Criação Ágil de uma Fábrica de Software com Membros Distribuídos

D. M. Cabral, S. V. Alves, R. A. A. Fagundes, L. C. Neto e W. M. A. Filho, *Mestrandos*, UFPE

Abstract — **The increasing need for software products has brought to attenction issues related to classic software development processes, where adaptations are necessary to follow the time-to-market preserving productivity and quality.**

Our work reports an experience on developing and executing an adapted process, based on RUP and eXtreme Programming, intended to produce software rapidly under diverging circumstances like geographic sparseness.

Keywords — **work coordination, distributed software development, process-centred environments.**

I. INTRODUÇÃO

O aumento da demanda por sistemas de software, associado à importância do papel por eles desempenhado na sociedade atual, tem levado a uma preocupação constante com a produtividade no desenvolvimento e a qualidade dos produtos criados.

Resultados obtidos ao logo do tempo mostram que vários projetos de desenvolvimento de software falham devido, principalmente, a problemas como baixa qualidade, baixa produtividade, incompreensão das necessidades do usuário, erros de construção, testes insuficientes, prazos ultrapassados e custos altos.

Na busca de soluções para os problemas presentes no desenvolvimento de software, as empresas produtoras de software buscam constantemente a melhoria de seus processos, implantando metodologias de desenvolvimento bem definidas, sejam elas estruturadas ou orientada a objetos.

Nesse sentido, muitas organizações de software estão adotando uma nova estrutura organizacional orientada a projetos, denominada Fábrica de Software. O conceito de Fábrica de Software está baseado na idéia de prover uma linha de produção de soluções que atendam às necessidades específicas de cada cliente através da formalização de todas as atividades e seus produtos, trabalhando em linha de produção, com etapas e tarefas bem definidas para cada tipo de profissional, indo da produtividade da linha de produção a rotinas de controle de qualidade[1].

Uma Fábrica de Software tem como objetivo aumentar a qualidade do produto, ajustando-o às necessidades do cliente, reduzir custos e prover mais produtividade no desenvolvimento do software, oferecendo assim produtos estáveis, expansíveis, adaptáveis e de qualidade[2].

Inserido nesse contexto, este artigo relata a experiência na disciplina de Engenharia de Software do mestrado da Universidade Federal de Pernambuco (UFPE). O objetivo da mesma era a definição e implantação de uma fábrica de software, composta por dez membros, em um período máximo de 4 meses.

A seção 2 introduz a definição do processo de desenvolvimento adotado pela fábrica e a seção 3 descreve em linhas gerais o RUP; a seção 4 descreve como o RUP foi adaptado para a fábrica; a seção 5 descreve o projeto CADEX e a seção 6 relata o experimento com o projeto CADEX; a seção 7 apresenta as boas práticas do processo e a seção 8 apresenta as dificuldades enfrentadas. Finalmente, na seção 9, são apresentadas as conclusões finais deste trabalho.

II. ESCOLHENDO O PROCESSO

Para o perfeito funcionamento de uma Fábrica de Software é fundamental a adoção de um processo de desenvolvimento que defina suas atividades, produtos e responsáveis pelas etapas do ciclo de vida do software.

Nesse sentido, fez-se uma contraposição de alguns processos de engenharia de software existentes, como Rational Unified Process (RUP)[3], eXtreme Programming(XP)[4] e Distributed eXtreme Programming(DXP)[5], e adotou-se o RUP como a metodologia base para a formulação do processo de desenvolvimento. Partindo deste processo procurou-se detalhar as atividades constituintes de cada fase do ciclo de vida, buscando a simplificação, para atender as necessidades e porte da fábrica.

A escolha de RUP deve-se a alguns fatores. Primeiro porque RUP é uma estrutura de processo que pode ser instanciado para uma empresa ou projeto, conforme o tipo de organização e o domínio da aplicação.

Segundo porque diferente dos processos clássicos de engenharia de software, onde o desenvolvimento ocorre linearmente, em RUP o desenvolvimento é iterativo e incremental. Dessa forma, muitos problemas são atenuados, uma vez que inconsistências entre exigências, construções e implementações são detectadas no início do ciclo de vida, possibilitando à equipe utilizar as lições aprendidas para melhorar continuamente o processo.

E finalmente, porque RUP descreve como projetar uma arquitetura sólida e flexível, proporcionando assim o reuso de software mais efetivo.

A seguir, será apresentada uma descrição resumida de RUP.

III. BREVE VISÃO DE RUP

O RUP foi criado para ser aplicável a uma grande classe de projetos diferentes, sendo por isso considerado um framework genérico para a definição de processos de desenvolvimento [6]. Isso significa que ele pode ser configurado para ser usado eficientemente.

O RUP descreve os papéis e as atividades que cada membro da equipe de projeto deve desempenhar ao longo do ciclo de desenvolvimento do software, os artefatos que devem ser gerados como resultado destas atividades e os fluxos. Os Fluxos descrevem a seqüência de atividades que produzem algum resultado de valor e mostram as iterações entre membros da equipe, como por exemplo, o fluxo de gerência e o fluxo de desenvolvimento.

O RUP considera que o ciclo de vida de um sistema consiste de quatro fases distintas, divididas em iterações e com objetivos e marcos bem definidos. As fases indicam a maturidade do sistema e são: Concepção, Elaboração, Construção e Transição.

Na fase de Concepção deve-se definir o escopo do projeto e a viabilidade técnica e financeira. Na fase de elaboração deve-se detalhar o planejamento feito na fase anterior, projetar uma arquitetura estável e robusta do sistema, e eliminar riscos de requisitos, reusabilidade de componentes e viabilidade técnica. Na fase de construção um produto completo é desenvolvido de forma iterativa e incremental, alcançando versões usáveis do sistema (alfa, beta e outros lançamentos de teste), minimizando custos de desenvolvimento, otimizando recursos e evitando re-trabalho. Finalmente, na fase de transição deve-se garantir que o software estará disponível aos usuários finais, podendo inclusive, ser iniciado um novo ciclo de desenvolvimento para a evolução do produto, o que envolveria todas as fases novamente.

Todas as fases descritas acima são concluídas com um marco (*milestone*) para avaliar o progresso e verificar se os objetivos específicos da fase foram atingidos. Cada fase agrupa várias iterações, também finalizadas com marcos secundários, que são organizadas em fluxos (*workflows*) que descrevem o que deve ser feito em termos de atividades, responsáveis e artefatos (ver figura 1). O RUP fornece ainda modelos (*templates*) para cada artefato e *guidelines* para a execução de suas atividades[6].

O RUP organiza o conteúdo em seis fluxos principais chamados de fluxos de processo: Modelagem do Negócio, Requisitos, Análise e Projeto, Implementação, Teste e Distribuição; e três fluxos complementares chamados de fluxos de suporte: Gerência de Projeto, Gerência de Configuração e Mudanças, e Configuração do Ambiente. Para cada fluxo, RUP define um conjunto de artefatos, atividades e papéis

Segue abaixo uma descrição resumida dos fluxos de processo e de suporte usados pela fábrica.

Modelagem do Negócio (*Business Modeling*) — a intenção é entender o contexto da organização cliente para desenvolver um modelo de negócios, assegurando que clientes, usuários e desenvolvedores tenham a mesma visão da organização.

Requisitos (*Requirements*) — envolve entender e gerenciar o contexto do sistema, através da captura de requisitos.

Análise e Projeto (*Analysis and Design*) — os requisitos capturados no fluxo anterior são transformados em especificações de como implementar o sistema. Modelos de análise e projeto são gerados com a utilização de UML.

Implementação (*Implementation*) — o projeto é transformado em código: classes, objetos, etc. A estratégia é desenvolver o sistema em camadas, particionando em subsistemas.

Teste (*Test*) — responsável pela verificação total do sistema, além de verificar se todos os requisitos foram cumpridos.

Distribuição (*Deployment*) — responsável pelo empacotamento do produto, instalação, treinamento de usuários e distribuição do produto.

Gerência de Projeto (*Project Management*) — envolve o planejamento e monitoramento do projeto.

Gerência de Configuração e Mudanças (*Configuration and Change Management*) — envolve todas as tarefas relacionadas com a gerência de versões dos artefatos, *releases* e gerência de mudanças de requisitos.

Configuração do Ambiente (*Environment*) — envolve a adaptação e organização do ambiente de trabalho para a equipe do projeto e a seleção de ferramentas de desenvolvimento.

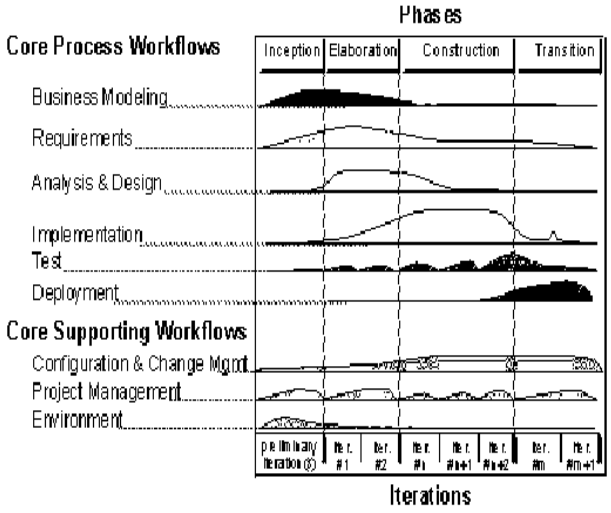

Fig. 1. Fases, iterações e fluxos do RUP

IV. ADAPTAÇÃO DO RUP

As principais características e fases do ciclo de vida do RUP foram incorporadas ao processo da fábrica. Para as fases foram definidos cinco fluxos, inspirados nos nove fluxos do RUP. A decisão de simplificar o RUP foi tomada devido ao tamanho da equipe e baseada na idéia apresentada em *Making RUP Agile*[7]. A redução no número de fluxos visou diminuir o número de documentações e artefatos criados, aproximando atividades que geravam artefatos dependentes num único fluxo, e inserir algumas das sugestões de qualidade apontadas pela ISO9000-3[8], não suportadas pelo RUP e definidas em *Uso do Processo RUP na Implantação da ISO9000-3*[6].

Foram gerados cinco fluxos: Vendas, Desenvolvimento, Gerência de Projetos, Qualidade e Gerência de Configuração,

sendo os dois primeiros de fluxo de processo e os demais, fluxos de suporte.

Os fluxos de Análise e Projeto, Implementação, Distribuição e Testes, foram agrupados em um único fluxo denominado Desenvolvimento. As atividades do fluxo de Configuração do Ambiente foram incorporadas no fluxo de Gerência de Configuração. O fluxo de Modelagem de Negócios foi agrupado com o fluxo de Requisitos, dando origem ao fluxo de Vendas, e foi definido um novo fluxo denominado de Qualidade.

Para cada um dos fluxos foi delegado um responsável como ponto de referência na comunicação com o gerente de projeto. A única exceção foi no fluxo de desenvolvimento, onde foram delegados dois responsáveis, um para análise e projeto e outro para implementação. Os papéis foram distribuídos considerando-se critérios como a disponibilidade de tempo e o know-how de cada integrante. No processo também foi permitido que cada membro pudesse ter mais de um cargo e cada cargo fosse ocupado por pelo menos dois membros da fábrica, que deveriam trabalhar juntos. Esse último argumento foi baseado na idéia atribuída no *Pair Programming* do XP como boa prática difusora de conhecimento, sendo aplicada não somente à programação, mas também a todos os demais fluxos. Segue abaixo uma descrição detalhada dos fluxos de processo usados pela fábrica.

Vendas — fluxo acionado no início do projeto. É responsável pela centralização das negociações, especificação das necessidades do cliente, identificação de visões de negócio e tecnológica e identificação de requisitos. Ao final do fluxo são elaborados o documento de requisitos funcionais e não funcionais, o cronograma macro do projeto, as propostas técnica e comercial, as estimativas de esforço e de custos e a SLA[9], concluindo-se, nessa última, a negociação com o cliente. Esse fluxo também é responsável pela criação e manutenção do *site* da fábrica, formando juntamente com o e-mail o canal de comunicação com o cliente. O papel responsável por esse fluxo é o gerente comercial.

Desenvolvimento — tem início após a coleta de requisitos de software realizada no fluxo de vendas. É responsável por garantir a transição correta entre a visão de negócio e os requisitos para uma visão de realização, ou implementação. Neste fluxo, uma equipe de analistas de sistema determina, produz e mantém: padrões de análise e de projeto, soluções, reuso e componentização. Esta equipe é, também, responsável pelo processo de construção do software conforme as especificações e pela manutenção de uma infra-estrutura de testes que permita a execução e avaliação de seus resultados.

Na fase de desenvolvimento, existem três papéis: o engenheiro de software, o analista de sistemas e o arquiteto de software, sendo esse último o engenheiro com maior experiência de programação na equipe. O analista e o arquiteto de software são os dois responsáveis pelo fluxo.

Dependendo do tamanho do projeto, pode existir mais de um arquiteto de software, cada um coordenando uma equipe de engenheiros. Essa distribuição é importante, pois, os arquitetos funcionam como pontos de referência para que a gerência de projetos possa acompanhar os marcos definidos no planejamento da implementação. O arquiteto de software é também o responsável por validar e discutir os artefatos gerados pelos analistas de sistema.

O analista de sistema é o responsável pela descrição dos casos de uso e testes de aceitação e pela criação dos diagramas de caso de uso, conceitual, de classes, seqüência e de entidade-relacionamento. Esses artefatos são criados baseados no documento de requisitos definido no fluxo de vendas.

Para facilitar a programação e o entendimento de novas linhas de código entre todos os engenheiros e arquitetos de software distribuídos remotamente na fábrica, foi adotado um padrão de convenção de codificação. Essa convenção foi uma adaptação resumida do padrão Code Conventions for the JavaTM Programming Language da Sun Microsystems[10]. Além disso, a programação foi feita utilizando Remote Pair Programming, extraída do Distributed eXtreme Programming[5].

Os fluxos de suporte compreendem atividades necessárias à execução dos fluxos de processo definidos no RUP. Segue, abaixo, uma descrição detalhada sobre os fluxos de suporte.

Qualidade — responsável pelas atividades diretamente ligadas à qualidade, buscando conformidade com padrões internacionais. O gerente de qualidade é o responsável por esse fluxo, sendo de seu encargo a gestão da qualidade durante o processo de desenvolvimento, coletando métricas e produzindo indicadores de qualidade, organizando workshops, distribuindo revisão de artefatos e garantindo que o produto final atenda às especificações e à qualidade exigida. Auxilia, ainda, na definição do contrato do fluxo de vendas, esclarecendo pontos como renegociações de especificações, e na gestão de pessoas do fluxo de gerência de projetos, com a realização de avaliações internas.

No que se refere à prática de workshops, eles são realizados como meio de disseminação do know-how das principais atividades dos fluxos entre os membros da fábrica. Os workshops devem ser aplicados no início, para nivelar o conhecimento da equipe, e no final de cada projeto, para identificar as boas e más práticas aplicadas.

Além disso, o processo de revisão e garantia da qualidade dos artefatos deve ser feita de maneira remota, devido à distribuição da equipe que impossibilitava o trabalho em grupo fisicamente presente. Cada artefato gerado deve seguir um padrão de estrutura de conteúdo[11] — nome do projeto, tabela de alterações e revisões, visão geral do documento, referência, entre outros — e apresentação, e ser revisado por no mínimo dois outros membros diferentes do seu criador, antes de ser publicado ao cliente. A delegação dos revisores é feita pelo gerente de qualidade, juntamente com o gerente de projeto, de maneira que os revisores dos artefatos sejam os mesmos membros que irão utilizá-lo em outra etapa do processo.

No fluxo de qualidade da fábrica o gerente de qualidade deve gerar um relatório periódico com métricas e indicadores da qualidade, tanto do produto quanto do processo. Este documento é necessário para buscar, a médio prazo, uma melhoria contínua do processo e assim alcançar uma maior qualidade e produtividade. As métricas utilizadas nesse relatório serão: quantidades de auditorias realizadas, esforço por ponto de caso de uso, esforço por fluxo do processo de desenvolvimento,

custo do projeto, número de bugs encontrados. Já os indicadores serão: esforço planejado *versus* esforço realizado, número de bugs por 1000 linhas de código, variação de cronograma, variação de custo, satisfação do cliente e taxa de evolução da quantidade de auditorias realizadas.

Gerência de Projetos — responsável pelo planejamento e controle do projeto em suas diversas etapas:alocação dos recursos, gerenciamento de riscos, estimativa dos prazos de desenvolvimento. Porém o acompanhamento deve ser feito de maneira remota a partir de ferramentas de Instant Message e de envio de mensagens por e-mails e para celular.

Gerência de Configuração — garante um ambiente operacional altamente funcional. O responsável por esse fluxo é o gerente de configuração e sua tarefa é realizar o suporte, atualização e manutenção da infra-estrutura tecnológica comum utilizada na fábrica.

Um dos encargos desse fluxo é disponibilizar um ambiente padrão e centralizado no qual os engenheiros de software e demais integrantes pudessem trabalhar organizadamente, com integração constante, em cima de um mesmo código e de outros artefatos gerados pela fábrica, como uma solução para o problema da distribuição e incompatibilidade de tempo da equipe. Assim, é de responsabilidade do engenheiro de configuração instalar serviços de Concurrent Versions System (CVS)[12] e Instant Message, bem como disponibilizar padrões de nomenclatura e uma hierarquia de diretórios para o armazenamento dos artefatos, proporcionando uma linguagem comum de comunicação na criação de artefatos pelos membros.

Também é responsabilidade do gerente de configuração, disponibilizar um esqueleto da arquitetura do projeto configurado para a ferramenta de desenvolvimento padrão utilizada na fábrica. Esse esqueleto serve de base para os engenheiros saírem inserindo funcionalidades, alcançando, assim, o produto final desejado.

Os artefatos mantidos foram os fundamentais e que adicionavam valor ao processo. Os artefatos possuem templates, que indicam como devem ser feitos. Além disso, cada artefato recebeu um formato padronizado estabelecido pelo processo especificado. Artefatos de modelagem de sistema, como modelo de casos de uso, modelo da arquitetura, modelo de classes e modelo de seqüência de atividades, foram criados utilizando UML.

V. PROJETO CADEX

Guiados pelas diretrizes e considerações apresentadas nas seções anteriores, o primeiro projeto foi iniciado em julho de 2003. O projeto, denominado CADEX, possuía como objetivo principal desenvolver um sistema de informação para o controle de alunos egressos de uma universidade.

O sistema deveria ser capaz de guardar informações da evolução profissional e acadêmica de seus ex-alunos, bem como realizar pesquisas personalizadas sobre esses dados, possibilitando à instituição acompanhar o perfil dos profissionais que ela está formando e a absorção dos mesmos pelo mercado de trabalho.

Além disso, o sistema deveria fornecer controle de acesso na manutenção, e mecanismos de buscas para a formação de listas de discussão por e-mail. O objetivo dessa última funcionalidade era auxiliar na aproximação dos profissionais formados pela instituição de suas atividades e eventos científicos, ajudando na produção de trabalhos de qualidade seja no ambiente acadêmico ou comercial.

VI. EXPERIMENTO

Nessa seção será apresentada a experiência de utilização dos fluxos anteriormente propostos pela fábrica. Porém, a execução não foi imediatamente em cima do projeto CADEX, uma vez que inicialmente foi feita uma negociação do cliente com o gerente comercial para o desenvolvimento de um projeto piloto, PILOT, onde o próprio cliente pôde avaliar e entender o funcionamento do processo da fábrica, antes da contratação de seus serviços. O projeto PILOT foi um protótipo com as funcionalidades de cadastro e manipulação de ex-alunos, sem inserir nele o conceito de turmas, listas de discussão, pesquisas com filtros e dados estatísticos.

Na Gerência de Configuração, procurou-se utilizar ferramentas de fácil uso que dessem suporte a múltiplas plataformas e ao desenvolvimento distribuído. Quando possível, também se buscou utilizar ferramentas gratuitas, numa tentativa de redução de custos. Como resultado foram escolhidos o CódigoLivre[13] como servidor CVS, ICQ Lite, MSN Messenger, Trillian e Miranda como ferramentas de Instant Messange e JBuilder 6 e 8 como *Integrated Development Environments* (IDE).

O esqueleto da arquitetura dos projetos foi desenvolvido baseado em JBuilder, utilizando Servlets[14] e Freemarker[15], numa arquitetura em três camadas empregando os padrões Façade, Abstract Factory, Factory Method e Singleton[16]. Nessa arquitetura os Servlets implementavam a camada de regras de negócio e a de acesso à banco de dados. O Freemarker foi utilizado para desenvolver a interface gráfica independente do código de negócios da aplicação, para que o mesmo pudesse facilmente ser adaptado.

Devido ao tamanho do projeto as fases foram divididas em poucas iterações, estas com duração de uma semana no máximo. As fases de Concepção, Elaboração e Transição foram realizadas em uma iteração, enquanto que a de Construção em duas. O objetivo da primeira iteração foi a implementação e validação de funcionalidades básicas, como cadastro e manutenção de ex-alunos e turmas. A segunda iteração enfocou as funcionalidades mais complexas, como pesquisas com filtros e estatísticas, além do envio de e-mails para as listas de turmas.

Os marcos das iterações foram acompanhados diariamente pelo gerente de projeto através de comunicações por e-mail, uso de Instant Messange e envio de mensagens para celular. Também foi uma prática da gerência, a realização de uma reunião semanal, para discutir o andamento do processo. Nestas reuniões os responsáveis por cada fluxo apresentavam o estágio das atividades e as dificuldades encontradas.

O desenvolvimento foi feito baseado na idéia do Pair Programming, utilizando ferramentas de Instant Message para a comunicação entre as duplas, quando a presença física era

inviável. A programação em par era agendada entre os próprios engenheiros e avisada ao gerente do projeto para que o mesmo pudesse planejar e acompanhar o andamento das atividades. Porém, ainda no projeto PILOT, percebeu-se dificuldades com esse tipo de prática, inviabilizando-a no projeto CADEX. A solução encontrada foi a programação individual.

Para a garantia da qualidade dos artefatos, o gerente de qualidade gerava quinzenalmente um relatório de auditoria, com o objetivo de identificar falhas na execução do processo. Além disso, ao final de cada projeto foi criado um relatório de indicadores e métricas, necessário para conhecer o desempenho do processo, a qualidade requerida do produto e identificar se o projeto foi desenvolvido dentro do prazo e custo estimado.

VII. BOAS PRÁTICAS

Durante a execução do projeto foram identificadas as seguintes boas práticas:

- Política de revisões de artefatos por no mínimo dois membros da equipe, antes da publicação para o cliente;
- Transmissão de mensagens para celular pela internet, juntamente com a delegação de responsáveis para cada fluxo da fábrica, otimizou a comunicação por parte da gerência;
- Acompanhamento diário de cada fluxo pela gerência, possibilitando a previsão de riscos e a busca de soluções para possíveis atrasos na entrega de artefatos;
- Utilização do site e de serviços de *Concurrent Versions System* como mecanismos centralizadores de informações;
- Geração dos artefatos de requisitos, análise e projeto, contribuíram para a coerência exigida entre a equipe de desenvolvimento que se encontrava fisicamente distribuída;
- Importância da gerência na coordenação distribuída da equipe, garantindo o cumprimento dos prazos definidos com o cliente;
- Negociação de atrasos com o cliente baseado nos termos da SLA proposta, evitando assim ônus à fábrica;
- Realização de workshops e a distribuição de papéis em pares ajudou a fábrica a enfrentar as dificuldades encontradas durante o CADEX. Ausência de alguns membros por motivos de saúde e de trabalhos externos, não afetaram significativamente o andamento das atividades nem o prazo de entrega do produto final;
- Execução de um piloto contribuiu significativamente para a redução de tempo na produção de artefatos do projeto CADEX, devido à prática adquirida na instanciação dos templates do processo. Além disso, o piloto ajudou na disseminação do conhecimento entre os membros da equipe;

VIII. DIFICULDADES

Durante a experiência do projeto CADEX, foram percebidas algumas dificuldades:

- Atraso na entrega de artefatos devido a pouca disponibilidade e falta de comprometimento de determinados integrantes da equipe. Como solução, foi feita uma redistribuição das responsabilidades, baseada na familiaridade com o processo e comprometimento de cumprimento de prazos.
- Sobrecarga do gerente comercial nas atividades de atualização e inserção de novos *links* no *site*. Para tanto, foi feita uma distribuição dessa atividade entre os demais membros da equipe e um enfoque maior na política de revisão de artefatos por dois membros, antes de disponibilizá-los no *site*.
- Atraso nas respostas às solicitações por e-mail e a não utilização de ferramentas de Instant Message por alguns membros da fábrica, dificultaram a comunicação entre a equipe e ocasionaram atrasos na geração de artefatos. Como solução fez-se uso de envio de mensagens para celular, quando era desejada uma resposta mais imediata ou marcar reuniões urgentes via chat. Essa idéia também serviu para reduzir o custo alto no gerenciamento do projeto, pois, anteriormente, um contato mais urgente era estabelecido via comunicação por voz em celular;
- Dificuldade na implantação do Pair Programming, devido à falta de recursos e dificuldade de adaptação;
- Programação em cima de mesmas classes trouxe problemas durante a junção de diferentes versões, mesmo usando o CVS. Erros de usuários ocorriam durante a utilização do CVS e ocasionavam grandes perdas de tempo para serem corrigidos;
- Dificuldade na realização de reuniões semanais com a maioria da equipe, devido problemas como distância e conflito de horários;
- Dificuldade de acesso ao repositório de artefatos do sistema. Alguns membros da equipe trabalhavam muitas vezes atrás de firewalls o que impossibilitava seus acessos diretamente ao repositório. Como solução muitos dos artefatos foram enviados por e-mail para que um outro membro os colocasse no CVS;
- Diferença de versões de ferramentas causavam problemas de comunicação e geração de artefatos. Para isso foi utilizado um local — o site da fábrica — para concentrar todas as ferramentas nas versões que deveriam ser utilizadas por todos os membros da equipe;
- Falta de uma política de backup efetiva e a dependência do CVS do CódigoLivre mostrou-se um problema quando os serviços do mesmo pararam de funcionar. Basicamente uma semana foi perdida coletando as últimas versões de artefatos com membros da equipe e instalando um servidor CVS alternativo, o CVSNT, em uma máquina sobre o controle da fábrica.

IX. CONCLUSÃO

Pode-se concluir que a criação de uma fábrica de software, num curto período de tempo, com uma equipe parcialmente distribuída é completamente viável.

A definição de um processo eficiente e o uso de boas ferramentas de *groupware* são extremamente importantes para o perfeito funcionamento de uma fábrica de software e para a garantia da qualidade do produto final.

Com a utilização do processo no CADEX, as estimativas tornaram-se mais precisas, os produtos gerados apresentaram maior qualidade e a equipe de desenvolvimento tornou-se mais produtiva.

O processo de desenvolvimento iterativo e incremental permite uma maior compreensão do sistema bem como a detecção e resolução de problemas em tempo hábil. Contudo, um processo também é um conjunto de pessoas praticando atividades, assumindo responsabilidades e produzindo artefatos, e por isso não pode substituir o comprometimento e a motivação dos seus integrantes.

X. REFERÊNCIAS

[1] J. A. J. Brito. Metodologia para Gestão do Processo de Qualidade de Software para Incremento da Competitividade da Mobile. Disponível em: <http://www.mct.gov.br/Temas/info/Dsi/PBQP/Reuniao%20Petropolis/Apresentacao%20Mobile.pdf>. Acesso em 11/06/2003.

[2] I. Aaen, P. Bøtcher e L. Mathiassen. Software Factories. Disponível em: <http:// http://www.cin.ufpe.br/~in953/Software_Factories_17.pdf>. Acesso em 14/05/2003

[3] P. Krutchen. Introdução ao RUP - Rational Unified Process. Editora Ciência Moderna Ltda., 2003. ISBN 85-7393-275-9

[4] F. Maurer e S. Martel. Extreme Programming. Disponível em: <http://sern.ucalgary.ca/~milos/papers/2002/MaurerMartel2002d.pdf>. Acesso em: 10/05/2003.

[5] M. Kircher, P. Jain, A. Corsaro e D. Levine. Distributed Extreme Programming. Disponível em: <http://www.xp2001.org/xp2001/conference/papers/Chapter16-Kircher+alii.pdf>. Acesso em: 20/08/2003.

[6] A. A. de Alcântara, T. M. de M. Maciel, S. L. Meira e F. Q. B. da Silva. Uso do Processo RUP na implantação da ISO9000-3. Disponível em: <http://www.qualiti.com.br/artigos/artigo_WQS99.PDF>. Acesso em 11/06/2003.

[7] M. Hirsch. Making RUP Agile. Disponível em: <http://www.agilealliance.com/articles/articles/MakingRUPAgile.pdf>. Acesso em 19/08/2003.

[8] Guidelines for the Application of ISO 90001:1994 to the Development, Supply, Installation and Maintenance od Computer Software. ISO 9000-3: 1997(E). British Standard Institute.

[9] R. Sprenkels e A. Pras. Service Level Agreements. Ron Sprenkels, Aiko Pras. Disponível em: <http://citeseer.nj.nec.com/cache/papers/cs/24543/http:zSzzSzing.ctit.utwente.nlzSzWU2zSzd2.7zSzING-D2-7-SLAs.pdf/service-level-agreements.pdf>. Acesso em: 17/07/2003.

[10] Code Conventions for the JavaTM Programming Language. Disponível em: <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>. Acesso em 10/06/2003.

[11] Template de documentação. Disponível em: <http://www.geocities.com/factprocess>. Acesso em: 01/08/2003.

[12] J. P. Oliveira. Breve Introdução ao CVS. Disponível em: <gsd.di.uminho.pt/jpo/cadeiras/ 2002-2003/as2/cvsintro.pdf >. Acesso em: 15/06/2003.

[13] CódigoLivre. Disponível em: <http://www.codigolivre.org.br>. Acesso em 10/07/2003.

[14] B. Kurniawan. Java para a Web com Servlets, JSP e EJB. Rio de Janeiro: Editora Ciência Moderna Ltda., 2002.

[15] FreeMarker. Disponível em: <http://freemarker.sourceforge.net>. Acesso em: 02/07/2003.

[16] E. Gamma, R. Helm, R. Johnson e J. Vlissides. Padrões de projeto: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Boolkman, 2000.