prizes. In none of the modalities a professor will be allowed to evaluate participants under his/her own guidance.

In the interactive modality, the implementation will happen in cycles, fulfilling a sequence of requirements in which the CE will fraction the initial specification of the applications. At the end of the deadline for implementing a requirement, the submitted contributions will be judged and graded by the credentialed professors. At the end of the last iteration, the score obtained by the groups in each stage will be computed, defining the winner of the contest.

4. A Case Study

In the section, a real case will be presented, where the proposal described in item 2 goes through some practical experiences.

Conception:
With the beginning of the school year at the Federal University of Campina Grande (UFCG) and the Integrated College of Patos (FIP), the deficiency for new software projects with real applications in the market is great. After the creation of the business model [6] to be implemented by Via Digital [Via Digital, 2006], we found the opportunity to create an inter-city contest among students of these two universities.

Construction:
The first contest was implemented in UFCG's Software Engineering Laboratory (SEL) discipline of the Computer Science course. In this class, there were 12 students enrolled, thus forming six groups of two students each. The professor of this discipline played the role of tutor of all the groups, lecturing about the YP process[easYProcess, 2006].

The groups received professor's tutorship on their questions, and were evaluated by a committee formed by three chairs. These were responsible for choosing the application which would be the target of the contest. For this choice a list of demands was surveyed, involving six city halls in Brazil during an Free Software conference(SOLISC, 2005). This assessed priorities from 1 (low) to 5 (high). The demand reaching the highest priority would be chosen for the contest. At the time, the chosen component was the Pharmacy and Medicine Control (ViaCOM). The CE met all the desired requisites at the city hall (specifically the Health Department). After that, a chronogram with all the activities and deliverables was elaborated to be given to each group.

Evaluation:
In a previous phase of interaction with city halls, two other teams of SEL discipline developed two applications, as project of disciplines, established in the demands of these city halls. That experimental phase tried to create an architectural standard that could allow interoperability among applications (done and to be done). Besides, the experiments were base to documentation standards aiming to

optimise communication and understanding of application. In addition, interface design standard was established. This has been important to users learning proccess [Moura et. all, 2006]:

These standards had collaborated in the definition of kits described in section 2. CE forwards these kits to professors, who divulged specifications to the groups. All of these initial artefacts were displayed on a web page, used to follow-up of the groups. On this page, each group could obtain information about the deliverables dates, case studies, patterns (architectural, documentation, and interface), observations about each group, as well as evaluation criteria adopted by the CE, as describe at table 1 following:

Criteria	
Eliminatory	Classificatory
Compliance to architectural pattern	• Code parameters • Tests • Project patterns • Cohesion e coupling
Functionalities (specific/delivered)	Compliance to documentation pattern

Table 1 - Evaluation criteria.

Evaluation was made in ten days intervals, due to the chronogram of the disciplines. All the groups used CVS [CVS, 2006] to manage system changes and versions. The CE, in turn, downloaded the projects and evaluated them according to the criteria described above. After each deliverable, a winning group was chosen, and the other groups began using that group's code, not necessarily having to erase the previous project, but taking advantage of the best part of each one and thus creating an even better project.

Experience of code exchanging among groups have been not positive and found strong resistance from students, because of considerable overhead demanded to continue other group's source code. Based on these results current evaluation methodology does not recommend any exchange of source code among participants.

At FIP, the Information Systems course, not having the same reality in the formation of students, the professor of the Information Systems Laboratory discipline divided the class into 5 groups, and contrary to UFCG, opted not to follow the patterns conceived by the CE. Instead, he used all this material (Didactic, Technology and Training Kits) and performed training with the students so that they could create a application with quality, even without following any patterns. At every iteration (fifteen days), the professor performed an evaluation of the groups. The one that obtained the best grades was evaluated by the CE examining the UFCG groups.

At the end of all the stages of the process, the group with the best performance had its system (code and documentation) awarded.

To the end of the contest some data had been raised referring the production of the implemented applications, as it shows the table 2:

Indicators:	Numbers:
Tutors	2
Students	30
Applications	7
Use Cases	300
Documentation (JavaDoc)	10.504 (lines)
Unit/Accept Testes	140/350
City halls (Clients)	5

Table 2 - Pós-Contest results.

From table 2 the great contribution can be observed to be given by the academy in the production of information systems with real applications. In a short period of learning stage (4 months) 30 students of software engineering heppen to produce 7 software applications.

5. Future Works

A next stage of experiments involving all the aspects of the OSS Factory has already started. In this round, the same elements will be working, however the community of students and professors included has be increased.

The purpose of the experiment is to evaluate the complexity of managing a larger quantity of people involved, geographically spread out, with more heterogeneous institutions, students and

professors than the first contest edition. Teachers and students from seven different undergraduate computing courses, with some geographical dispersion, are involved in competition activities.

Higher number of participants and geographic dispersion have brought important insights concerning OSS Factory elements, such as evaluation methodology, teacher-coordination and student-coordination communication, requirements granularity and description details and, indeed, budget minimal variation.

At present, we have only a sentiment of compatibility with the model proposed with the modus operandi of the OSS community. With a greater popularization of OSS Factory, this contact could be extended. We intend to more precisely detect the community's acceptability of the model OSS Factory, with the development induced by requisites and awards, consequently less free than dynamics of classic communities.

6. Conclusions

The proposal of the OSS Factory is to have coordination among its three vertexes, so as to extract potentialities from each one of them, attending to the demands of the others.

Recent experiments performed show that its dynamics contributes on teaching of practical software engineering courses, with real and motivating activities using specialized didactic material, which makes possible improvements in students qualification process. These results have been obtained in real cases of OSS Factory operation. At the same time, the performance of OSS Factory showed to be capable of responding to demands and solving real problems at small city halls, through typical practices of OSS communities.

7. References

LIMA, C. A. Práticas para Gerência e Desenvolvimento de Projetos de Software Livre. Federal University de Campina Grande, 2005.

MySQL – Available at: http://www.mysql.org – Last access: February 20, 2007.

Hibernate - Available at: http://www.hibrenate.org – Last access: February 20, 2007.

Eclipse - Available at: http://www.eclipse.org – Last access: February 20, 2007.

SOFTEX, Report - Projeto Aplicação de Software Livre em Prefeituras. SOFTEX Association, 2004.

IBGE, Censo 2001, Instituto Brasileiro de Geografia e Estatística (IBGE - The Brazilian Institute of Geography and Statistics). Available at www.ibge.gov.br , 2001.

Raymond, E. (2000, August). *The Cathedral and the Bazaar,* August, 2002, from http://www.tuxedo.org/"esp/writings/cathedral-bazaar/cathedral-bazaar

MOURA, A. et. All. Open Source Software in Small City Governments and the Promotion of Regional Entrepreneurship. eChallenges Proceedings 2006. Barcelona, october 2006.

FLOAPP - Available at: http://flopref.paqtc.org.br:8080/flop/–Last access: February 20, 2007.

MUNIZ, I. and GARCIA, F. - FLOApp: A Cooperative Environment for Development of Free Governmental Applications. ECEG (The European Conference on e-Government) Proceedings 2007. Den Haag, Netherlands.

Via Digital – O caminho inteligente para a informatização pública.Available at: http://www.viadigital.ufsc.br – Last access: February 20, 2007.

easYProcess Development Process. Available at: http://www.dsc.ufcg.edu.br/~yp – Last access: February 20, 2007.

SOLISC - Available at: http://www.solisc.org.br – Last access: February 20, 2007.

CVS - Available at: http://www.cvshome.org – Last access: February 20, 2007.