Definição e Melhoria de Processos em uma Fábrica de Software Livre

Breno Spindola[1], Carlos Albuquerque[1], Jorge Mascena[1], Karine Coelho[1], Ryan Albuquerque[2], Thayssa Rocha[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851 – Cidade Universitária - Recife - PE – Brasil

[2]C.E.S.A.R – Centro de Estudos Avançados do Recife
Rua Bione, nº 220, Cais do Apolo – Recife Antigo - Recife - PE – Brasil

{bcsc, cama, jccpm, kcao, tar}@cin.ufpe.br, ryan@cesar.org.br

Abstract. *In the last years, it has been observed an increasing movement around open source software development, using more distributed environments, where a development team member can be in any place in the world. However, this scenario very was little argued, and the majority of the current software factories do not have shaped processes that can contemplate this particularity. In this article will present the results of an academic experience of an Open Source Software Factory, using a geographically distributed team, based in the use of adapted processes, capable of attending such necessities based on quality requirements and high productivity.*

Resumo. *Nos últimos anos, tem-se observado uma crescente movimentação em torno do desenvolvimento de software livre, utilizando ambientes mais distribuídos, onde um integrante da equipe de desenvolvimento pode estar em qualquer lugar do mundo. Porém, este cenário foi muito pouco discutido, e a maioria das fábricas de software atual não possui processos modelados que possam contemplar esta particularidade. Neste artigo serão apresentados os resultados da experiência acadêmica de uma fábrica de software livre, utilizando uma equipe geograficamente distribuída, baseada na utilização de processos adaptados, capazes de atender tais necessidades, considerando requisitos de qualidade e alta produtividade.*

1. Introdução

Durante muito tempo se imaginou que o desenvolvimento de software livre devia seguir o estilo de desenvolvimento “bazar”, ou seja, desprovido de qualquer processo com um nível de formalização maior em detrimento de um estilo mais comercial e formal denominado de “catedral” [RAYMOND 2004].

Porém, a cada dia é observada uma crescente movimentação em torno do desenvolvimento de software livre, utilizando ambientes cada vez mais distribuídos, onde um integrante da equipe pode estar em qualquer lugar, caracterizando o desenvolvimento em equipe geograficamente distribuída. Além disso, a crescente demanda de mercado por software livre de uso comercial necessita que sejam satisfeitas restrições de custo, prazo e qualidade. [HECKER 1999].

Motivados por este cenário, foi executado um experimento prático de tentar organizar o ambiente de desenvolvimento de software livre, de forma a atender às exigências do mercado, que no caso foi representado por um cliente real. Para conceber a experiência, foi criado um modelo de fábrica de software livre, muito influenciado pelo estilo de desenvolvimento "catedral" e por estudos realizados em [ROCHA 2003], mas inserindo também características inerentes ao desenvolvimento suportado através de comunidades. Este artigo relata a execução desta experiência, mostrando os pontos fortes e fracos, e as melhorias realizadas para atender ao contexto.

As seções 2 e 3 apresentam, respectivamente, a fábrica de software e sua estrutura interna, e a definição dos processos para a fábrica montada. A quarta seção descreve o estudo de caso onde foi aplicado o modelo proposto, caracterizando a experiência prática da disciplina. A seção 5 apresenta a avaliação sobre o processo final, mostrando lições aprendidas e indicando os trabalhos futuros acerca deste tema.

2. A Fábrica de Software "*Smart Factory*"

O cenário atual de comercialização de produtos de software apresenta duas particularidades ainda pouco discutidas: a atuação em mercados de software livre e desenvolvimento distribuído de software [ACM Queue 2004]. O software livre, ao contrário de seu mercado, hoje em dia é um fato, e vem sendo alvo de grande discussão no meio acadêmico e comercial.

Para simulação deste cenário definimos o conceito de *Smart Factory*, um modelo de fábrica distribuída de software livre, capaz de agir proativamente em relação à captação de negócios que se baseia na interseção entre uma empresa de serviços e uma fábrica de software comum. Nesse modelo, a fábrica de software é composta por uma equipe fixa, geograficamente distribuída, que será responsável pela captação de negócios e desenvolvimento inicial de um produto de software livre, que poderá ser posteriormente disponibilizado para que uma comunidade de desenvolvimento participe no processo de melhoria do mesmo.

2.1. Estrutura da Fábrica

Para a concepção da fábrica, foi definido inicialmente o modelo organizacional, apresentando a estrutura geral da OpenGadgets (Nome da fábrica de software simulada), e apresentando a fábrica como uma área da empresa, integrada aos demais setores da organização (Figura 1).

A OpenGadgets foi subdividida em três áreas diferentes (Suporte Organizacional, Comitê Gestor e Fábrica) compostas por unidades que representam papéis que devem ser desempenhados para o funcionamento do processo.

A área de Suporte Organizacional compreende unidades que fornecem infra-estrutura para o funcionamento da fábrica. O Comitê Gestor engloba os principais papéis gerenciais envolvidos com a produção da fábrica e esclarece a existência de uma forma de controle da produção, através da gerência da fábrica e da gerência comercial. Por fim, a Fábrica envolve os papéis relacionados às atividades produtivas da organização.

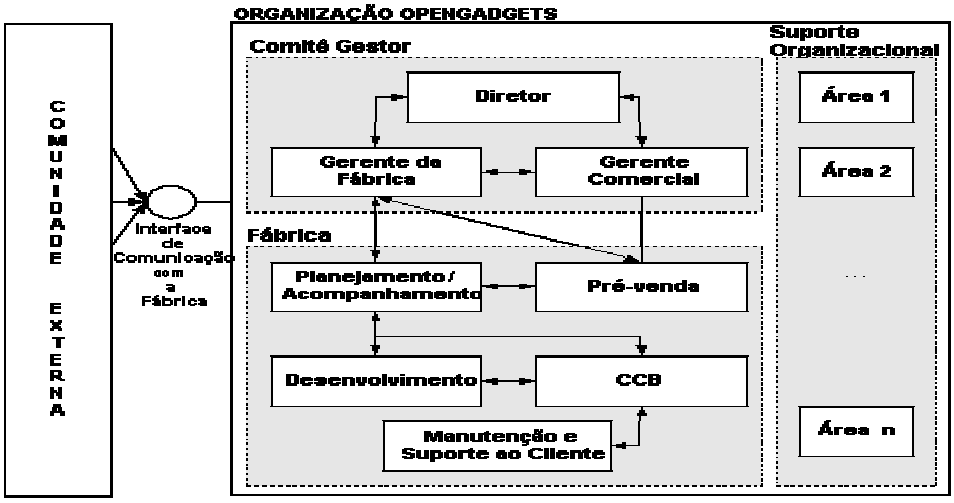

Figura 1 - Organização da Fábrica OpenGadgets

Além das três subdivisões internas à organização, a OpenGadgets foi projetada para suportar interações de desenvolvedores externos em seus projetos de código aberto, e para isso, provê uma interface pré-definida de comunicação com a comunidade. Esta interface é descrita em detalhes na seção 4.3.

Dentro da organização, o ambiente de estudo deste artigo é a Fábrica (Figura 2). Cada área da fábrica agrupa atividades com propósito comum, e solicita ou fornece serviços às demais áreas.

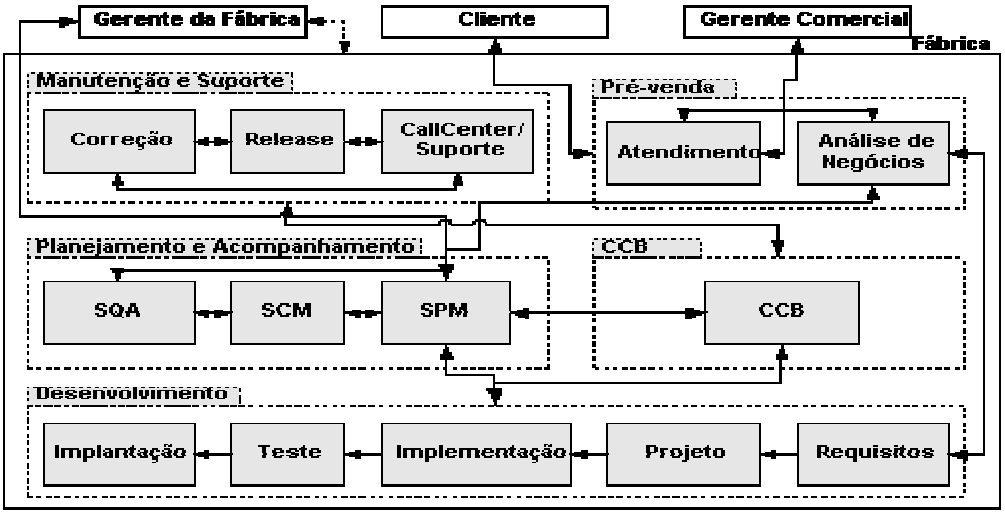

Figura 2 - Detalhamento da Fábrica

A área de *Pré-Venda* é composta pelas unidades de atendimento e de análise de negócios, que se complementam para melhor atendimento das necessidades do cliente e para maior sincronização com a área de produção.

O objetivo do *CCB (Change Control Board)* é garantir as mudanças aos produtos de trabalho estejam sendo controladas. É o grupo responsável por avaliar, classificar, atribuir e verificar a correção/implementação de cada pedido de mudança registrado. É ainda responsabilidade do CCB, após aprovação do produto de trabalho,

solicitar que este seja incluído na baseline, onde os trabalhos futuros serão baseados, ou seja, atua diretamente na geração do produto final.

A área de *Planejamento e Acompanhamento* envolve as atividades de garantia da qualidade, gerência de configuração e mudança e gerência de projetos, caracterizando as três unidades responsáveis pelo controle do desenvolvimento..A área de *Desenvolvimento* abrange as unidades de produção de cada fase de desenvolvimento: elaboração de requisitos (funcionais e não-funcionais), definição de arquitetura do software, implementação dos componentes baseando-se na arquitetura estabelecida, identificação dos erros do sistema através dos testes e elaboração de estratégias de implantação do software desenvolvido.

A área de *Manutenção e Suporte* auxilia a fábrica no atendimento aos clientes e captação de recursos, uma vez que a fábrica se destina à produção de software livre. Isto pode ser visto através das unidades de correção, release e call center/suporte. O objetivo da unidade de correção é atualizar, manter e realizar modificações em versões de Software e produtos da fábrica. Já a unidade de release tenta estabelecer os produtos que entram na versão do software e gerar o Release Notes de Correções efetuadas para a versão corrente. A unidade de Call Center e Suporte atende as reclamações e sugestões dos usuários ou clientes, direcionando as mesmas para dentro da fábrica.

3. Definição dos processos

Após a definição das áreas e da fábrica no processo produtivo, o processo de cada área foi instanciado, baseando-se no RUP – Rational Unified Process [KRUCHTEN 2000] e no CMMI – Capability Maturity Model Integration [CMMI 2004].

A definição formal destes processos se fazia necessária a fim de uniformizar a linguagem entre os membros da fábrica, uma vez que a proposta se baseava em implementar o conceito de desenvolvimento distribuído, o que exigia uma padronização a fim de facilitar a utilização do processo pelos membros da equipe. A descrição mais detalhada dos processos pode ser encontrada no web site da fábrica montado para a disciplina [OPENGADGETS 2004]. Os processos-chave definidos foram o de: manutenção, gerência de configuração, desenvolvimento, vendas, planejamento e acompanhamento de projetos e garantia da qualidade.

Todos os processos foram definidos com a intenção de serem o mais leve possível, mantendo entretanto a formalidade necessária para o desenvolvimento distribuído. Os processos de gerência, como se configuravam mais críticos para o sucesso da fábrica estão descritos em mais detalhes nas seções seguintes.

3.1. O processo de Planejamento e Acompanhamento de Projetos

O processo de Planejamento e Acompanhamento de Projetos foi definido a fim de controlar as atividades da equipe de desenvolvimento, uma vez que se fazia necessário atingir os objetivos do cliente dentro das restrições de escopo e prazo. Foi criada, então, uma instância baseada no RUP [KRUCHTEN 2000], com o objetivo de ser mais leve que o processo original, contemplando um conjunto menor de artefatos para acompanhar a execução dos projetos na fábrica. O processo contemplava as seguintes macro-atividades: início do projeto, planejamento do projeto, execução e acompanhamento de projeto, e finalização de etapa/projeto.

O início do projeto refere-se à autorização formal do inicio do projeto, onde seriam definidas as responsabilidades dos membros da fabrica e seria gerada uma ordem de serviço para a equipe de desenvolvimento começar a trabalhar, baseada nas restrições de escopo e prazo definidas nas propostas técnica e comercial.

O planejamento do projeto envolve o levantamento detalhado de todos os insumos necessários para a execução do projeto, envolvendo a elaboração de planilhas de riscos, planos de iteração e plano do projeto. No plano do projeto são definidos o cronograma e as tarefas a serem executadas e acompanhadas na atividade seguinte.

A atividade de execução e acompanhamento do projeto envolve atividades que, com base nos artefatos desenvolvidos no planejamento, mantenham atualizas as informações do andamento do projeto e do atendimento, ou não, das metas definidas, replanejando o projeto quando necessário. Ainda a manutenção do site do projeto deveria ser realizada a cada período pré-definido para manter a transparência da execução do processo ao cliente e para a equipe como um todo.

A finalização de etapa / projeto abrange as atividades necessárias para documentar e formalizar junto ao cliente a conclusão de um projeto ou de uma iteração, registrando ao final as lições aprendidas e liberando a equipe formalmente para outros projetos da fábrica através de reuniões de fechamento.

3.2. O processo de Garantia da Qualidade

A definição do processo de garantia da qualidade da OpenGadgets utilizou alguns princípios do modelo CMMI [CHRISSIS 2003], onde a área da garantia da qualidade de processo e de produto (PPQA) envolve

- A avaliação objetiva de processos, produtos de trabalho e serviços de acordo com normas, padrões e procedimentos existentes
- A identificação de não-conformidades durante as avaliações realizadas
- Acompanhamento da resolução das não-conformidades encontradas
- Comunicação das atividades da garantia da qualidade à equipe de projeto e alta gerência da organização.

O grande desafio durante a definição do processo de garantia da qualidade foi criar um conjunto de atividades e artefatos que não causassem sobrecarga de trabalho para a fábrica. Alinhando os conceitos de fábrica de software e software livre, era visível a dificuldade em conciliar os princípios da garantia da qualidade baseada no CMMI, onde havia a necessidade de evidências da execução do processo, com a linha de software livre proposta, que possuía uma interface da fábrica com a comunidade externa.

Baseando-se nestes fatores, o processo de garantia da qualidade foi definido contendo um conjunto de atividades que garantissem a execução dos demais processos dentro da fábrica. O processo engloba de maneira geral três macro-atividades:

- Planejamento das atividades de garantia da qualidade
- A execução da garantia da qualidade
- Finalização da garantia da qualidade

O planejamento da garantia da qualidade da Opengadgets envolve a elaboração e aprovação do plano de garantia da qualidade do projeto a ser desenvolvido. Durante a

definição do processo, o planejamento obteve uma importância significativa para a fábrica, já que seria necessário estabelecer regras para a execução das atividades dentro da fabrica, alinhando-se sempre aos demais processos da organização. A partir desta etapa, surgiu o plano de SQA, que continha os princípios que regiam a garantia da qualidade e as atividades de avaliação dos processos da fábrica.

O subprocesso de execução das atividades da garantia da qualidade foi definido com a preocupação em ser prático e objetivo, conforme apresentado no início desta subseção. Este subprocesso envolve a realização de auditorias nos processos e revisões nos artefatos produzidos durante o projeto, ambas planejadas no subprocesso anterior.

As atividades de avaliação dos processos e produtos de trabalho da OpenGadgets foram definidas para garantir a evidência da execução das revisões, solicitada pelo CMMI. Baseado nisto, foram criados vários artefatos para suportar estas atividades, como planilhas de avaliação de processos e produtos, planilhas de revisão dos artefatos, planilhas de testes e templates de relatórios de avaliações.

A atividade definida para a finalização da garantia da qualidade envolvia a elaboração do *post-mortem*, cujo objetivo é registrar os pontos fortes e fracos do projeto finalizado, para evitar a repetição dos erros e garantir a execução das boas praticas para os próximos projetos.

3.3. O processo de Gerência de Configuração

O processo de gerência de configuração também usou como base as definições do nível 2 do CMMI. Em suma, este processo está subdivido nas macro-atividades:

- Planejamento da configuração;
- Preparação do repositório de *baselines*;
- Controle de mudanças às *baselines*;
- Realização de auditorias;
- Acompanhamento dos desvios de processos levantados em auditorias;
- Realização de backups;

O planejamento é a fase onde serão identificados os itens de configuração e definidos todos os padrões (e.g. nome de arquivos, rótulos, *branches*) e procedimentos (e.g. auditorias, *releases*, *backups*) a serem seguidos por toda a equipe.

Uma vez planejada a configuração do sistema, o repositório de *baselines* precisa ser preparado. Para isso, necessitava-se levantar quais as ferramentas a serem utilizadas, instalá-las e configurá-las para o uso da equipe.

Depois de toda a infra-estrutura preparada, a equipe iniciava os trabalhos e a gerência de configuração executava atividades periódicas de gerência e verificação do repositório, ou seja, controlando sistematicamente as mudanças às *baselines* estabelecidas, verificando o bom uso dos padrões e cumprimento dos procedimentos definidos no planejamento e reportando atividades realizadas e status das *baselines*. Os desvios detectados nas auditorias serviriam de base para o amadurecimento do processo de gerência de configuração.

O processo de release deveria ser modificado entre os dois projetos. No segundo projeto, todas as alterações ao repositório de versões deveriam ser processadas pelo "integrador" e todos os membros do projeto deveriam seguir o processo de contribuição externa, ou seja, como a comunidade.

4. Estudo de Caso

Após a fase inicial de definições, a fábrica teve dois projetos de partida, um projeto piloto, para ajustes da fábrica, e um segundo projeto, para validação das melhorias e lições aprendidas durante o piloto, que tinham por objetivo prover subsídios de avaliação do processo de desenvolvimento e da organização da própria fábrica, além da interação da fábrica com a comunidade.

Ambos os projetos simularam um cenário, cliente e sistema reais para ser melhorado pela fábrica. Isso fez com que as lições aprendidas ao longo dos projetos tivessem ainda mais valor, pois valida a estrutura da fábrica e o processo de desenvolvimento adotado do ponto de vista prático.

4.1. Projeto Piloto – Canto Livre

O primeiro projeto, chamado de projeto piloto, foi destinado intencionalmente para validar os modelos adotados e para realização de ajustes finos na fábrica. Os membros da fábrica puderam se conhecer melhor, assim como os pontos fortes e fracos de cada um. Houve algumas trocas de papéis durante o curso do projeto como resultado da adaptação de cada membro à estrutura da fábrica, caracterizando um desenvolvimento extremamente voltado ao presencial e pouco distribuído. O objetivo do projeto piloto era prover uma interface *web* e criar novas funcionalidades em um sistema brasileiro de troca de música digital, de arquitetura *peer-to-peer* e baseado em uma política estrita de segurança, chamado Canto Livre.

Os requisitos iniciais foram listados pelo cliente em forma de uma RFP – *Request for Proposal* - e o time de pré-venda da fábrica ficou responsável por preparar as propostas técnicas e comercial com o apoio do time de desenvolvimento, porém, na prática, pouco apoio foi dado pela equipe de desenvolvimento durante essa fase, o que causou inúmeros problemas durante o decorrer do projeto, uma vez que compromissos inatingíveis, principalmente em termos de prazos, foram acordados com o cliente.

Com as propostas aceitas, o time de desenvolvimento elaborou o documento de requisitos e o documento de casos de uso, e baseado nas capacidades e disponibilidades reportadas durante as discussões iniciais, o gerente de projeto dividiu o trabalho a ser realizado entre os membros da equipe.

Dado que a divisão de trabalho não foi baseada em experiências anteriores da fábrica, houve várias falhas de julgamento que levaram o projeto ao fracasso total, onde prazos não foram atingidos e nem todos os requisitos implementados. Além disso, o processo de desenvolvimento não foi seguido completamente, pois era muito burocrático, incluindo tarefas e artefatos desnecessários.

Um ponto interessante a se ressaltar no desenvolvimento do piloto foi o esforço de alguns membros do time, especialmente na reta final do projeto, a fim de superar as dificuldades encontradas e entregar o sistema com o máximo de funcionalidades que

pudessem ser implementadas, o que caracteriza claramente uma fábrica nível 1 CMMI, baseada em esforços "heróicos".

Do ponto de vista da Garantia da Qualidade, houve uma distância muito grande entre o planejamento de SQA e as atividades executadas. As revisões dos artefatos do projeto estavam muito burocráticas, através da utilização de planilhas, o que tornava lento o processo de elaboração e aprovação dos artefatos. Além disto, as auditorias não foram realizadas, visto que o despreparo de toda a equipe na execução dos processos não deu margem à realização destas atividades. O planejamento de SQA abordou a coleta de três métricas, que também não chegou a ser realizada.

De um modo geral, o processo, que deveria garantir que a qualidade do projeto, não conseguiu atingir seu objetivo, e foi deixado em segundo plano, para dar espaço ao desenvolvimento dos requisitos acordados na RFP.

No tocante à gerência de configuração, a equipe definiu que seriam usadas ferramentas comumente usadas pela comunidade de Software Livre para desenvolvimento de aplicações de forma distribuída. A ferramenta escolhida foi o Tigris, da Tigris.org [TIGRIS 2004] que contava, entre outras coisas, com um sistema para controle de versões [CVS 2004] e uma ferramenta para controle de mudanças (IssueZilla), que são os 2 tipos de ferramentas básicas para o controle de configuração. O Tigris é um sistema disponível e acessível pela Internet, ou seja, os desenvolvedores e a comunidade teriam acesso livre à ferramenta de forma simples.

Em adição a estas ferramentas, foram usadas outras ferramentas auxiliares, entre elas, sistema para compartilhamento de conhecimento e lista de e-mails. O sistema de compartilhamento de conhecimento, *Atlassian Confluence* [ATLASSIAN 2004] também foi usado como web-site, onde todos os processos foram descritos e o projeto era acompanhado pelo cliente.

O principal problema identificado com relação à gerência de configuração foram os releases. Não se tinha um controle muito eficaz das alterações sendo feitas no CVS. Todos trabalhavam nas últimas versões do repositório, ou seja, em código não comprovadamente estável, o que ia de encontro aos preceitos do CMMI.

Ao final do projeto, o cliente estava insatisfeito com o serviço executado, uma vez que muitos requisitos não foram implementados e os poucos que foram tinham erros que impediram que as necessidades do cliente fossem atendidas. Uma constatação também bastante pertinente por parte do cliente foi a falta de comunicação dentro do grupo e a falta de transparência do processo de desenvolvimento.

4.2. Calibragem dos processos

Os resultados do projeto piloto foram desestimulantes para a fábrica, o que diminuiu a motivação da equipe. Entretanto, havia o segundo projeto a ser enfrentado, ressaltando então a necessidade de se encontrar uma solução ou um conjunto de soluções para os problemas encontrados, antes do início do projeto final.

Foram realizadas algumas reuniões que resultaram em uma lista de lições aprendidas e soluções propostas, gerando algumas mudanças no processo de desenvolvimento. Dentre as mudanças ocorridas, as principais foram:

- Foi estabelecida uma reunião semanal para acompanhamento das atividades, uma vez que a comunicação se mostrou falha no piloto;
- O processo de revisão de artefatos foi simplificado, passando a requerer apenas um revisor;
- Foram abolidos alguns artefatos de SQA e automatizadas algumas atividades de SCM;
- Foram criados grupos (analista, arquiteto e engenheiro) dentro da equipe de desenvolvimento e cada grupo teve um líder associado.
- Foi realizado um controle de desenvolvimento dos requisitos via Tigris
- Foi implementado um controle semanal das atividades através do site do projeto.

Depois de realizados os ajustes, o time se sentia bem mais confiante no sucesso da fábrica e iniciou-se assim o projeto Tupan.

4.3. Projeto Real – Tupan

O segundo projeto teve as mesmas fases do projeto anterior, porém, o envolvimento da equipe de desenvolvimento desde o início, como a pré-venda, caracterizou o exercício do conceito de *Smart Factory* e fez com que as estimativas fossem bem mais precisas.

O escopo do projeto foi realizar a reengenharia de todo o sistema desenvolvido no piloto e implementar funcionalidades adicionais, como integração com mecanismo de *instant messaging*, suporte a licenças *Creative Commons* [Creative Commons 2004] e aumentar a robustez do sistema.

Mesmo com muito trabalho a ser feito em pouco tempo, todo o time estava determinado a atingir os resultados esperados pelo cliente da melhor forma possível e dentro do cronograma proposto.

Uma das mudanças mais significativas foi a divisão do time de desenvolvimento em grupos que eram coordenados por líderes de grupos. O gerente de projeto interagia com os líderes de grupo (analistas, arquitetos e engenheiros) para consolidar o andamento do projeto em relatórios semanais e planejar as atividades da próxima semana. Os líderes de grupo dividiam o trabalho de sua área entre os membros de seu grupo. Dado que o líder de cada grupo entendia melhor o problema a ser atacado e conhecia as capacidades e disponibilidades de cada um dos integrantes do grupo, fazia mais sentido adotar essa estratégia, ao invés de deixar esta delegação a cargo do gerente de projetos. Ao final do projeto ficou comprovado o sucesso desta prática.

Outra mudança significativa foi a forma de interação com o cliente. A equipe de SQA iniciou um acompanhamento semanal com o cliente em forma de relatórios de satisfação baseados nas responsabilidades definidas para a semana e na interação com a fábrica como um todo. Esta estratégia forçou os membros da equipe a manterem o foco em objetivos claros e de duração bem determinada, ao invés de se preocupar com objetivos de escopo maior e de difícil acompanhamento.

Além disto, a sobrecarga de trabalho existente no processo anterior de SQA foi minimizada: as atividades de avaliação dos processos e produtos de trabalho foram

redefinidas visando a agilidade e o objetivismo, através da utilização da ferramenta de controle de mudanças para auxiliar algumas atividades de SQA.

As revisões dos artefatos passaram a ser realizadas de maneira mais ágil, usando o Tigris, onde a mudança do status de uma atividade de elaboração de um artefato indicava o momento de revisão do mesmo. Além disso, os questionamentos dos revisores eram registrados no próprio documento, através do uso de comentários e controle de alterações. Dessa forma, foi garantida a evidência da execução das revisões (solicitado pelo CMMI), e a atividade se tornava independente da localização dos membros da fábrica ou comunidade, já que a ferramenta utilizada era web.

Com relação às atividades de auditoria de processo, foram utilizados *checklists* para verificar se os processos da fábrica estavam sendo seguidos pelos seus membros. As entrevistas com os membros da fábrica foram realizadas remotamente e consolidadas pelo SQA, que apresentou o resultado da auditoria nas reuniões semanais da fábrica.

O processo de Gerência de Configuração também sofreu modificações. O replanejamento contou com o mesmo conjunto de ferramentas e itens de configuração da 1ª fase, com pequenas modificações no esquema de aprovações, auditorias sob demandas e o controle separado da documentação e artefatos de código. O controle do código-fonte foi feito através de uma *baseline* chamada "*baseline* de *release*" e incorporou algumas idéias do desenvolvimento de softwares livre. Ainda foram adotadas algumas medidas para um controle mais eficaz das alterações feitas no CVS: o bloqueio do repositório e criação da função do integrador.

O processo de release foi modificado da primeira para a segunda fase do projeto. Não se entendia mais "*release*" como uma entrega formal ao "cliente", como era comumente usado em um desenvolvimento tipo "catedral". Um "release" passou a ser entendido como toda versão pronta para uso, seja para a comunidade, seja para a equipe de desenvolvimento interna. Sendo assim, foram criados vários tipos de releases, com diferentes objetivos e neste contexto, e ficou decidido que somente as mudanças completas deveriam ser adicionadas ao controle de versões.

Esta resolução fez com que o repositório fosse povoado somente com versões prontas para um possível release, o que não acontecia anteriormente, pois todos os desenvolvedores enviavam suas alterações ao repositório diariamente, sem garantir a estabilidade do sistema. Muitas vezes isso causava impactos no trabalho dos outros desenvolvedores que trabalhavam sob as últimas versões encontradas no CVS, o que vai contra as práticas do CMMI, pois não era um conjunto de artefatos estável.

Assim, ficou decidido que todos os desenvolvedores, internos ou externos, teriam somente acesso de leitura ao repositório de versões e que todos eles deveriam seguir o processo de contribuição externa, explicitado na subseção 4.3. Dessa forma, embora perdêssemos um pouco da agilidade do desenvolvimento, teríamos um repositório estável a qualquer momento e também conseguiríamos simular o comportamento de uma futura comunidade de colaboradores.

Ao final do segundo projeto, todos requisitos foram implementados dentro do prazo estipulado, terminando o desenvolvimento com os maiores conceitos em todos os quesitos no questionário de satisfação. Infelizmente, não foi possível exercitar a interação com a comunidade devido ao curto espaço de tempo do projeto, porém, o

processo de contribuição foi simulado com os próprios integrantes e está melhor descrito na próxima subseção.

4.3. Interação com a Comunidade *Open Source*

O processo de interação da fábrica com a comunidade externa está muito ligado ao processo de gerência da configuração do software e de suas mudanças. Ou seja, há a necessidade de um controle bem definido da configuração e de como as mudanças do software afetarão a comunidade que está contribuindo ou usando os produtos desenvolvidos. O processo de contribuições externas foi criado para que as alterações ao repositório fossem feitas de forma controlada.

Para que uma alteração fosse feita em um release qualquer, o colaborador deveria inicialmente escolher um problema ou uma funcionalidade a ser implementada. Feito isso, ele deveria utilizar a versão de release mais recente para implementar a correção ou nova funcionalidade. Essa implementação deveria ser feita na área local do colaborador, pois este só possuía acesso de leitura ao repositório. Dessa forma, ao isolarmos o desenvolvimento, estaríamos simulando o funcionamento de um branch.

Terminada a implementação, o colaborador deveria preencher um formulário e submetê-lo ao integrador, membro da equipe do projeto responsável por fazer as integrações das alterações no CVS. Este formulário possui, além da própria contribuição anexada, informações básicas com relação à mesma, como: Nome contato do colaborador; problema ou funcionalidade implementada; descrição detalhada da implementação; dependências funcionais e de compilação; testes executados.

Uma vez submetido o formulário, o integrador deveria fazer alguns testes de sanidade e incorporar as alterações ao controle de versões. Um novo release deveria ser produzido e comunicado aos interessados. Toda nova contribuição deveria daí por diante utilizar esse *release* como base. Caso mais de um formulário fosse submetido ao mesmo tempo, o primeiro era processado e se, após os testes de sanidade, nada anormal tivesse sido encontrado, a contribuição submetida por último deveria ser rejeitada e atualizada com o release recém liberado.

5. Conclusões

Após o desenvolvimento de dois projetos foi verificada uma evolução efetiva em relação aos processos executados pela fábrica, enfatizando a diferença entre uma fabrica de software comum e uma com equipe geograficamente distribuída.

Para a adaptação dos processos foi possível concluir que os mesmos devem ser bastante flexíveis viabilizando a sua execução de forma distribuída, ou seja, existe a necessidade utilizar uma modelagem que facilite a redução dos impactos produzidos pelas características da fábrica. A definição e a execução dos processos devem ser evolucionárias e não revolucionárias.

O comprometimento na execução dos processos foi um dos fatores de grande atenção, pois foi bastante visível diferença entre os dois projetos, onde houve a necessidade de intensificar a atuação do gerenciamento do projeto e prover uma maior utilização de ferramentas que aproximassem os desenvolvedores, a fábrica e o cliente.

Podemos concluir que distribuição e liberdade são a base do modelo onde processos devem ser altamente ágeis, porém é preciso ter cuidado com excessos. A implantação dos processos definidos pelo modelo *Smart Factory* permitiu uma maior rapidez na absorção cultural com a possibilidade de nos próximos projetos a distribuição geográfica entre os membros da equipe seja bem mais transparente e robusta.

Referências

ACM Queue, Distributed Development, Vol. 1 No. 9 - December/January 2003-2004. Disponível em: http://www.acmqueue.com/

"Atlassian Confluence", http://www.atlassian.com/confluence, Último acesso em Agosto de 2004.

Bays, Michael E. (1999). "Software Release Methodology". 1a. Edição. Prentice Hall PTR.

Brito, R., Ferreira, P., Silva, K., Burégio, V. and Leite, I. (2004). "Desenvolvimento distribuído: Um estudo de caso de uma fábrica de software livre"

"Capability Maturiry Model Integration", http://www.sei.cmu.edu/cmmi/cmmi.html, Último aceso em Agosto de 2004.

Chrissis, Mary Beth; Konrad, Mike; Shrum, Sandy. (2003). "CMMI – Guidelines for Process Integration and Product Improvement". Boston.

"CVS – Concurrent Version System", http://www.cvshome.org, Último acesso em Agosto de 2004.

Hecker, F., (1999). "Setting Up Shop:The Business of Open-Source Software", IEEE Software January/February Edition, pp 45-51.

"IN953 – Engenharia de Software: fábricas de software livre", http://www.cin.ufpe.br/~in953/, Último acesso em Agosto de 2004.

Kruchten, Philippe. (2000) "The Rational Unified Process An Introduction Second Edition". Addison-Wesley, Boston.

"Opengadgets", Disponível em: http://www.cin.ufpe.br/~opengadgets. guest: guest. Último acesso em Agosto de 2004.

"Pair Programming.com", http://www.pairprogramming.com/, último acesso em Agosto de 2004

Raymond, E. S., The Cathedral and the Bazaar. http://catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html#catbmain, Último acesso em Agosto de 2004.

Rocha, Cristine E. Dias, Almeida, Thiago J. M., Rocha, Thayssa Águila. (2003). "Implantação de uma fábrica de software". Centro Universitário do Pará, Brasil.

"Tigris.org – Open Source Software Engineering", http://www.tigris.org, Último acesso em Agosto de 2004.