

Fábricas de Software e o processo de desenvolvimento segundo a experiência da FábricaUm

H. M. Marques, *Engenheira de Software, UFPE*, I. G. L. Silva, *Engenheira de Requisitos, UFPE, R.*
T. Ramos, *Engenheiro de Software* and T. M. M. Maciel, *Gerente de Qualidade, C.E.S.A.R*

Resumo – A indústria de *software* vêm demonstrando crescente interesse pelas fábricas de *software*, organizações que provêm serviços de desenvolvimento de sistemas com alta qualidade, a baixo custo e de forma rápida, utilizando um processo de desenvolvimento de *software* bem definido e tecnologia de ponta, além de algumas formas de *feedback* para reconhecer e lidar com oportunidades de melhoria do processo. Tal interesse é decorrente da competitividade existente no mercado atual, no qual o equilíbrio ideal entre custo e qualidade dita o sucesso dos negócios. Este artigo tem por objetivo apresentar a experiência de criação de uma fábrica de *software* no ambiente acadêmico, descrevendo o processo concebido e as lições aprendidas.

Palavras-chave – desenvolvimento remoto, engenharia de *software*, fábrica de *software*, processo, programação extrema.

I. INTRODUÇÃO

Diante de um mercado exigente e extremamente dinâmico, no qual o equilíbrio ideal entre custo e qualidade dita o sucesso no mundo dos negócios, a produtividade passou a ser um aspecto fundamental para a sobrevivência de qualquer organização.

Particularmente no setor de *software*, as organizações vêm sofrendo transformações no sentido de se adequarem a um modelo que atenda às demandas de mercado. Neste cenário, muitos modelos de fábrica de *software* têm sido propostos e adotados [1,2,3]. Processos de *software* bem definidos e flexíveis, componentização, reuso de código e processos, representam estratégias bastante adotadas por estas organizações.

A terceirização de serviços de desenvolvimento de *software* (total ou parcial) no Brasil através de fábricas de *software* ainda é uma atividade recente. Contudo, os pesquisadores e empresas que trabalham na área de engenharia de *software* vêm atentando para o grande negócio que há por trás da manufatura de *software* em fábricas [1].

Inserido neste contexto, este artigo apresenta a experiência de construção de uma fábrica de *software*, a FábricaUm, criada como projeto de uma disciplina do mestrado em Ciência da Computação da Universidade Federal de Pernambuco. A fábrica é composta por onze membros, entre estudantes e profissionais de algumas empresas do ecossistema de tecnologia da informação de Pernambuco, que colaboraram

com suas pesquisas e vivências para este empreendimento.

Na próxima Seção, são apresentados conceitos de fábrica de *software*. A Seção 3 relata o estudo de caso da FábricaUm. Na Seção 4 as lições aprendidas durante a experiência são apresentadas. A Seção 5 apresenta as conclusões do trabalho, e as Seções 6 e 7 apresentam os agradecimentos e as referências bibliográficas, respectivamente.

II. FÁBRICAS DE SOFTWARE

Uma fábrica de *software* é uma organização que provê serviços de desenvolvimento de sistemas com alta qualidade, a baixo custo e de forma rápida, utilizando um processo de desenvolvimento de *software* bem definido e tecnologia de ponta, além de algumas formas de *feedback* para reconhecer e lidar com oportunidades de melhoria do processo [3].

Há mais de trinta anos a idéia de fábrica de *software* vem sendo continuamente moldada. As primeiras fábricas surgiram no final da década de 60 e ainda hoje o termo fábrica possui uma interpretação controversa quando o desenvolvimento de *software* é comparado à produção em massa de produtos industriais [2]. Ao invés disso, este termo deve ser interpretado como uma organização projetada de maneira particular e sistematizada, composta por pessoas envolvidas em um esforço comum, onde as tarefas são organizadas e a padronização é utilizada com o objetivo de auxiliar na coordenação e formalização do processo [2].

Neste contexto, as fábricas de *software* podem constituir núcleos de desenvolvimento dentro de grandes empresas, como é o caso das fábricas da Lucent Technologies [3], Politec e Softtek [1], ou serem empresas independentes. Além disso, o escopo do processo dentro do ciclo de desenvolvimento de *software* varia conforme a fábrica. Existem fábricas de *software* que prestam serviços a partir da análise de sistemas e outras que trabalham partindo da fase de implementação. Desta forma, as fábricas de *software* podem se tornar estruturas complementares à organização do cliente, ampliando de forma eficaz e qualificada a capacidade de atendimento à demanda de serviços de *software*.

Fábricas localizadas na Índia exportam muitos serviços de programação de aplicações para os EUA e países da Europa, cujos mercados de tecnologia da informação são os maiores do mundo. Entretanto, grandes investimentos na criação de fábricas de *software* vêm sendo realizados no Brasil, China e Rússia [1].

Em [1] são listadas cinco razões para o crescimento das fábricas de *software* no Brasil:

- O Brasil se tornou uma opção interessante para exportação de programação devido à desvalorização cambial em relação aos EUA e ao baixo custo de homem/hora;
- Grande parte dos trabalhos de fábrica de *software* é decorrente de revisão de processos ou projetos de integração como, por exemplo, serviços de customização de produtos produzidos por multinacionais para o mercado brasileiro;
- As arquiteturas de sistemas são projetadas fragmentadas em camadas, o que possibilita o desenvolvimento remoto de partes desses fragmentos;
- O crescimento de fábricas que possuem conhecimentos de negócios e prestam serviços de análise de sistemas, e não apenas implementação;
- A tendência existente de terceirização de serviços por parte de empresas que se concentram em suas atividades principais e transferem para parceiros as atividades que não estão diretamente ligadas ao seu negócio principal.

Dois grandes desafios da engenharia de *software* na atualidade que estão relacionados à prestação de serviços de desenvolvimento de *software* são a gerência de projetos desenvolvidos remotamente e a boa definição de um processo de desenvolvimento.

A. Desenvolvimento remoto

Segundo Ambler [4], existem duas categorias de desenvolvimento remoto de *software*: o desenvolvimento distribuído e o desenvolvimento disperso. O primeiro consiste no trabalho de times remotamente localizados, compondo uma equipe maior, enquanto que no segundo cada desenvolvedor que faz parte da equipe trabalha sozinho e está remotamente localizado. Essas duas categorias podem existir em fábricas de *software*, uma vez que se uma empresa terceiriza seus serviços para determinada fábrica esse desenvolvimento torna-se distribuído. Ainda assim, os colaboradores da desta fábrica podem desenvolver o *software* de forma dispersa ou distribuída.

Existem algumas abordagens para a utilização de desenvolvimento distribuído, entre elas estão a OSS (*Open Source Software*) [5,6,7] e DXP (*Dispersed Extreme Programming*) [8].

As comunidades de OSS podem ser analisadas como organizações virtuais, considerando-se os seguintes aspectos [9]: i) seus membros compartilham um interesse comum com relação ao *software* sendo desenvolvido; ii) em geral, são geograficamente distribuídas e usam a Internet como ferramenta principal para coordenar as ações; e iii) usam tecnologias de comunicação e informação para gerenciar interdependências, tais como sistemas de controle de versão, listas de *e-mails*, etc.

A DXP aplica os valores e princípios da *Extreme Programming* (XP) [10], adaptando suas práticas para o ambiente de desenvolvimento distribuído de *software*. Alguns

requisitos são essenciais para esta abordagem [8]:

- Conectividade entre os membros;
- Gerência de configuração;
- Familiaridade entre os membros;
- Motivação dos membros do time;
- Tolerância a problemas relacionados à falhas de comunicação, tais como desconexão, etc.

B. Processos de desenvolvimento

Nos últimos anos, os processos de desenvolvimento de *software* vêm recebendo grande atenção principalmente no que diz respeito a sua definição, automação, medição e melhoramento. De acordo com Pressman [11], no contexto da engenharia de *software*, um processo é um *framework* para as tarefas necessárias à construção de *software* com alta qualidade, definindo a abordagem que será adotada enquanto o *software* estiver em desenvolvimento.

A qualidade dos produtos de *software* está diretamente relacionada à qualidade do seu processo de desenvolvimento, desta forma, é importante que o processo de desenvolvimento de *software* de uma fábrica seja de alta qualidade e vise melhorias contínuas [12]. Processos de desenvolvimento bem-definidos devem possuir as seguintes características [13]:

- Abrangência: deve definir todo o ciclo de vida. Isto inclui todas as macro-atividades, documentos internos e externos, padrões relacionados e restrições;
- Profundidade: deve definir aspectos do processo em diferentes níveis de abstração. Isto inclui todas as conexões entre atividades, fases e produtos de trabalho;
- Flexibilidade: deve estar apto a descrever atividades contínuas (por exemplo, a definição de requisitos) e atividades pontuais, tal como inspeções;
- Praticidade: deve garantir a adaptação a diferentes tipos de projetos;
- Facilidade de medição: deve facilitar a gerência e visibilidade do desempenho do mesmo através de métricas de processo;
- Ser passível de auditoria: deve ser específico e concreto suficiente para que um agente independente tenha o mesmo julgamento sobre o uso do processo que está sendo seguido;
- Aptidão ao envolvimento: deve incluir a provisão para solicitação de mudanças.

III. ESTUDO DE CASO – A FÁBRICAUM

O conceito de fábrica de *software* está baseado na idéia de manufatura de *softwares* que provêm soluções para atender às necessidades específicas de cada cliente. O sucesso do projeto, e a conseqüente satisfação do cliente devem ser garantidos através da utilização de um processo de desenvolvimento que facilite o apoio administrativo da empresa, a interação com o cliente, a utilização de boas práticas de gerenciamento de projetos e um bom levantamento dos objetivos do sistema [14]. As organizações devem ser capazes de construir *software* de qualidade, seguindo um ciclo

de desenvolvimento rápido e eficiente, a um baixo custo.

Diante deste cenário, a FábricaUm foi criada com o objetivo de oferecer serviços de qualidade, a um baixo custo focando em produtividade. Os princípios básicos que guiaram tal organização foram:

- Simplicidade;
- Produtividade;
- Foco no processo de desenvolvimento de *software*;
- Uso de técnicas estabelecidas como padrão no mercado.

Com base nestes princípios, o primeiro passo foi desenvolver um processo padronizado de desenvolvimento de *software*.

O processo da FábricaUm foi desenvolvido ao longo de cinco meses. Durante as primeiras quatro semanas, aconteceu a modelagem do processo em si, incluindo a definição dos fluxos de trabalho e suas atividades. Nas seis semanas seguintes, foi executado o projeto piloto com o objetivo de avaliar e coletar o *feedback* em relação ao processo definido, a fim de identificar possíveis ajustes. Estes ajustes foram implementados a longo de uma semana. Por fim, com o objetivo de consolidar os ajustes realizados, um novo projeto, com duração de sete semanas, foi implementado pela FábricaUm seguindo o processo de desenvolvimento já ajustado.

A. Modelagem do processo da FábricaUm

O *ProcessOne*, como foi denominado o processo criado pela FábricaUm, foi definido de maneira que pudesse ser instanciado de acordo com as necessidades específicas de cada projeto.

Para a escolha de metodologias adequadas, primeiramente foram analisados alguns processos ágeis, como o Scrum, Crystal e XP [15, 16]. Foram observadas a sua integração com a gerência de negócios, adaptação em pequenas equipes, produtividade e garantia de qualidade. Todas estas características apresentaram problemas em relação à análise de custos do projeto, por se basearem numa metodologia adaptativa, na qual o projeto evolui gradualmente e prazos e custos são definidos através da experiência anterior da equipe.

Devido à necessidade de um processo preditivo, onde os custos e prazos são definidos no início do projeto, foi decidido utilizar uma instância leve (processo simplificado em relação ao número de papéis, atividades e artefatos) do Rational Unified Process (RUP) [17, 18] como processo de desenvolvimento de *software* para a FábricaUm.

O RUP oferece um *framework* de processos centralizado na arquitetura, que visa inicialmente os riscos de projeto, e é baseado em boas práticas de desenvolvimento. Ele foi escolhido por ser maduro, amplamente difundido e utilizado, e por atender todas as necessidades da empresa desde a fase de planejamento até a fase de implantação.

Algumas práticas e lições aprendidas no XP [10], como código coletivo, testes unitários, cliente *on-site* e *pair programming* foram adicionadas como forma de agilizar o andamento o processo. Além disso, foram incorporados ao

ProcessOne aspectos de desenvolvimento de *software* remoto, devido à natureza da equipe envolvida, formada por pessoas que trabalham geograficamente separadas.

A fim de resolver as quatro principais problemas existentes em times distribuídos [19, 20, 21] (visão geral do projeto, integração entre as equipes, independência e comunicação), foram adotadas algumas técnicas definidas por Wills e Ambler [4, 8]:

- Realização de tarefas distribuídas somente quando necessário;
- A equipe trabalhando inicialmente unida para criar um maior entrosamento e ter conhecimento dos processos e métodos utilizados pela empresa;
- A equipe estando, sempre que possível, disponível para discussões, independente de seu real horário de trabalho;
- Utilização de ferramentas para facilitar a comunicação como *instant messaging* e *e-mail*, e de padrões para codificação e modelagem [22, 23];
- Quando necessário, a utilização da prática de *long-distance pair programming*, utilizando ferramentas como Netmeeting e Vnc.

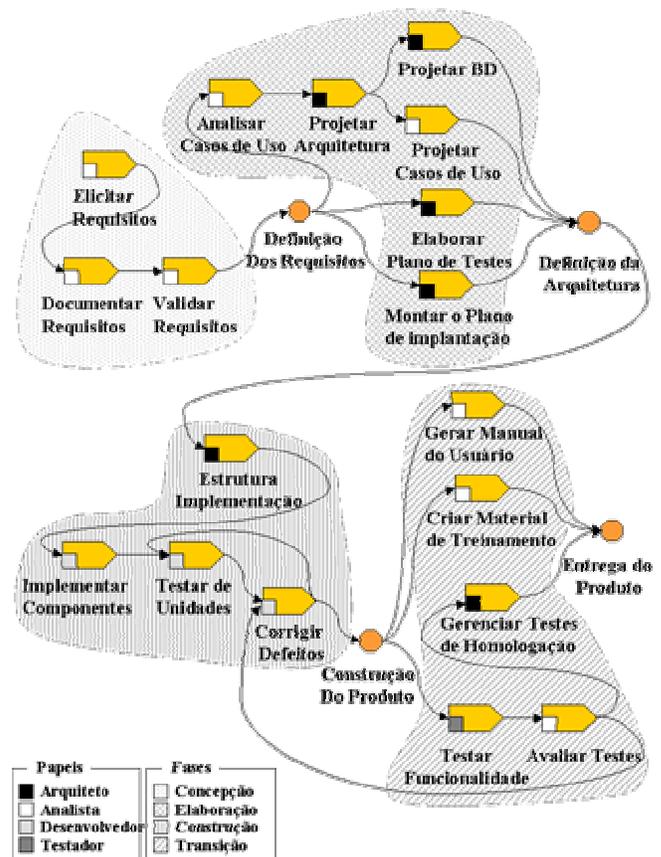


Fig. 1. Visão Consolidada do *ProcessOne*

O *ProcessOne* considera os seguintes fluxos de atividades: Planejamento e Acompanhamento, Gerenciamento de Qualidade, Gerenciamento Comercial, Gerenciamento de Requisitos, Gerenciamento de Configuração, Análise e Projeto, Implementação, Testes e Implantação.

A Fig.1 apresenta uma visão consolidada dos fluxos de

atividades acima mencionados com as fases do RUP e os marcos associados às mesmas, enfocando os fluxos de requisitos, análise e projeto, implementação, testes e implantação. Os outros fluxos (comercial, planejamento, configuração e qualidade) não foram incluídos nesta figura por serem fluxos de apoio, e estarem dissolvidos por todas as fases.

Para cada atividade do processo foram definidos os artefatos de saída, indicando se estes são obrigatórios ou não, assim como os papéis responsáveis por executar tal atividade. Por exemplo, a atividade ‘Analisar Casos de Uso’ gera como artefato de saída o modelo de classes de análise. Esta atividade é classificada como ‘Opcional’ e deve ser executada pelo Analista.

Considerando uma visão ortogonal do processo, cada fase do RUP é marcada por características-chave que devem ser mapeadas ao longo da execução das atividades abrangidas por cada uma destas fases. A Tabela I ilustra as características-chave das fases do RUP.

TABELA I
CARACTERÍSTICAS-CHAVE DAS FASES DO RUP

| Concepção | Elaboração | Construção | Transição |
|--|---|---------------------------|----------------|
| Foco nos objetivos, Princípios e Metas | Interação com o Cliente | Definição da Arquitetura | Builds |
| Alocação de Papéis | Documentação Simples e Completa | Foco em projeto | Foco em Testes |
| Divisão de Atividades | Estruturação do Ambiente de Desenvolvimento | Implementação Distribuída | Implantação |

Na fase de Concepção é definido e validado todo o escopo e as reais necessidades do projeto. Nesta fase não é feito nenhum tipo de trabalho distribuído, o trabalho é realizado em conjunto e a equipe deve estar bastante entrosada com o cliente, pois esta fase é de extrema importância para o sucesso de todo o projeto. A participação do cliente é essencial nesta fase.

Na fase de Elaboração, o principal objetivo é deixar o projeto arquiteturalmente pronto para que se possa iniciar o seu desenvolvimento. Com base nos requisitos especificados na fase anterior inicia-se o processo da análise dos casos de uso paralelamente à elaboração dos planos de testes e de implantação.

Durante a fase de Construção são construídos os componentes de *software* que farão parte do produto final, levando em conta a arquitetura previamente definida. O fluxo que recebe maior destaque nesta fase é o de Implementação, no qual é criado um ambiente único de desenvolvimento e é estruturado um *framework* que serve de guia para os desenvolvedores. A partir daí, a implementação do sistema acontece de forma distribuída, quando pequenos módulos do sistema, com escopos e interfaces bem definidas, são entregues aos desenvolvedores.

Na fase de Transição, são realizados os testes de funcionalidade, conforme previstos na fase de Elaboração, e

de homologação. Caso o resultado dos testes aponte que um requisito não foi implementado corretamente, uma CR (requisição de mudanças) deve ser aberta para que um desenvolvedor possa alterá-la. Devido ao aspecto distribuído do fluxo de implementação, foi adotada a ferramenta Web Bugs [24] para o cadastro de falhas e requisição e de mudança.

B. Estratégias de Desenvolvimento

Uma vez definido um processo formal, foi desenvolvido um projeto piloto como meio de obter *feedback* em relação ao processo definido, o *ProcessOne*. A partir da experiência com o projeto piloto, os resultados mostraram a necessidade de uma reformulação do processo. Em consequência disto, foram implementadas ações para diminuir o *overhead* de atividades no processo como, por exemplo, definir algumas atividades como opcionais.

Durante os projetos realizados utilizando-se o *ProcessOne*, todos os fluxos foram seguidos de forma distribuída. Contudo, as equipes que ficaram responsáveis pelo fluxo de Implementação trabalharam de forma dispersa, com uma pessoa da equipe responsável por gerar *builds* periodicamente e garantir a coerência entre as partes produzidas.

Na próxima seção serão apresentados os resultados e lições aprendidas obtidos a partir da experiência vivenciada pela FábricaUm com a implantação do *ProcessOne* em seus projetos.

IV. LIÇÕES APRENDIDAS

Diversas lições de sucesso e insucesso puderam ser extraídas a partir da experiência prática dos membros da FábricaUm em criar uma fábrica de *software* e desenvolver projetos caracterizados por: desenvolvimento remoto, código aberto limitado aos integrantes da equipe (*Royalty Free Bin*) [25, 26] e esforço médio semanal instável.

Ao longo do desenvolvimento, algumas métricas foram coletadas no âmbito gerencial e tecnológico. As métricas gerenciais foram concentradas praticamente no esforço da equipe de desenvolvimento para análise de efetividade do planejamento, além de suporte para tomada de decisões referentes a prazos. Ao todo, 487 horas foram gastas no desenvolvimento, com uma estimativa de 484,2 horas.

As métricas de implementação coletadas incluem:

- Medidas quantitativas (pacotes, classes e métodos);
- Dependências entre classes (acoplamento, inclusive cíclico);
- Documentação por classe/método;
- NCSS (*Non Comment Source Statement* - linhas de código reais, sem contar linhas em branco e comentários);
- CCN (complexidade ciclomática - possíveis caminhos a partir de desvios condicionais encontrados no código).

Durante todo o desenvolvimento do processo adotado, foram apresentadas sugestões e críticas pelos integrantes da equipe nas reuniões de projeto. Neste contexto, lições aprendidas foram identificadas, as quais podem ser categorizadas em lições referentes a: ferramentas de apoio;

comunicação; tecnologia e processo de *software*; e, planejamento e gerência do projeto.

A. Ferramentas de Apoio

As ferramentas automatizam as tarefas garantindo uma maior produtividade, além de dar suporte a fatores críticos de sucesso como comunicação com o cliente. Neste contexto, algumas lições aprendidas pode ser ressaltadas:

- Utilizar o controle de versão desde o início dos projetos facilitou bastante o funcionamento da fábrica como um todo. Este controle possibilitou uma melhor organização dos documentos, atividades distribuídas, além da própria comunicação interna;
- O uso *instante messaging* (MSN, ICQ) facilitou a comunicação interna da equipe, tendo em vista a dificuldade de conciliar as disponibilidades de horário dos seus onze membros. Esta ação possibilitou a realização de reuniões virtuais;
- A utilização de ferramentas com apoio a desenvolvimento de OSS facilitou o gerenciamento do desenvolvimento remoto. Foram utilizadas ferramentas disponibilizadas pelo site Código Livre [24], como controle de bugs e tarefas;
- A utilização de *frameworks* e bibliotecas abertas facilitou a reutilização de componentes, ajudou a aumentar a produtividade da equipe e diminuiu os custos associados aos projetos.

B. Comunicação

Como o desenvolvimento dos projetos foi realizado de forma remota, a realização de reuniões constantes e objetivas foi fundamental para garantir a sinergia da equipe. Algumas boas práticas foram relevantes neste contexto, conforme a seguir.

- Treinamentos de todos os integrantes sobre o processo, bem como sobre tecnologias padrões utilizadas pela fábrica ajudaram a criar uma comunicação uniforme na equipe, a coordenar suas atividades e aumentaram a motivação de seus integrantes;
- A transparência da maioria das ações executadas durante o projeto aumentou o entrosamento da equipe e serviu como motivação após grandes marcos ocorridos nos projetos;
- A realização de reuniões semanais de nivelamento, virtuais e/ou presenciais, permitiu o acompanhamento por toda a equipe sobre o andamento do projeto, isto foi essencial para manter a integridade das atividades e comprometimento de cada um. As reuniões sempre que possível eram presenciais para fortalecer a sinergia do grupo, porém, quando não havia disponibilidade as reuniões virtuais via MSN eram adotadas;
- A publicação de um site do projeto (www.fabricaum.kit.net) facilitou a comunicação com o cliente e entre os próprios desenvolvedores. Especialmente para equipes distribuídas, a disponibilidade de informações sobre o projeto na *Web* tornou-se uma excelente prática para comunicação com o cliente, acompanhamento do

projeto e interação da equipe. O *site* disponibilizou informações gerais, artefatos disponíveis, progresso do desenvolvimento e metas dos projetos, informações sobre a equipe, etc;

- A discussão do processo também pôde ser realizada por todos a partir de uma lista de *e-mails*, ajudando também na transparência de todas as ações executadas pelos membros da FábricaUm.

C. Tecnologia e Processo de Desenvolvimento

O processo de desenvolvimento foi definido antes de serem iniciadas as atividades de desenvolvimento do projeto piloto. A partir de então, o compromisso com a melhoria contínua do processo precisou ser constante e efetivo. As seguintes lições foram identificadas, neste contexto:

- A avaliação contínua do processo ao longo do desenvolvimento e a partir da colaboração de todos foi extremamente válida. Em todas as reuniões foram levantados pontos de melhoria, os quais foram agregados ao processo quando apropriado;
- Definir o processo com antecedência, incluindo procedimentos, guias e *templates* para apoio às atividades, facilitou bastante a execução das atividades e geração de artefatos;
- Pelo processo ter sido definido de forma distribuída, inconsistências entre fluxos surgiram. A existência de um profissional dedicado ao processo facilitou substancialmente o controle destas inconsistências;
- Houve constantemente a preocupação de que módulos criados pudessem ser reutilizados futuramente. No nosso caso de estudo, durante a construção do segundo projeto reusamos módulos implementados no piloto;
- A utilização de desenvolvimento incremental (simples e sob demanda) foi fundamental para o atendimento de prazos.

D. Planejamento e Gerência do Projeto

As lições obtidas em relação ao planejamento e gerência dos projetos foram as seguintes:

- A existência de um profissional exclusivo para gerência do projeto contribuiu para que o desenvolvimento fluísse da melhor forma possível. Inicialmente foi criado um grupo de gerência, mas todos os componentes estavam envolvidos em outras tarefas, o que prejudicou bastante o acompanhamento. Em um segundo momento um integrante assumiu totalmente a função facilitando a comunicação e o atendimento de metas e prazos.
- Apesar de revisões e validações dos artefatos gerados demandarem tempo, e nem sempre serem estimados, é fundamental a revisão técnica dos artefatos a serem entregues antes de chegarem às mãos do cliente.
- Para estimativa de esforços do projeto, foi adotada a técnica de pontos de caso de uso, utilizando uma média semanal de esforço disponível de cada desenvolvedor. Diante a instabilidade desta média, ou seja, em algumas

semanas a dedicação poderia ser maior do que em outras, tornou-se difícil acompanhar as estimativas de esforço pelo cronograma, baseado em esforço por tarefa e prazos estabelecidos. Neste contexto, a prática de XP para planejamento/acompanhamento baseado em semanas ideais e velocidade foi identificada pelo grupo como mais indicada e será adotada pela equipe nos próximos desenvolvimentos.

V. CONCLUSÃO

Este trabalho apresentou a experiência da FábricaUm, enquanto fábrica de *software*, na definição de um processo de desenvolvimento e na execução deste processo em seus projetos.

A FábricaUm foi criada como uma experiência acadêmica, e nela pudemos integrar de forma ampla conhecimentos teóricos sobre engenharia de software e nossas vivências no mercado de trabalho, aplicando-os à construção do *ProcessOne*. É importante ressaltar que o envolvimento de todos os participantes com a melhoria e consistência do processo de desenvolvimento foi fundamental, colaborando para o aumento da qualidade de nosso processo de desenvolvimento.

Verificamos através desta experiência que para que o processo adotado em uma fábrica de *software* de pequeno porte alcance produtividade e qualidade, deve incorporar características encontradas em metodologias ágeis [10, 15, 18], como minimização e simplificação das tarefas e artefatos, e grande comunicação entre a equipe e o cliente. Apesar disso, características encontradas em processos mais maduros como o RUP [17, 18] não podem ser ignorados, como o auxílio ao processo de venda do produto/serviço, a garantia da qualidade, e a redução dos riscos no início do projeto através da definição e uso de uma arquitetura estável.

Consideramos que o ganho de conhecimento neste experimento foi superior ao estimado, e a aplicação do *ProcessOne* em nossos projetos muito satisfatória. Contudo, ainda há muito trabalho a ser feito para que a FábricaUm atinja um elevado nível de maturidade organizacional, principalmente no que diz respeito ao gerenciamento de desenvolvimento remoto, com produtividade e qualidade aceitável pelo mercado.

VI. AGRADECIMENTOS

Os autores agradecem a contribuição dos demais participantes da FábricaUm (Celso Rosa, Elson Melo, Fábio Buchmann, João Bosco, Júlio César, Rafael Marques e Silvio Cortez) por toda sua dedicação durante o desenvolvimento deste, bem como as contribuições de Jones Albuquerque e Silvio Meira para o sucesso desse empreendimento.

VII. REFERENCES

- [1] *Fábrica de software: uma vocação nacional?* Disponível em: <http://computerworld.terra.com.br/AdPortalV3/adCmsDocumentoShow.aspx?documento=24655&Area=1>

- [2] Aaen, I., Böttcher, P. & Mathiassen, L. *The Software Factories: contributions and Illusions*. In: Proceedings of the Twentieth Information Systems Research Seminar in Scandinavia, Oslo, 1997.
- [3] *Making the Software Factory Work: Lessons from a Decade of Experience*. Disponível em: <http://mockus.us/papers/factory.pdf>
- [4] Ambler, S. W. *Bridging the Distance*. Disponível em: <http://www.sdmagazine.com/documents/s=826/sdm0209i/>
- [5] Schach, S.R. and Offutt, A. J. *On the Nonmaintainability of Open-Source Software*. Proceedings of the 2nd Workshop on Open Source Software Engineering, Orlando, FL, May 2002.
- [6] Hecker, F. *Setting Up Shop: The Business of Open-Source Software*. Disponível em: <http://www.hecker.org/writings/setting-up-shop.html>
- [7] *SourceForge.net*. Disponível em: <http://sourceforge.net/>
- [8] Wills, A. C. *Dispersed Agile Software Development and Dispersed eXtreme Programming*. Disponível em: <http://www.fastnloose.org/cgi-bin/wiki.pl/dad/?algheroworkshop>
- [9] Crowston, K. and Scozzi, B. *Open source software projects as virtual organizations: Competency rallying for software development*. IEE Proceedings Software, 2002, 149(1), pp. 3-17.
- [10] *Extreme Programming: A Gentle Introduction*. Disponível em: <http://www.extremeprogramming.org/>
- [11] Pressman, R. *Software Engineering: A Practioner's Approach*. McGraw-Hill, 2000.
- [12] Carvalho, A. E. S., Tavares, H. C. A. B., Castro, J. F. B. *Uma Estratégia para Implantação de uma Gerência de Requisitos visando a Melhoria dos Processos de Software*. WER2001 - IV Workshop on Requirements Engineering, 2001, Buenos Aires.
- [13] Maciel, T. M. M. *Definição e Melhoria de Processo de Software*. Disponível em: http://www.uflatec.com.br/spin/PQ2003/Lavras/Teresa/ProQuality_Apresentacao_Teresa.zip
- [14] *Micro Projects Cause Constant Change*. Disponível em: <http://www.agilealliance.com/articles/articles/Chapter30-Johnson.pdf>
- [15] *The AgileAlliance*. Disponível em: <http://www.agilealliance.com>
- [16] Fowler, M. *The New Methodology*. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html>
- [17] Rational Software Corporation, RUP - Rational Unified Process, Version 2002.05.00. Disponível em: <http://www.rational.com/products/rup/index.jsp>
- [18] *A Comparison of RUP® and XP*. Disponível em: <http://www.rational.com/media/whitepapers/TP167.pdf>
- [19] Starrett, E. *Which Distributed Development Concept Did You Have in Mind?* Disponível em: <http://www.stsc.hill.af.mil/crosstalk/2001/11/publisher.html>
- [20] Amako, K. *Distributed Software Development in GEANT4 Project*. Disponível em: <http://www.wasd.web.cern.ch/wwwasd/geant/documents/lhcPD96/>
- [21] Guck, R. *Managing Distributed Software Development*. Disponível em: <http://www.stickyminds.com/sitewide.asp?ObjectId=6002&Function=DDETAILBROWSE&ObjectType=ART>
- [22] Sun Microsystems, Code Conventions for the Java™ Programming Language. Disponível em: <http://java.sun.com/docs/codeconv/html/CodeConvTOC.doc.html>
- [23] Ronin International, Coding style. Disponível em: www.codingstyle.info
- [24] Código Livre. Disponível em <http://www.codigolivre.org.br>
- [25] The Free Software Foundation (FSF), <http://www.fsf.org/>
- [26] Free/Libre and Open Source Software Classification. Disponível em: <http://www.infonomics.nl/FLOSS/index.htm>