Ensino da Engenharia de Software por meio de Fábricas de Software no contexto Distribuído: Um Relato de Experiência

Catarina Costa[1], Rodrigo Rocha[1], Jair Figueirêdo[1], Marcos Duarte[1], Silvio Meira[1], Rafael Prikladnicki[2]

[1]Centro de Informática – Universidade Federal de Pernambuco (UFPE)
50.732-970 – Recife – PE – Brasil

[2]Pontifícia Universidade Católica do Rio Grande do Sul – PUCRS – FACIN
90619-900 – Rio Grande do Sul – RS – Brasil

{csc, rgcr, jjcf, mpd, srlm}@cin.ufpe.br, rafaelp@pucrs.br

Abstract. *Globalization has enabled the development of software evolved significantly, creating the Distributed Software Development. Given this development environment, it is necessary to hire professionals who can perform well development activities in this scenario. Therefore, the learning centers need to create a motivating environment in which to prepare professionals for the market. This paper aims to report the experience of students in learning the concepts of software engineering, using a pedagogical tool a small software factory through of a real project and working in a distributed environment.*

Resumo. *A globalização permitiu que o desenvolvimento de software evoluísse significativamente, criando o Desenvolvimento Distribuído de Software. Diante deste ambiente de desenvolvimento, faz-se necessário a contratação de profissionais que possam desempenhar bem as atividades de desenvolvimento neste cenário. Dessa forma, os centros de ensino precisam criar um ambiente motivador, no qual seja possível preparar profissionais para o mercado. O objetivo nesse artigo é relatar a experiência de alunos no aprendizado dos conceitos de engenharia de software, utilizando como ferramenta pedagógica uma pequena fábrica de software com um projeto real trabalhando de maneira distribuída.*

1. Introdução

A evolução do software faz com que diversas áreas do conhecimento reconheçam sua importância como posição estratégica perante o mercado. Com esse reconhecimento, a demanda por software aumenta, transformando mercados nacionais em mercados globais. Essas transformações alteraram a forma como os produtos são concebidos, construídos, testados e entregues aos clientes finais. Desta maneira, o software tem se tornado um componente estratégico para diversas áreas de negócio, mais especificamente na área de Engenharia de Software (ES), criando novas formas de cooperação e competição que vão além das fronteiras dos países [Herbsleb e Moitra, 2001]. É possível verificar essas mudanças a partir do final da década passada, no qual, visando diminuir os custos e buscando recursos mais qualificados, muitas

organizações começaram a experimentar o Desenvolvimento Distribuído de Software (DDS) [Herbsleb e Moitra, 2001].

O ambiente distribuído herdou os desafios do desenvolvimento tradicional (co-localizado) de software e acrescentou outros desafios ao processo ao adicionar fatores como dispersão física, distância temporal e diferenças culturais. Diante deste complexo ambiente de desenvolvimento, faz-se necessário cada vez mais a contratação de profissionais que possam planejar, analisar e especificar requisitos, projetar arquitetura de sistemas e coordenar todas estas atividades em meio as adversidades trazidas no novo cenário de desenvolvimento global. Em meio à crescente demanda, os centros de ensino que preparam profissionais para o mercado, precisam de ferramentas pedagógicas que possam criar um ambiente motivador, no qual seja possível aplicar e preparar os alunos para solucionar os problemas demandados pelo mercado.

Neste contexto, este trabalho tem como principal objetivo relatar a experiência de uma equipe de alunos de pós-graduação em Ciência da Computação da Universidade Federal de Pernambuco no aprendizado dos conceitos de engenharia de software, utilizando como ferramenta pedagógica de aprendizado pequenas fábricas de software [Siy, 2001] com a missão de solucionar um problema real trabalhando de maneira distribuída. Desta forma, os alunos foram divididos de acordo com o perfil de conhecimento e lhes foi atribuída a tarefa de organização completa da fábrica: definição de papéis, hierarquia, processo de desenvolvimento, seleção de ferramentas de gerencia, entre outros.

O trabalho está organizado da seguinte maneira: na seção 2 é apresentado o contexto da fábrica de software criada para a disciplina de Engenharia de Software; na seção 3 é apresentado o projeto desenvolvido, o papéis, processo e ferramentas, assim como os desafios e adaptações realizadas; na seção 4 uma reflexão sobre a aprendizagem no uso de fábrica de software para o ensino da engenharia de software é apresentada; e por fim, na seção 5 são apresentadas as considerações finais.

2. A Fábrica de Software: TechnoSapiens

O estudo de caso é parte de uma disciplina de Engenharia de Software (IN953, 2008) com foco na criação de fábricas de software que façam uso de desenvolvimento distribuído para a realização de projetos reais. Com a definição dos clientes e projetos, os alunos têm quatro meses para executar um ambiente de fábrica de software e entregar os produtos definidos em comum acordo com o cliente. A TechnoSapiens foi uma das fábricas criadas durante a disciplina e foi composta por nove estudantes, todos trabalhando de forma distribuída. Estavam envolvidos no projeto membros distribuídos nas cidades de Recife – PE, João Pessoa – PB e Campina Grande – PB. Nenhum dos integrantes havia trabalhado anteriormente com outro membro da equipe ou de forma distribuída.

A fim de permitir adequações e melhorias na fábrica, um projeto piloto foi executado no primeiro mês da disciplina. O objetivo foi validar a estrutura organizacional da fábrica e do processo de desenvolvimento adotado. Após o primeiro projeto, um segundo projeto de dimensões maiores foi executado num

período de três meses. O projeto assumido pela fábrica foi o desenvolvimento de uma versão para dispositivos móveis do serviço citIX[1]. O citIX utiliza um sistema de informações geográficas que possibilita que os usuários do serviço construam uma rede de conhecimento sobre as características (segurança pública, entretenimento, infraestrutura, serviços públicos, entre outros.) de uma determinada localidade ou região.

3. O Projeto CitIX

O projeto CitIX é um projeto liderado pelo Centro de Estudos Avançados do Recife – CESAR, seu principio é montar uma rede social sobre uma plataforma de sistemas de coordenadas geográficas, produzindo conhecimento categorizado em determinada região. Uma versão Web do projeto já havia sido construída, cabendo a TechnoSapiens construir uma versão para dispositivos móveis.

Como princípios base do desenvolvimento do projeto buscou-se utilizar as mais adequadas e eficientes técnicas, práticas, métodos e ferramentas de engenharia de software, tendo em vista as peculiaridades de um projeto distribuído. Embora o projeto fosse acadêmico havia um cliente real e um prazo determinado a ser cumprido, demandando muito empenho e responsabilidade por parte do time.

Logo após a definição do projeto a ser desenvolvido, o time precisou tomar diversas decisões, tais como: os papéis dentro da fábrica, o processo a ser seguido, as ferramentas a serem utilizadas, as formas de comunicação, entre outras. Para direcionar os trabalhos da equipe, o primeiro passo foi definir um plano de projeto, com todas as informações necessárias, para que o trabalho fosse conduzido da melhor forma. O plano de projeto continha uma visão geral do projeto: objetivos, escopo, os papéis e responsabilidades bem definidos, as entregas do projeto, os riscos identificados até aquele momento, um plano de comunicação entre o time e o cliente, com a periodicidade de reuniões e os artefatos a serem produzidos, os critérios de aceitação, os treinamentos necessários a equipe, e finalmente, o cronograma do projeto.

Um gerente e um vice-gerente foram eleitos de acordo o perfil identificado pelo time para o planejamento e acompanhamento dos trabalhos. Eles foram responsáveis por definir um plano de projeto e o cronograma de atividades, assim como um plano de comunicação e os meios de acompanhamento dos trabalhos. Um SQA (*Sofware Quality Assurance*) ficou responsável por acompanhar a evolução e realizar as adaptações durante o andamento do projeto. Também visando garantir a qualidade, um membro ficou alocado para o planejamento e realização de testes do projeto e reportar aos demais membros e principalmente a gerência as necessidades e impedimentos para a realização dos mesmos.

Para garantir um adequado ambiente de desenvolvimento, bem como uma gestão de configuração eficiente, um líder técnico com a ajuda de outros membros da fábrica ficou responsável pelas questões técnicas do projeto. Com os principais papéis definidos, todos os demais membros, e até mesmo os que já tinham assumido

[1] citIX: http://www.citix.net/

algum dos papéis já citados, eram engenheiros de software, papel que exercia diversas atividades no processo (especificação dos requisitos, modelagem de dados, implementação, atualização do site de acompanhamento do projeto, entre outros).

O processo utilizado para conduzir o projeto, chamado de TechnoProcess (Figura 1), foi baseado no *Rational Unified Process* (RUP) [Cavalcanti, Bandeira e Donegan, 2004], que por ser muito amplo, foi adaptado para uma versão mais leve, reduzindo-se o número de processos e artefatos de acordo com a realidade do projeto. Apesar das muitas aplicações do RUP no TechnoProcess, a equipe sentiu a necessidade de adicionar a ele algumas práticas ágeis que se tem verificado em projetos de sucesso. Com isso, foram incorporadas à metodologia escolhida algumas práticas e lições aprendidas no XP (*eXtreme Programming*), como por exemplo: revisão permanente do código, programação em par, integração contínua e frequente comunicação.

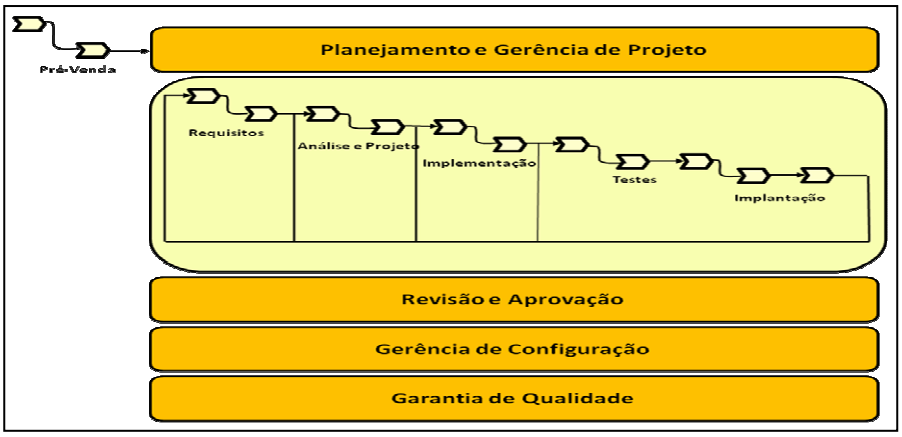

Figura 1. TechnoProcess

As principais fases definidas para o processo foram: Pré-Venda, Requisitos, Análise e Projeto, Implementação, Testes e Implantação. Fases de suporte também foram definidas: Planejamento e Gerência de Projeto, Revisão e Aprovação, Gerência de Configuração e Garantia de Qualidade.

Em paralelo a definição dos papéis e processo, estudavam-se as ferramentas de apoio ao projeto. A ferramenta *web* XPlanner de planejamento e acompanhamento foi utilizada para que, mesmo com a distribuição, a equipe tivesse visibilidade do projeto e pudesse acompanhar o andamento das atividades e saber, além das atividades onde cada membro estava alocado, o tempo dedicado e a estimativa de término, gerenciando assim as dependências, além da possibilidade de utilização de métricas de esforço e controle do projeto. Outras ferramentas foram fundamentais para a comunicação entre os membros, tais como e-mails, comunicadores instantâneos ou *messengers* e o próprio *website* da fábrica. Finalmente, para o controle de versões dos artefatos foi configurado um CVS (*Concurrent Version System* ou Sistema de Versões Concorrentes).

3.1. Desafios e Adequações

Com boa parte das principais decisões do projeto definidas a equipe começou a enfrentar alguns desafios, principalmente ligados à distribuição física dos envolvidos

e falta de domínio das tecnologias utilizadas. Os principais desafios identificados durante o projeto são listados abaixo:

Carência de Processos: A definição do processo foi uma das primeiras preocupações da equipe, tendo em vista que sem um processo bem definido, com as fases e os padrões a serem seguidos, os riscos seriam maiores. A maioria dos processos de desenvolvimento amplamente reconhecidos pela literatura não prevê a distribuição do time, e podem não atender adequadamente a esse modelo de desenvolvimento.

Parte da equipe ficou responsável pela definição do processo, que deveria levar em conta a distribuição física dos participantes, a pouca experiência da equipe com a tecnologia de desenvolvimento JavaME e a necessidade de um processo leve, que contemplasse uma quantidade menor de artefatos. O Processo definido pela equipe buscou integrar melhores práticas de metodologias reconhecidas pela literatura, adaptando para a realidade do projeto.

Falta de domínio e experiência no ambiente e ferramenta de desenvolvimento: Um novo projeto, uma nova equipe e uma nova tecnologia podem trazer insegurança ao time. Adicionando a isso o fato de a equipe trabalhar distante, pode aumentar mais ainda o problema. O colega com mais experiência não está ao lado para responder a uma dúvida rápida, ajudar em um impedimento ou problema, ou compartilhar uma solução. A troca de conhecimento sobre o projeto e a tecnologia envolvida é dificultada. A maioria dos membros da equipe não tinha conhecimento das tecnologias necessárias para o desenvolvimento.

Para contornar tal problema, houve treinamento presencial para a equipe sobre a instalação, configuração e adaptação do ambiente. O gerente de configuração atendeu individualmente todos aqueles membros que, mesmo após o treinamento, permaneceram com problemas. A equipe, embora tenha atrasado algumas atividades pelas dificuldades na configuração do ambiente de desenvolvimento, conseguiu, principalmente com os esforços dos membros mais experientes, resolver os problemas e dar continuidade aos trabalhos.

Estimativas: Com a distribuição do time, o acompanhamento por parte da gerência é menor, e a falta de comprometimento com as atividades do projeto pode acontecer, levando as estimativas irreais e ao atraso no andamento do projeto. A gerência começou a atribuir às atividades a equipe em um encontro presencial no início de cada semana e a estimar a quantidade de horas necessárias para a realização de cada atividade, buscando uma distribuição equivalente. Ao final do tempo previsto, o membro deveria reportar à gerencia a realização ou não da mesma, as dificuldades e impedimentos. O acompanhamento era realizado por e-mails e em reuniões presenciais no início e fim da semana.

Logo nas primeiras semanas percebeu-se a fragilidade desta abordagem, já que, como o gerente era o responsável pelas estimativas, os membros não estavam realmente comprometidos com as datas que eram estabelecidas e com o decorrer dos trabalhos, houve um aumento e uma maior dependência entre as atividades, atrasando o projeto. Quando a gerência percebeu que as estimativas não eram realistas, e os prazos do projeto estavam comprometidos, a equipe começou a usar uma estratégia,

no qual, todos os membros passaram a utilizar um processo de autogerenciamento. Com o auxílio de uma ferramenta *web*, a gerência ou o próprio responsável cadastrava as atividades e o tempo estimado para a realização. Percebeu-se que, neste projeto, quando o próprio membro era responsável pela estimativa de suas atividades, o grau de comprometimento era maior.

***Feedbacks* demorados e deficiência nas ferramentas de comunicação**: Embora o time tivesse contato face-a-face em determinados períodos, a maior parte da comunicação era por meio de ferramentas. A principal ferramenta de comunicação utilizada foi e-mail, o que demandava um tempo de espera bem maior do que se o outro membro estivesse trabalhando no mesmo ambiente físico. O e-mail, por ser um meio de comunicação assíncrono, em diversos momentos não atendeu completamente as necessidades do time, pois, as dúvidas e informações só eram respondidas quando outros membros estavam disponíveis. Foi observada também a deficiência de algumas ferramentas, já que na maioria dos softwares testados para conversas via áudio, a equipe teve dificuldades, como: (1) Suportar todos os membros na chamada; (2) Falhas no áudio; e, (3) Instabilidade de conexão.

Assim, quando era necessário realizar uma reunião *online* via conferência, um moderador era definido para coordenar a conversa, controlando a ordem de fala dos membros do time, dessa forma, evitava-se sobrecarga na conversação. Como as ferramentas utilizadas não conseguiam suportar a quantidade de usuários da equipe, a solução foi intensificar o uso de ferramentas assíncronas, principalmente o e-mail.

Controle dos Riscos: Embora os riscos tenham sido levantados no início e durante todo o projeto, assim como as respostas aos mesmos, a implementação dos planos de resposta dentro de um desenvolvimento distribuído é difícil. A gerência acaba ficando com a responsabilidade de mitigar ou conter o risco sem que o time se envolva efetivamente. Com a alta probabilidade, pela pouca experiência da equipe em projetos distribuídos, de riscos serem ignorados, a equipe se reuniu e identificou em conjunto os riscos do projeto logo no início dos trabalhos. Como a gerência era responsável pelo acompanhamento do projeto e por desempenhar outras atividades, não conseguir acompanhar e realizar as ações de resposta aos riscos, foi definido que cada membro, de acordo com o papel que exercia dentro da equipe, ficar responsável por acompanhar e implementar ações de mitigação para reduzir a possibilidade de o risco relacionado ao seu papel acontecer.

4. Ensino / Aprendizagem

Aprender os conceitos de Engenharia de Software em sala de aula é essencial para a formação de um profissional de software, porém, aplicar esse conhecimento em projetos reais ainda na formação desses profissionais é um diferencial e importante caminho a ser seguido para a melhoria desse processo tão complexo. A disciplina na qual o projeto CitIX foi realizado exigiu que os membros estudassem e aplicassem os conceitos da Engenharia de Software em ambientes distribuídos, favorecendo a aplicação e ampliação do conhecimento teórico dos estudantes e a preparação dos mesmos nessa abordagem de desenvolvimento distribuído que vem se tornando cada vez mais presente nas organizações em conseqüência da globalização.

A experiência de desenvolver um projeto de software com todas as etapas formais que são sugeridas pela engenharia de software ou parte delas, com um cliente e prazos reais, fornece aos alunos um conhecimento prático e real da dinâmica, das incertezas e dos riscos que envolvem o desenvolvimento de um software. Além disso, com o desenvolvimento, o uso de técnicas, métodos, processos e ferramentas acontecem no dia-a-dia do projeto, permitindo aos envolvidos conhecer e aplicar as diversas soluções existentes e ter uma visão crítica das mesmas, o que seria difícil apenas com o conhecimento teórico da literatura cientifica.

O acompanhamento das fábricas de software se dava semanalmente pelos professores das disciplinas. Os alunos reportavam as atividades realizadas e os artefatos gerados no decorrer da disciplina, expondo como realizaram tais atribuições. Por meio desse acompanhamento, era possível gerar discussões entre a fábrica, os professores e as outras fábricas de software sobre boas práticas, metodologias, técnicas, ferramentas, com intuito gerar análises e debates para mitigar riscos e identificar melhores soluções para possíveis problemas.

O projeto proporcionou aos seus participantes uma experiência única, principalmente pelas suas características singulares, que se tornaram grandes desafios: a equipe nunca havia trabalhado junta ou em um projeto distribuído, ninguém se conhecia; e, boa parte da equipe desconhecia a tecnologia necessária para o desenvolvimento do projeto.

Na pratica diversos conceitos/praticas puderam ser trabalhados, levando alguns participantes do grupo a ter o primeiro contato real ou aprimorar os conhecimentos que tinham, dentre os principais:

- Adequação e utilização do processo de desenvolvimento RUP (utilizando práticas das metodologias ágeis) para a realidade distribuída;
- Gerência de configuração: nomenclatura de documentos, controle de versão, solicitações de mudanças, auditoria, backup, entre outros;
- Gerência de projeto: planejamento, estimativas, distribuição e acompanhamento de tarefas, e utilização de ferramenta para gestão;
- Gerência de qualidade de software: definição de um processo com padrões e procedimentos, e definição de métricas;
- Plano de teste de software: planejamento e execução de testes, definição de casos de testes, ferramentas de gestão de mudanças;
- Gerência de Comunicação: definição e padronização dos diferentes meios de comunicação e suas formas de utilização.

Diversas conclusões podem ser extraídas desse projeto, o cliente nem sempre estará presente para acompanhar a evolução e dar seu *feedback*, a equipe precisa ter um plano de comunicação estabelecido e conhecido por todos, os papéis devem estar bem definidos, o processo deve garantir padrões e a qualidade do produto, entre outros. Porém, a principal conclusão que se pode chegar é que, por mais que a teoria nos direcione e ajude a fazer as melhores escolhas, é na utilização das teorias,

práticas, métodos, processos e ferramentas da engenharia de software que é possível ter um conhecimento realmente sólido, real e crítico da utilização dos mesmos, pois a aplicação prática, em paralelo ao ensino da teoria, solidifica fortemente o aprendizado e visa garantir um maior aproveitamento dos conceitos teóricos.

5. Conclusões e Trabalhos Futuros

Com a crescente demanda por produtos de software, é necessário que os envolvidos no processo de desenvolvimento de software busquem por alternativas que possibilitem melhorias seja em custo, qualidade ou tempo de desenvolvimento.

Este estudo relatou a experiência de uma equipe de alunos de pós-graduação que foram incumbidos da realização de um projeto de desenvolvimento distribuído de software na disciplina de engenharia de software por meio de uma fábrica de software. A experiência prática dos alunos permitiu aos mesmos identificar desafios e soluções durante a realização do projeto, além de ter uma visão de como a experiência prática em um projeto real ajuda a equipe a ter um melhor entendimento das teorias, práticas, métodos, processos e ferramentas que envolvem a engenharia de software.

Com o relato é possível perceber o cuidado da equipe em utilizar as melhores práticas da engenharia de software e adaptá-las para a realidade de um projeto distribuído de desenvolvimento, tornando a experiência dos alunos bastante interessante para outros estudantes e cursos na área que busquem melhorar seus programas de ensino por meio do uso de fábrica de software e projetos com clientes reais, no qual, os alunos têm prazos e responsabilidades bem definidos.

Neste sentido, uma abordagem semelhante à aplicada neste relato pode ser utilizada por outros grupos no processo de ensino e aprendizagem da Engenharia de Software, possibilitando o compartilhamento de experiência semelhante e buscando proporcionar melhor qualificação dos profissionais, por meio da teoria aliada a prática.

Referências

Cavalcanti, Ana Paula de Carvalho, Bandeira, Liane Ribeiro Pinto, Donegan, Paula Marques. (2004). Um modelo de gerência de projetos baseado no RUP com aplicações em PMBOK, VI Simpósio Internacional de Melhoria de Processos de Software, São Paulo.

Herbsleb, J. D.; Moitra, D. (2001). Guest editors' introduction: global software development. IEEE Software.

IN953 (2008), Software Engineering: Building Open Source Software Factories. Disponível em http://www.cin.ufpe.br/~in953/.

Siy, H. P., Herbsleb, J. D., Mockus, A., Tucker, G. T., and Krishnan, M. (2001). "Making the Software Factory Work: Lessons from a Decade of Experience". In Proceedings of the 7th international Symposium on Software Metrics (April 04 - 06, 2001). METRICS. IEEE Computer Society, Washington, DC, 317.