Uma experiência na implantação de processo em uma fábrica de software livre

Regiane Brito, Patrícia Ferreira, Kleber Silva, Vanilson Burégio, Ivan Leite

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851, 50732-970, Recife – PE – Brazil
{rab3,pmf,khfts,vaab,iobl}@cin.ufpe.br

Abstract. *Nowadays, the OSS (Open Source Software) development model had gained the attention of TI market, because it appears as a possible estrategic choice that promisses to transform the global software market. However, the firms have been encontering difficulties in the adoption of this model. In these times, didn't exist many experiences that evaluate the efficience of this open source approach as bussiness model. The purpose of this paper is to present a academic experience based on the concept of a distributed software factory [Aaen, 1997] using a development process adapted to the OSS model.*

Resumo. *Ultimamente, o modelo de desenvolvimento OSS (Open Source Software) tem atraído a atenção do mercado de TI, pois surge como uma possível alternativa estratégica que promete revolucionar a indústria de software mundial. Entretanto, as empresas vêm encontrando dificuldades na adoção deste modelo. Até hoje, ainda são poucos os experimentos que avaliam com precisão a eficácia da abordagem open source como modelo de negócio. O objetivo deste artigo é apresentar um experimento acadêmico baseado no conceito de fábrica de software [Aaen, 1997] distribuída utilizando um processo de desenvolvimento adaptado para o modelo OSS.*

1 Introdução

Ultimamente, o modelo de desenvolvimento OSS (*Open Source Software*) tem atraído a atenção do mercado de TI, tanto no âmbito das empresas públicas quanto das privadas, pois surge como uma possível alternativa estratégica que promete revolucionar a indústria de software mundial. Alguns acreditam que o desenvolvimento *open source* será capaz de transformar a indústria de "manufatura" de software para uma indústria de software baseada em serviços [Sharma, 2002]. Metas como rapidez no desenvolvimento com baixo custo e alta qualidade, que podem ser alcançadas através do modelo OSS, têm servido de motivação para que algumas empresas de software busquem adotá-lo.

Entretanto, as empresas vêm encontrando dificuldades na adoção deste modelo, principalmente, por se tratar de um modelo que acarreta uma mudança de paradigma tanto do ponto de vista estrutural e cultural, quanto de processos, o que pode levar a uma modificação profunda no seu modelo de negócios. Até hoje, ainda são poucos os experimentos que avaliam com precisão a eficácia da abordagem *open source* como modelo de negócio [Sharma, 2002]. As dificuldades para realização desses experimentos vão desde a criação de um adequado processo de desenvolvimento até a criação e manutenção de uma comunidade *open source* com suas contribuições.

O objetivo deste artigo é apresentar um experimento acadêmico baseado no conceito de fábrica de software [Aaen, 1997] distribuída utilizando um processo de desenvolvimento adaptado para o modelo OSS. Este experimento simulou um cenário com uma característica peculiar, que foi a presença de um cliente com um projeto real a ser executado, obedecendo a cronogramas com prazos bem definidos. Esse tipo de cenário é diferente daqueles comuns do modelo OSS, onde as contribuições são feitas de acordo com a escolha do colaborador. Aqui precisamos utilizar mecanismos para delegação e controle do desenvolvimento, além de um forte esquema de acompanhamento gerencial para atendimento dos prazos. O projeto em questão será detalhado posteriormente.

Na segunda seção é apresentado o contexto relacionado a fábricas de software livre. A terceira seção descreve como a fábrica foi definida. Na quarta seção, são detalhadas características da aplicação desenvolvida. Na quinta seção é apresentada a execução do processo. Na sexta seção são descritos em quais aspectos o processo evoluiu. Na sétima seção, são apresentadas as considerações finais e na sétima seção, as referências.

2 Fábricas de Software Livre

Software livre é um termo usado para indicar que o software é desenvolvido e distribuído sob algum tipo de licença que garanta ao seu usuário liberdades relacionadas ao uso, alteração e redistribuição. Seu aspecto fundamental é o fato do código fonte estar livremente disponível para ser lido, estudado ou modificado por qualquer pessoa interessada [Reis, 2003].

Um projeto de software livre é uma organização composta por um **conjunto de pessoas** que usa e desenvolve **um único software livre**, contribuindo para uma base comum de código-fonte e conhecimento. Este grupo terá à sua disposição **ferramentas de comunicação e trabalho colaborativo**, e um conjunto de experiências e opiniões técnicas que usam para discutir o andamento do projeto [Reis, 2003].

Uma fábrica de software é uma organização que provê serviços de desenvolvimento de sistemas com qualidade, a baixo custo e de forma rápida, utilizando um processo de desenvolvimento de software bem definido e com apoio de tecnologias de mercado, além de reconhecer e lidar com oportunidades de melhoria do processo [Herbsleb, 1999].

Nesse contexto, uma fábrica de software livre é aquela que desenvolve projetos de software livre, em sua totalidade ou em parte, e possui um modelo de negócios diferente do modelo tradicional, onde o código fonte do produto não é disponibilizado. Hecker (2000) apresenta diferentes modelos de negócio para que empresas interessadas em software livre possam obter lucro.

3 Definição da fábrica

Eric Raymond (1999) em “The Cathedral and the Bazaar”, comparou o modelo de programação comercial denominado Catedral e o modelo de desenvolvimento com código aberto denominado Bazaar. Neste último, qualquer um com acesso a Internet e habilidades de programação pode integrar o processo de desenvolvimento do software.

A proposta do experimento era a construção de uma fábrica de software para o desenvolvimento de projetos de software livre. Para isso, foi necessária a definição do processo a ser utilizado. Por opção da fábrica, foi construída uma "Catedral" e a integração com a comunidade foi definida através de uma interface externa.

O processo foi baseado no RUP (*Rational Unified Process*) e possuía as gerências de qualidade, configuração e projetos sendo desempenhadas exclusivamente por membros da fábrica.

3.1 Participantes

A fábrica contava com 11 pessoas, entre estudantes e profissionais de Tecnologia da Informação. Com exceção dos papéis de gerência, a mesma pessoa assumiu por vezes um outro papel. Essas pessoas foram divididas de acordo com suas experiências, como mostrado na tabela a seguir.

Tabela 1. Composição da fábrica de software

Papel	**Responsabilidades**	**Quant.**
Analista de negócios	Elicitar os requisitos com o cliente e garantir que o sistema seja implementado de acordo como foi especificado.	1
Arquiteto de software	Definir a arquitetura do sistema de maneira adequada e acompanhar os programadores nas suas atividades.	1
Programador	Implementar o que foi especificado pelo analista de negócios, de acordo com a arquitetura definida.	6
Gerente de Projeto	Gerenciar e acompanhar todas as fases do projeto e negociar prazos e entregas com o cliente.	1
Gerente de Configuração	Configurar o ambiente de desenvolvimento do projeto, através da nomenclatura e localização dos artefatos gerados.	1
Gerente de Qualidade	Assegurar que o processo está sendo seguido e criar meios para garantir a qualidade dos artefatos gerados.	1

3.2 Modelo do processo

As atividades de suporte (configuração, qualidade e projeto) atuam para garantir que o projeto seja realizado como foi esperado, dentro do orçamento e prazo definidos. Além disso, a qualidade do produto deve tentar ser alcançada e o desenvolvimento do projeto deve ser feito de maneira ordenada.

Para que todos esses requisitos sejam atendidos, as gerências de configuração, qualidade e projeto possuíam um conjunto de artefatos, onde os principais são o plano de projeto, configuração e qualidade que devem ser preparados tão logo o projeto seja iniciado e devem ser seguidos por todas as pessoas envolvidas.

O processo de desenvolvimento contém as atividades que compõem o ciclo de vida do desenvolvimento de um software [Rocha, 2001]. O modelo do processo de desenvolvimento definido está representado na figura 1. Esse modelo foi construído de acordo com a notação SPEM (*Software Process Engineering Metamodel*).

Os papéis envolvidos são: analista de negócios, arquiteto de software, programador, integrante da comunidade e integrador. O integrante da comunidade é uma

pessoa que está participando no desenvolvimento do projeto através da interface com a comunidade de software livre e o integrador é a pessoa responsável por receber as contribuições (internas da fábrica ou da comunidade) e integrar ao produto, podendo ser um integrante da fábrica ou um integrante da comunidade que tenha recebido mérito no desenvolvimento (através, por exemplo, de sucessivas contribuições). Esse estilo de premiação visa incentivar a participação da comunidade e é conhecido como meritocracia [Raymond, 1999].

Os artefatos obrigatórios previstos são documento de requisitos, documento de arquitetura, resultado de testes e o produto em si.

Em relação à contribuição da comunidade externa, a mesma deve ser feita da seguinte forma. Um integrante se cadastra como participante do projeto, faz o *download* do código fonte e documentos disponibilizados. Então, pode submeter alguma contribuição (informe de *bugs* ou implementação de nova funcionalidade). Essa submissão vai ser verificada pelo integrador para ver se está de acordo com o que foi especificado no processo. Se a contribuição for relevante e estiver no formado adequado, a mesma será incorporada ao sistema.

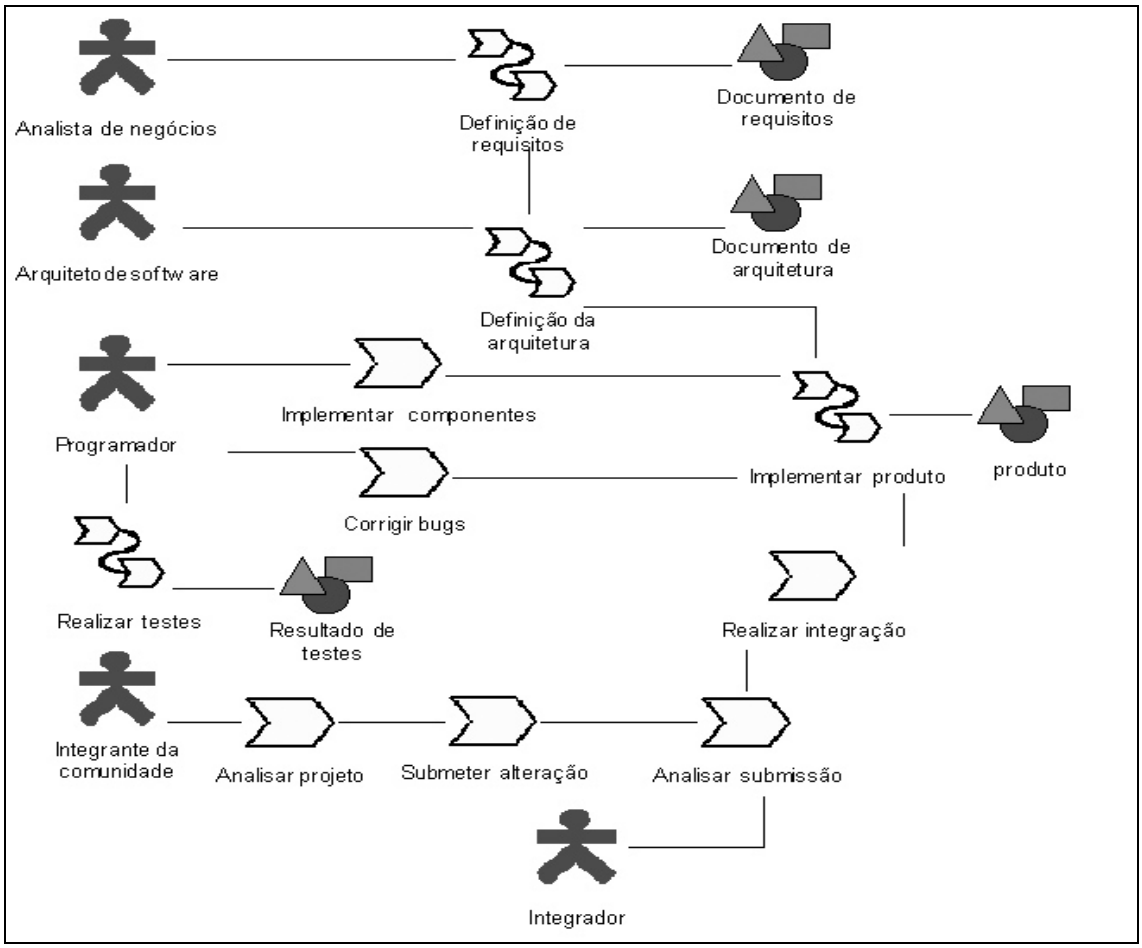


Figura 1 Processo de desenvolvimento

4 Projeto Canto Livre: características e necessidades

O projeto Canto Livre surgiu com o objetivo de criar um espaço de convergência do conteúdo artístico do Brasil. Do ponto de vista de pesquisadores e amantes das artes, pode ser visto como um local onde é possível encontrar, de forma simples e rápida, grande parte do que foi e é produzido no cenário artístico brasileiro. Do ponto de vista do artista, é um meio simples, seguro e igualitário de criar, produzir, divulgar e distribuir sua arte, sem que seja restrito aos interesses comerciais dos detentores dos já escassos meios de produção, divulgação e distribuição artísticas, incapazes de satisfazer às necessidades dos artistas e dos consumidores das artes.

Para isso o sistema deve permitir que os usuários possam cadastrar sua produção artística e intelectual, assim como localizar e manipular a produção de outros usuários. Além disso, o sistema deve prover mecanismos para controles autorais, definidos pelos próprios usuários, além de mecanismos para integração dos mesmos.

Em relação à arquitetura, o projeto Canto Livre é composto por dois *softwares*, que, integrados, formam uma rede ponto-a-ponto centralizada (possui servidor). Um dos *softwares*, módulo cliente, será usado diretamente pelos usuários, como aplicações locais, de onde eles podem inserir e recuperar informações, ou seja, a produção artística e intelectual dos próprios usuários. As várias aplicações do módulo cliente podem se comunicar e trocar informações, daí a característica ponto-a-ponto. O segundo *software*, módulo servidor, deve controlar o acesso de usuários à rede Canto Livre e centralizar o sistema de busca

O projeto em si abrange muitas outras funcionalidades, além de uma ampla discussão sobre o seu contexto de aplicação. O escopo do projeto na fábrica se restringiu às funcionalidades de cadastro e consulta de dados, serviço de mensagem instantânea e controle autoral de obras, através do uso das licenças *Creative Commons* [Creative, 2004]. Também foi prevista a definição da arquitetura do sistema.

A figura 2 descreve a estrutura geral do sistema. O servidor, como centralizador de consultas e os clientes como verdadeiros responsáveis pelo controle de arquivos e suas licenças de uso.

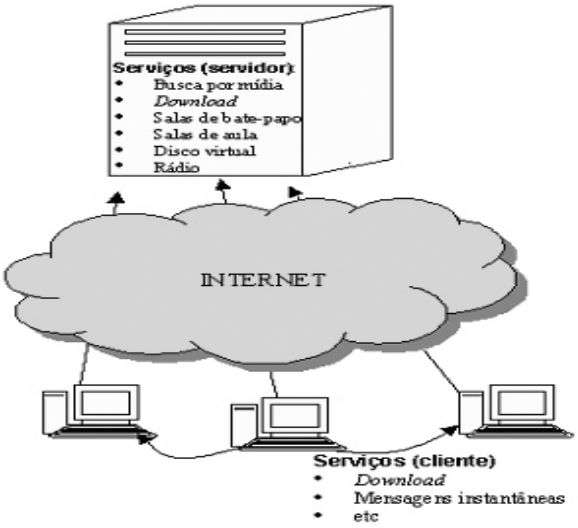


Figura 2 Visão Geral da Rede Canto Livre

O código-fonte do software cliente será distribuído pela licença CC-GNU GPL (tradução da distribuição do *General Public License*, da *Free Software Foundation* para *Creative Commons*). Como a licença prevê, esse código poderá ser utilizado em quaisquer outros sistemas não comerciais.

O código-fonte do servidor também será distribuído pela licença CC-GNU GLP. Porém, as melhorias serão feitas de forma mais restrita. As modificações visando melhoria só poderão ser feitas por usuários com mérito no desenvolvimento do software cliente.

5 Desenvolvimento do projeto

Um processo de software particular, da forma como implementado por uma organização, é algo concreto e individualizado: cada equipe de desenvolvimento **instancia** sua versão. Uma descrição de um processo de software específico oferece uma forma de entender e avaliar seus objetivos, forças e fraquezas [Reis, 2003].

O desenvolvimento foi dividido em duas fases, a primeira com um projeto pilo e a segunda para o projeto principal. Essa seção descreve como ocorreu o desenvolvimento e a instanciação do processo em relação às duas fases do projeto, fornecendo também um histórico desse desenvolvimento em relação à colaboração entre os membros da equipe, e destes com o cliente.

Para que a fábrica pudesse testar e validar seu processo, primeiramente foi proposto o escopo da fase piloto, cuja premissa era ter um conjunto reduzido de funcionalidades. Na RFP (*Request for Proposal*) recebida, o cliente solicitava algumas funcionalidades em um sistema existente (descrito na seção 4). A fábrica deveria inserir cadastro de arquivo e usuário e discutir e implementar modelos de segurança para evitar que ocorresse a pirataria em transferências de arquivos utilizando o sistema.

A fábrica teve cinco semanas entre apresentação de propostas técnica e comercial e entrega do sistema para o cliente.

Dificuldades foram encontradas durante o desenvolvimento do projeto piloto. A seguir uma lista de algumas delas:

- Falta de um gerenciamento eficaz;
- Falha de comunicação entre os membros da equipe;
- Falta de conhecimento do processo por todos os integrantes da fábrica;
- Falta de documentação do projeto desenvolvido *a priori* pelo cliente;
- Complexidade do projeto piloto, por se tratar de uma aplicação *peer to peer* e a fábrica não possuir *know how* na tecnologia utilizada.

Após um período dedicado a ajustes no processo, a fábrica recebeu uma nova RFP referente à segunda fase do projeto, onde o cliente solicitava melhorias no código atual (*refactory*, modularidade e robustez), suporte às licenças do *Creative Commons* e inserção de um serviço para troca de mensagens instantâneas.

Problemas continuaram a ocorrer, em virtude da pouca disponibilidade de alguns membros da equipe que estavam envolvidos com outras atividades. No entanto, o desenvolvimento deste segundo projeto foi mais bem controlado, pois a fábrica passou a

ter um gerente de projetos dedicado e a equipe já possuía conhecimento na tecnologia utilizada. Este desenvolvimento durou oito semanas.

A seção 5.1 apresenta questões referentes à comunicação e a seção 5.2 discute as ferramentas adotadas pela fábrica.

5.1 Comunicação

Em desenvolvimento distribuído, onde os participantes não estão fisicamente próximos, surgem alguns problemas de comunicação. Herbsleb (1999) descreve alguns problemas que prejudicam a comunicação efetiva em equipes distribuídas:

- Perda de contato não planejado: ocorre em situações como encontros no corredor ou durante o horário do almoço, por exemplo. Nesses encontros discute-se muita coisa, inclusive assuntos referentes ao projeto em desenvolvimento. Nessas ocasiões, podem-se esclarecer dúvidas sobre o projeto;
- Dificuldade em saber quem procurar sobre um problema: o responsável por uma parte do projeto (arquitetura, por exemplo) está identificado em algum documento ou site e existe um tempo gasto até sua localização;
- Custo para iniciar o contato: referente ao fato do desenvolvedor procurar saber se o outro está disponível;
- **Perda de confiança para comunicar abertamente:** o que leva na maioria das vezes a reuniões estritamente formais.

Visando diminuir os problemas de comunicação em equipes totalmente distribuídas (e porque nesse caso todos os integrantes estavam em uma mesma localidade), os membros da fábrica se reuniam duas vezes por semana para planejar novos passos e acompanhar o progresso dos projetos.

Além dessas reuniões presenciais, ferramentas para troca de mensagens instantâneas e *email* também eram utilizadas para reuniões virtuais e eventuais discussões ou tomada de decisões.

Também foi utilizada uma ferramenta de colaboração via web, onde era possível atualizar as informações sobre o projeto. Nessa ferramenta, foram construídas duas visões: uma para o processo, e outra para o projeto.

A visão do processo apresentava detalhes de todo o processo que foi definido, possuindo *links* que permitiam o *download* dos *templates* definidos, possibilitando assim sua fácil utilização.

A visão do projeto possuía todo o planejamento do gerente do projeto, gráficos de acompanhamento, cronograma de atividades, *download* dos artefatos produzidos. Durante a segunda fase, o cliente passou a acompanhar a entrega dos artefatos a partir desta visão, mais precisamente a partir de *links* disponibilizados no cronograma. Dessa forma, a fábrica conseguiu acabar com o problema da falta de visibilidade, que foi uma das reclamações do cliente em relação à primeira fase.

Outra ferramenta utilizada no desenvolvimento foi o *Issue Tracker* disponibilizada no site do *Tigris*, onde estava hospedado o projeto. Através dessa

ferramenta, era possível controlar as atividades em desenvolvimento na fábrica, sejam elas atividades de correção de *bugs* ou implementação de novas funcionalidades e até atividades que não envolvessem codificação, por exemplo, gerar o documento de requisitos.

A importância da utilização dessa ferramenta, aliada as atividades de gerência só foi percebida após o desenvolvimento do projeto piloto, onde a fábrica teve problemas em acompanhar a evolução do desenvolvimento do sistema. Depois que a fábrica passou a utilizá-la, cada desenvolvedor sabia exatamente as atividades que estavam associadas a ele e podia reportar quando a mesma fosse concluída.

5.2 Ferramentas de suporte

Essa seção descreve algumas ferramentas essenciais no desenvolvimento da aplicação. O critério para escolha das ferramentas é que fossem *free*. Apesar dessa opção por ferramentas *free* não ter sido imposta, a fábrica se propôs a testar a viabilidade dessas ferramentas no desenvolvimento de um projeto real.

- *Atlassian Confluence*: utilizado como ferramenta de colaboração, onde os membros da fábrica atualizavam o status do projeto e faziam ajustes no processo se necessário. Essa ferramenta é gratuita para projetos *open source.*
- *Issue Tracker*: utilizado para controle de itens a fazer (implementação, correção de *bugs*). Essa ferramenta está disponível no Tigris (2004), que é uma comunidade *open source* com intuito de construir melhores ferramentas para o desenvolvimento de software colaborativo. Cada projeto está relacionado com uma das áreas de interesse da comunidade.
- ArgoUML: Ferramenta para modelagem UML.
- Eclipse: Ambiente de desenvolvimento de aplicações Java.
- WinCVS: Cliente da ferramenta de controle de versão.

5.3 Integração com a comunidade livre

Apesar da proposta de construir uma Fábrica de Software Livre, não houve tempo para receber contribuições de desenvolvedores da comunidade. No entanto, todo o trabalho produzido está disponível através do site do *Tigris*, onde essa contribuição pode eventualmente ocorrer.

Krishnamurthy (2002) e Hecker (2004) apresentam discussões sobre o problema de como fazer com que desenvolvedores participem de um projeto de software livre. Ele apresenta cinco fatores que motivam os participantes de projetos *open source*:

- Participar de um projeto intelectualmente estimulante;
- Melhorar suas habilidades;
- Ter a oportunidade de trabalhar com um projeto *open source;*
- Fornecer suporte ou distribuir o produto;
- Interesse em customizar o software.

6 Evolução do processo

Nos dias de hoje a competitividade entre as empresas têm levado com que essas procurem melhorar a qualidade dos seus produtos de software. O processo de software tem se mostrado o fator determinante para a qualidade do produto final [Rocha, 2001]. Dessa forma, é necessário que o processo seja constantemente avaliado na tentativa de atingir seu modelo ideal.

A seção 6.1 apresenta uma descrição mais detalhadas dos problemas enfrentados em relação a execução do processo para cada área e a seção 6.2 como esses problemas foram resolvidos e contribuíram para a evolução do processo.

6.1 Problemas

Durante o desenvolvimento do projeto piloto, foram percebidas algumas dificuldades na utilização do processo, principalmente porque o mesmo estava muito sobrecarregado de atividades e a fábrica não possuía pessoal disponível para sua execução. Devido a isso, alguns processos não foram completamente executados. A seguir uma descrição dos principais problemas por área:

- **Vendas**: Houve falta de apoio técnico para auxiliar o vendedor na definição do escopo do projeto, causando uma definição incompleta e gerando atrasos na entrega final. Ainda, houve pouco contato com o cliente para esclarecimentos e falta de comunicação com a equipe para auxílio na estimativa de esforço, o que gerou uma estimativa menor que o necessário.
- **Gerência de projetos**: Este processo não foi executado completamente devido a indisponibilidade do gerente de projetos. Além disso, não houve um acompanhamento da equipe, levando a uma falta de controle sobre o status do projeto.
- **Garantia de qualidade**: O processo de garantia da qualidade foi parcialmente seguido. As revisões não foram realizadas formalmente em todos os artefatos. Ainda, a auditoria dos processos realizada envolveu uma pequena parte da fábrica e a auditoria de produto não foi realizada. O relatório de *post-mortem* também não foi realizado, pois a equipe estava concentrada na finalização do produto.
- **Gerência de Configuração**: O software para controle de mudanças, Atlassian Jira [Jira, 2004] apresentou vários problemas e teve que ser substituído pelo *Issue Tracker* da Collabnet (disponível através do site do *Tigris*). Essa mudança não trouxe maiores problemas porque a equipe ainda não havia começado a utilizar o Jira. O principal problema dessa área foi a falta de um treinamento eficaz para todos os integrantes da fábrica e a falta de comprometimento da equipe em ler e seguir o plano de configuração.
- **Desenvolvimento**: Foi o processo mais prejudicado, devido, principalmente, ao atraso na conclusão dos planos de configuração e projeto, falta de acompanhamento do gerente de projeto e erros na estimativa de esforço. Este projeto foi parcialmente executado e sua execução não foi linear, uma vez que a equipe de desenvolvimento teve de se deter mais na codificação para

conseguir atender aos prazos estabelecido, levando a insatisfação e sobrecarga dos integrantes dessa equipe.

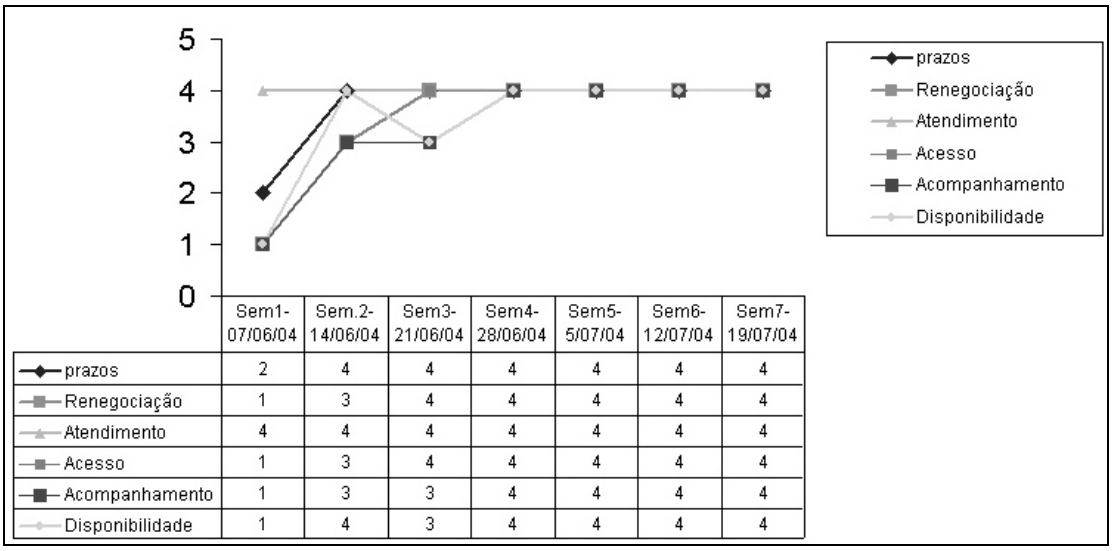
6.2 Calibragem

Devido ao fato que apenas alguns processos foram completamente executados, a avaliação adequada de todos os processos definidos foi prejudicada. No entanto, para todos os processos, a equipe conseguiu identificar oportunidades de melhoria através dos problemas enfrentados.

Na fase de calibragem do processo, foram definidos ajustes onde algumas atividades ficaram como opcionais e outras foram retiradas, adequando também alguns *templates* dos artefatos propostos. A seguir as principais alterações no processo, por área:

- **Vendas**: As atividades de vendas passaram a ser realizadas por um analista de negócios, e o escopo do sistema passou a ser mais bem definido. Além disso, muitos dos artefatos foram descartados, deixando apenas as propostas técnica e comercial.
- **Gerência de projetos**: Houve a necessidade da criação de alguns artefatos, como a matriz de rastreabilidade. Além disso, a utilização de uma ferramenta de colaboração via web [Atlasian, 2004] tornou mais efetivo o acompanhamento da equipe distribuída.
- **Gerência de configuração**: Passou-se a adotar um modelo mais rígido de integração de código fonte que abrangesse contribuições da comunidade de software livre. Esse novo modelo passou a adotar o papel de um **integrador**, semelhante como ocorre nas comunidades de software livre. Essa pessoa é responsável por receber, analisar e incorporar as submissões. Esse processo foi simulado com os integrantes da própria fábrica, durante o desenvolvimento da segunda fase do projeto.
- **Garantia de qualidade**: Foram retirados alguns artefatos que tornavam o processo lento, sem impacto nas atividades definidas. A revisão também se tornou mais ágil através da utilização da ferramenta de *Issue Tracker* para controle de revisões e da inserção de comentários no próprio documento que estava sendo revisado.
- **Desenvolvimento**: O documento de requisitos foi modificado e passou a conter também a definição dos casos de uso. Além disso, houve a verticalização da área através da definição de responsáveis por cada grupo (codificação, análise, arquitetura e teste). Os responsáveis de cada área eram encarregados de distribuir as tarefas.

Além das alterações citadas acima, a fábrica passou a adotar uma pesquisa semanal de satisfação do cliente sob a responsabilidade da equipe de qualidade, para que possíveis problemas fossem detectados e corrigidos antes da conclusão do projeto. A figura 3 apresenta essa evolução no decorrer do desenvolvimento.



	Sem1-07/06/04	Sem.2-14/06/04	Sem3-21/06/04	Sem4-28/06/04	Sem5-5/07/04	Sem6-12/07/04	Sem7-19/07/04
prazos	2	4	4	4	4	4	4
Renegociação	1	3	4	4	4	4	4
Atendimento	4	4	4	4	4	4	4
Acesso	1	3	4	4	4	4	4
Acompanhamento	1	3	3	4	4	4	4
Disponibilidade	1	4	3	4	4	4	4

Figura 3 – Acompanhamento do nível de satisfação do cliente

Após esses ajustes, o desenvolvimento do projeto principal pôde ser mais bem controlado. Além disso, devido à fábrica estar mais experiente em relação ao processo, modificações dinâmicas puderam ser propostas e incorporadas.

7 Considerações Finais

Esse artigo apresentou a experiência de uma equipe formada por estudantes e profissionais de TI na concepção de uma Fábrica de Software Livre, onde houve a definição de um processo de software e desenvolvimento de projetos para um cliente real. Ou seja, simulou-se um ambiente real com pressões (de prazos, cronograma, entregas) e problemas (entre os membros da fábrica e entre esses e o cliente) também reais onde todos tiveram muitas lições aprendidas.

Em relação a um processo de software para projetos *open source* para cliente reais, pôde-se perceber que este deve ser o mais simples possível e adaptável (ajustável) de acordo com as necessidades do projeto em desenvolvimento. O processo definido teve pontos considerados positivos:

- Apesar da resistência inicial na utilização do processo de gerência de configuração, a garantia de que sempre a fábrica tinha uma versão pronta para ser entregue (pois o integrador garantia isso) justifica sua utilização;
- A existência de um gerente dedicado e realizando acompanhamentos é imprescindível em qualquer projeto;
- Pesquisas com clientes servem como indicativos de que o processo precisa evoluir.

Após uma análise dos problemas enfrentados e do processo como um todo, pontos negativos também foram encontrados, em especial:

- Não houve treinamento e divulgação adequada do processo no *startup* da fábrica;
- O processo não cobria planos de contingência;

§ Não havia um padrão ou *template* para atribuição das *issues* e notificação de *bugs*, o que é essencial quando se trata de uma equipe distribuída.

Contudo, a experiência adquirida na concepção dessa fábrica de software foi bastante importante, principalmente por ser uma experiência prática pouco convencional em cursos de pós-graduação. No entanto, ainda há muito que definir em relação a Fábricas de Software Livre, principalmente em relação à integração desta com comunidades livres.

8 Referências

Aaen, I., Bottcher, P. and Mathiassen, L. (1997). "The Software Factory: Contributions and Illusions". In: Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo.

ArgoUML (2004), http://argouml.tigris.org/, Agosto.

Atlassian Confluence (2004), http://www.atlassian.com/software/confluence/, Agosto

Atlassian Jira (2004), http://www.atlassian.com/software/jira, Outubro.

Creative Commons (2004),http://www.creativecommons.org, Agosto.

Eclipse (2004). "Eclipse", http://www.eclipse.org/, Agosto.

Hecker, F. (2004). "Setting Up Shop:The Business of Open-Source Software", http://www.hecker.org/writings/setting-up-shop.html, Novembro.

Herbsleb, J. D., Grinter, R. E. (1999). "Splitting the Organization and Integrating the Code: Conway's Law Revisited". In: *Proceedings of ICSE*. Los Angeles: IEEECSP, 1999. p. 85–95

Krishnamurthy, S (2002). "Cave or Community? An Empirical Examiniation of 100 Mature Open Soruce Projects". *First Monday*, v. 7, n. 6 http://www.firstmonday.dk/issues/issue7 6/krishnamurthy/, Junho.

Raymond, Eric S.(1999) "The cathedral and the Bazaar". Sebastopol, CA, USA: O'Reilly.

Reis, C. (2003). "Caracterização de um processo de software para projetos de software livre" Dissertação de mestrado do ICMC de São Paulo.

Rocha, A. R, Maldonado, J.C, Weber, K. C. (2004) "Qualidade de Software: Teoria e Prática". Prentice Hall

Sharma, S., Sugumaran, V. and Rajagopalan, B. (2002) "A framework for creating hybrid-open source software communities".

Software Process Engeneering Metamodel, Version 1.0 (2004), http://www.omg.org/technology/documents/formal/spem.htm, Outubro.

Tigris.org (2004). "Open Source Software Engineering", http://www.tigris.org, Agosto.

WinCVS (2004). "WinCVS", http://www.wincvs.org/, Agosto.