SCRUM APLICADO AO DESENVOLVIMENTO DO *SOFTWARE* EM DISCIPLINAS DE DESENVOLVIMENTO WEB

SCRUM APPLIED TO SOFTWARE DEVELOPMENT IN WEB DEVELOPMENT DISCIPLINES

Marcelino Santos Nascimento

Analista de TI. Mestre em Ciência da Computação (UFPE). Docente da FAMETRO.

Joseane Pereira Rodrigues

Analista de TI. Mestre em Ciência da Computação (UFPE). Docente da FAMETRO.

RESUMO

O mercado de trabalho requer profissionais que possam se adaptar de forma rápida às novas tecnologias e metodologias, que trabalhem em equipe e que possuam habilidade de comunicação e aptidão em gestão, para que desenvolvam produtos de forma rápida e com qualidade. Entretanto, esse cenário não é comumente encontrado em ambiente acadêmico dos cursos de tecnologia e computação, onde, na maioria das vezes, os estudantes executam na mesma disciplina, exercícios práticos não relacionados na aplicação prática dos conceitos teóricos vistos em sala de aula. E, com o intuito de propor melhorias nesse cenário, este artigo apresenta os resultados da aplicação da metodologia ágil Scrum no ciclo de desenvolvimento de *software* em ambiente acadêmico.

Palavras-chave: Engenharia de software. Desenvolvimento ágil.

ABSTRACT

The job market requires professionals who can adapt quickly to new technologies and methodologies, work as a team, and have communication skills and management skills to develop products quickly and with quality. However, this scenario is not commonly found in the academic environment of technology and computing courses, where, in most cases, students perform in the same discipline, practical exercises not related to the practical application of the theoretical concepts seen in the classroom. And, with the purpose of proposing improvements in this scenario, this article presents the results of the application of the Agile Scrum methodology in the software development cycle in an academic environment.

***Keywords**: Software engineering. Agile development.*

1 INTRODUÇÃO E MOTIVAÇÃO

Hoje, as organizações de Tecnologia da Informação (TI) estão enfrentando um sério desafio, que é a presença crescente e contínua de *software* e serviços consumidos diariamente pela sociedade. Nos últimos anos, esse cenário tem se potencializado com a popularização dos dispositivos móveis.

Assim, nichos de mercados cada vez mais competitivos, com mudanças quase que rotineiras nos softwares e uma variedade imensa de dispositivos e sistemas operacionais, são parte das ações capitaneadas pelas organizações de TI. Aliado a isso, os sistemas ficam cada vez mais complexos, integrados a outros serviços e exigindo um alto grau de confiabilidade e disponibilidade dos mesmos, demandando equipes cada vez mais qualificadas e um processo de implantação cada vez mais eficiente e robusto.

Em paralelo à evolução da tecnologia, as faculdades e universidades têm enfrentado desafios na formação de profissionais dessa área. Alguns dos principais problemas enfrentados por docentes e estudantes na academia são - como aliar as práticas do mercado de trabalho durante a vivência acadêmica dos estudantes e - como, no desenvolvimento de *software* em ambiente acadêmico, transformar uma boa ideia em um sistema e entregá-lo utilizável, aplicando as mesmas práticas de mercado em ambiente acadêmico.

Construir, implantar, testar e entregar software em apenas um semestre, e, ao mesmo tempo, preparar o estudante para o mercado de trabalho são questões a serem avaliadas no decorrer deste artigo. Para isso, utilizaremos o conceito e práticas ágeis no desenvolvimento de *software* com o objetivo de aliar a teoria com a prática, de modo a construir um conhecimento sólido e crítico. Dessa forma, os alunos puderam aprender na prática como entregar software de forma confiável e rápida, apresentando resultados compatíveis com entregas feitas em empresas.

A utilização de uma metodologia ágil para o desenvolvimento de *software* em um ambiente acadêmico, onde os estudantes precisam interagir da forma mais colaborativa possível, melhorando o feedback do processo, de modo que os problemas são identificados e resolvidos o mais cedo possível (HUMBLE; FARLEY, 2014) se demonstrou produtiva. Esse processo permite a liberação de partes do software menores, funcionais, mais rapidamente.

Este artigo relata a experiência em adotar uma metodologia ágil no desenvolvimento de software durante o ensino da disciplina de Desenvolvimento de Sistemas para WEB. Cada equipe pôde atuar de forma colaborativa, contribuindo para que os estudantes adquirissem uma valiosa experiência em projetos reais.

O resultado desse estudo baseia-se em dados quantitativos e descrições do desenvolvimento de software, coletados das equipes de desenvolvedores da disciplina de Desenvolvimento de Sistemas para WEB dos cursos de Sistemas de Informação e Análise e Desenvolvimento de Sistemas de uma faculdade provada em Fortaleza-CE.

O restante deste artigo está estruturado da seguinte forma: na Seção 2, abordamos a base desta pesquisa com um resumo dos trabalhos relacionados; na Seção 3, apresentamos a estrutura do processo de desenvolvimento de software com o modelo ágil em ambiente acadêmico; na Seção 4, discutimos os resultados do estudo de caso realizado em uma faculdade; e na Seção 5, apresentamos as conclusões.

2 TRABALHOS RELACIONADOS

Catalogamos nesta seção alguns trabalhos publicados que descrevem experiências com métodos relacionados ao descrito neste artigo. E, a seguir, eles são apresentados.

Borges, Carvalho e Morais (2012) propõem uma análise das competências e habilidades dos residentes de Fábrica de *Software*, utilizando metodologias ágeis. Como metodologia adotou-se o *SCRUM* (para a gerência do projeto), combinando com algumas práticas de *Extreme Programming* (para desenvolvimento de *software*). Os autores mencionam que um dos aspectos interessantes desta me-

todologia é que a equipe de desenvolvimento mantém um contato direto com o cliente e isso auxilia no processo de maturidade profissional dos alunos e no engajamento com prazos, responsabilidades e expectativas do cliente.

Dos Santos e Pinto (2012) utilizaram processos ágeis como *Scrum*, *Kanban*, aliados ao PBL, criando uma metodologia de desenvolvimento de software, alinhando o conteúdo das disciplinas e o contexto das práticas de mercado para o desenvolvimento de aplicações. A estratégia foi aliar os conceitos da técnica PBL (*Problem-Based Learning*), que sugere um estilo de pedagogia, cujo ensino é baseado em práticas reais de soluções de problemas com metodologias ágeis. Para isso, os estudantes participavam de um programa que tinha como objetivo desenvolver duas aplicações móveis por estudante: uma aplicação de baixa complexidade e outra de maior complexidade. Os estudantes realizavam o levantamento das ideias durante as aulas práticas, relacionadas à concepção e identificação de produtos. As ideias foram selecionadas pelos membros da equipe, e, em seguida, enviadas ao cliente para validação.

Kropp e Meier (2013) propõem uma abordagem para o ensino de desenvolvimento de *software* ágil, em que as práticas e valores ágeis são aplicados e repetidos até se tornar um hábito para alunos de graduação. Analisando a utilização de metodologias ágeis na indústria, eles identificaram que apenas algumas das práticas ágeis eram usadas e ainda menos eram aplicadas de forma frequente e completa. A avaliação constatou que a falta de profissionais qualificados, com conhecimentos sólidos em práticas ágeis, era o principal motivo desse *gap* na indústria. Os alunos de um curso de Ciência da Computação aplicaram práticas de engenharia e gerenciamento ágeis, com atenção especial aos valores ágeis, dentro de um projeto de desenvolvimento de *software*. Os alunos foram divididos em equipes *Scrum* de seis a oito alunos e trabalharam no desenvolvimento de jogos de computador com o objetivo de fortalecer as habilidades ágeis de desenvolvimento de *software*. Os autores aplicaram um questionário que mostrou que a metodologia utilizada no curso foi bem recebida e que os participantes aprenderam muito sobre metodologias ágeis.

Zoezo, De Ponte e Lucredio (2013) estruturaram um curso de especialização (*Lato Sensu*), em uma universidade, para que todo o curso se ajustasse aos princípios do *Scrum*. O foco da metodologia é preparar os estudantes para trabalhar com metodologias ágeis. Nesta abordagem, os alunos aprenderam, na prática, desenvolvimento de *software* em diferentes técnicas e conceitos, abordados em diferentes disciplinas, tais como programação, modelagem, *design* de banco de dados, interface homem-computador, desenvolvimento web e gerenciamento de projetos. Os autores relataram algumas dificuldades enfrentadas nos projetos desenvolvidos. Por exemplo: Muitos projetos chegaram ao final do curso incompletos ou com falhas fundamentais que não poderiam ser corrigidas no tempo. A avaliação final é de que os objetivos principais da metodologia foram alcançados, desde os *workshops* teóricos mais simples para uma atividade prática mais rica, com o foco no desenvolvimento real, simulando desafios reais em ambiente dinâmico.

Nas abordagens descritas nesta seção, fica claro o objetivo dos autores em proporcionar para os estudantes um ambiente de aprendizagem baseada na experiência, onde os estudantes participam no desenvolvimento de vários projetos de *software* que adquirindo expertise para atender a uma necessidade de uma empresa real.

3 PROCESSO DE DESENVOLVIMENTO DE *SOFTWARE*

A demanda por profissionais mais capacitados para o mercado de trabalho de TI tem aumentado a cada ano. A qualidade do profissional de informática passa pela formação de diferentes perfis de profissionais, sendo impulsionado pelas mudanças ocorridas nas empresas, afetando assim, o perfil exigido pelo mercado, e, consequentemente, a qualidade de sua formação.

Aplicações de negócio são difundidas por todos os setores organizacionais de

uma empresa. Essas aplicações são utilizadas por áreas como contabilidade, finanças, vendas, produção e assim por diante. Com isso, é crescente a demanda por profissionais com conhecimento do desenvolvimento e gerenciamento de sistemas. Tais perfis de profissionais demandam conhecimentos que favoreçam à inovação, planejamento e coordenação da infraestrutura de informação. Com esse pensamento, surgiu a ideia de implementar em sala de aula, em um curso de TI, na disciplina Desenvolvimento de Sistemas para WEB, todas as fases de desenvolvimento de *software* de forma contextualizada, como praticado na indústria.

É comum, nos cursos de graduação e nos livros utilizados nesses cursos, o estudo conjunto de vários enfoques, apresentando-se, de modo amplo e desconexo, uma série de conceitos como sendo complementares. A disciplina de Desenvolvimento de Sistema para WEB possui um conteúdo teórico e prático. Geralmente, essa disciplina é ministrada através da apresentação de conteúdo teórico com a execução de exercícios isolados para cada conteúdo abordado em sala. No entanto, é sabido que um enfoque eclético, no qual se misturam conceitos de contextos diferentes durante atividades práticas não é a melhor solução para a preparação do estudante para o mercado de trabalho. Percebeu-se a necessidade de alinhar o conhecimento técnico de programação à parte processual do ciclo de desenvolvimento de um software. A partir dessa premissa, aliada ao curto tempo de um semestre para o desenvolvimento de um *software*, decidiu-se adotar um processo de desenvolvimento de *software* ágil que pudesse proporcionar uma entrega rápida mais eficiente, baseadas em controles.

3.1 Metodologia e estudo de caso

Nesta seção, descreveremos os principais passos da metodologia aplicada para a utilização no desenvolvimento do *software* em uma disciplina de graduação em um curso da área de Tecnologia utilizando o *Scrum*. A aplicação da metodologia apresentada neste artigo descreve o uso do *Scrum* como forma de acompanhar as atividades práticas de entrega do *software* no decorrer do semestre, dentro um ciclo de desenvolvimento.

3.3.1 Ciclo de vida do desenvolvimento

A metodologia descrita neste artigo foi adotada na disciplina de Desenvolvimento de Sistemas para WEB do curso de Sistemas de Informação e Análise e Desenvolvimento de Sistemas para WEB, ministrada para os alunos no 1° e 2° semestre de 2015 e 2016, com um total de 120 alunos. A disciplina é obrigatória e é ofertada no 3° ano do curso, possuindo dois créditos, com sessenta horas-aula no semestre. O objetivo desta disciplina é apresentar aos alunos as técnicas de engenharia de *software* e tecnologias utilizadas no processo de desenvolvimento de *software*.

O processo de desenvolvimento do *software*, conforme proposto neste artigo, em sua primeira fase do processo, se inicia no primeiro dia de aula, quando o professor seleciona as equipes que deverão entregar ao final do semestre um produto (*software*) funcional que irá atender um nicho de marcado definido pelo docente. Cada equipe irá trabalhar seguindo os princípios da metodologia ágil de desenvolvimento de *software SCRUM* (SCHWABER, 2017). As equipes têm no mínimo 5 (cinco) estudantes e no máximo 9 (nove), dependendo do tamanho da turma, visando manter um time de desenvolvimento com no mínimo 3 (três) desenvolvedores. Até a escrita deste artigo, essa metodologia havia sido aplicada em três turmas, cada turma com a média de 40 alunos. Neste momento também foram definidos os papéis previstos no *SCRUM*, o *Product Owner* e o *Scrum Master*.

Neste primeiro encontro, é agendada a reunião de levantamento dos requisitos para cada time. Os times *SCRUM* terão a oportunidade de coletar os requisitos da aplicação para iniciar a montagem do *Product Backlog* para cada *software* a ser desenvolvido. O professor, fazendo o papel de Cliente, repassa ao *Product Owner* a demanda de desenvolvimento, em conjunto com os requisitos da aplicação. Isso

permitiu a aplicação realista da metodologia ágil *Scrum*, que prega interação constante com o cliente. Nesta fase, cada time precisa montar as *Sprints* com base nos itens do *Product Backlog*, sendo que as *Sprints* têm em média de 15 a 20 dias. Os *softwares* desenvolvidos dentro dos projetos acadêmicos não são complexos o suficiente para que demandem *Sprints* maiores que 20 dias. Além disso, um dos objetivos é promover o trabalho em equipe, comunicação, interação e colaboração, que são situações que podem ser vivenciadas em um período menor. Durante o período do projeto (um semestre), os times *Scrum* terão no máximo quatro *Sprints* para entregar o produto final. Esse preparo inicial é feito a partir da reunião de levantamento de requisitos, combinado com uma reunião de planejamento de *Sprint*.

Na segunda fase, fase de desenvolvimento, de cunho totalmente prático, realizam-se inspeções periódicas programadas dentro da *Time-Box* do *Scrum*, chamada *Daily Scrum*. Aqui fizemos a primeira adaptação na metodologia *Scrum* para adequar à realidade do ambiente acadêmico. Esta reunião não é realizada diariamente conforme recomendada pelo *Scrum*, pois encontramos mais efetividade, para este cenário, realizá-la a cada três dias, para que o aluno possa ter um tempo maior para absorver os conceitos teóricos e aplicá-los no desenvolvimento. As reuniões são feitas de forma *online* e rápida, via *whatsapp* ou *Skype*. Desta forma, o professor pode acompanhar a evolução de cada equipe de desenvolvimento. O acompanhamento por parte do professor também é feito através da inspeção do código submetido ao repositório pelas equipes. Inspeções visam levar os alunos a desenvolverem projetos de forma gradual. Ao final de cada *Sprint* é realizada a reunião de *Sprint Review*, na qual cada time apresenta um produto final de acordo com o que foi planejado. Este momento é importante para avaliar o quão bem sucedido foi o time para aquela *Sprint* e o que pode ser melhorado para a próxima.

Durante o ciclo de desenvolvimento, cada time precisa entregar alguns artefatos do projeto. O *Product Backlog* é um destes artefatos que deve ser gerado logo após reunião de levantamento de requisitos. Para construção do backlog do produto e de cada *sprint*, cada equipe construiu um *kanban board* utilizando *Track & Plan*, para descrever as histórias de usuário e a sequência de implementação. Além disso, essa ferramenta permite relacionar as atividades descritas no *backlog* com o código fonte desenvolvido pela equipe de desenvolvimento, por meio de ferramentas de controle de versões. Nos projetos, adotamos o *Git* para o versionamento do projeto, que assim como o *Track & Plan*, faz parte da estrutura de *PaaS IBM Bluemix*.

Além do *Product Backlog*, os times também precisam entregar *Sprint Backlog* e os requisitos definidos para cada *Sprint* desenvolvidos e testados. As *Sprints* são montadas de acordo com os requisitos levantados, respeitando o conteúdo teórico ministrado durante o semestre, por exemplo: A primeira *Sprint* contempla apenas a construção do *Front-end* da aplicação.

Nos primeiros semestres de implementação dessa metodologia, utilizamos como ferramenta para gerenciamento do ciclo de vida da aplicação e versionamento do código a nuvem *PaaS IBM Bluemix* (IBM Bluemix, 2017), mantendo a ideia de termos o ambiente de desenvolvimento, centralizado e mais acessível a todos da equipe, e para facilitar a interação entre professor e aluno em sala de aula, e, principalmente, fora dela, através da utilização de ambiente de desenvolvimento na nuvem. Dentro da esfera educacional, conceber e administrar situações problema, desenvolvendo a cooperação entre os alunos e formas simples de ensino mútuos, mesmo geograficamente separados, construindo um conhecimento coletivo.

No momento de escrita deste artigo, estamos fazendo alguns testes com ferramentas grátis. Para controlar o *Product Backlog*, estamos testando o Trello. O Trello é uma ferramenta de gerenciamento de projetos muito utilizada para controle de tarefas. A terceira e última fase do processo consiste na entrega e apresentação dos *softwares* desenvolvidos. Nessa fase ocorre a avaliação dos projetos na disciplina. Notas individuais foram dadas aos alunos conforme a participação no projeto, de acordo com a função que cada aluno exer-

cia. Os critérios de avaliação foram definidos baseados em três pilares principais: 1) documentação do projeto (planejamento do projeto (*backlog* e *Sprints*), diagramas da UML e plano de testes); 2) uso da metodologia de desenvolvimento (participação em reuniões, cumprimento de prazos, divisão das tarefas pelos membros do grupo e uso adequado do Scrum para o gerenciamento do projeto); e 3) produto desenvolvido (atendimento dos requisitos). Essa avaliação final de tudo que foi produzido é combinada com as informações coletadas durante as inspeções periódicas, feitas durante todo o semestre.

3.3.2 Scrum - estudo de caso

O método tradicional de ensino utilizado nos laboratórios das faculdades e universidades consiste na execução de atividades de implementação sobre algum problema a ser modelado por uma linguagem de programação. Geralmente, cada aluno implementa o mesmo exercício com o objetivo de aprender de forma prática o que foi abordado de forma teórica na sala de aula. Sabemos que em uma empresa, o desenvolvedor não recebe um documento escrito com os passos a seguir para implementar um *software*. As necessidades do cliente são coletadas durante o levantamento de requisitos, e o desenvolvedor deve abstrair os detalhes para o desenvolvimento do produto. Para isso, ele utiliza de técnicas e metodologias da engenharia de *software*. Cada requisito é avaliado e implantado de acordo a atender as necessidades do cliente. Neste momento, surgem as seguintes perguntas: O que acontece depois da identificação de requisitos, projeto, desenvolvimento e testes da solução?

Como unir e coordenar todas essas atividades para tornar o processo tão eficiente e confiável quanto possível durante a execução? Como fazer desenvolvedores, testadores e pessoal de operação trabalhar juntos de maneira eficiente (HUMBLE, 2014)? Em um ambiente acadêmico, essas perguntas são ainda mais pertinentes, considerando que os estudantes precisam trabalhar juntos. Porém, esses momentos de trabalho colaborativo ocorrem com pouca frequência, geralmente somente nos dias de aula. Buscando resolver esse problema e tornar o ambiente de desenvolvimento mais contextualizado, similar ao que podemos encontrar uma empresa, incluímos o processo de entrega contínua. Proporcionando ao aluno a oportunidade de enfrentar e analisar em conjunto situações complexas, práticas e problemas profissionais.

Utilizando metodologias ágeis, os times *Scrum* trabalham em equipes, nas quais os componentes estão geograficamente separados. Os estudantes utilizaram um repositório principal para armazenar o código e manter versões com cada modificação realizada. As equipes trabalham de forma simultânea com a criação de *branches*, que irão conter uma cópia completa do código para que seja modificado por parte da equipe sem a interferência no código principal ou em *branches* de outros programadores. As atividades de *build*, testes e implantação devem, então, ser executadas a cada nova modificação concluída, fazendo com que as alterações incrementais de *software* sejam testadas automaticamente e, frequentemente, implantadas em ambientes de produção e fiquem disponíveis para inspeção pelo professor. Um ciclo de desenvolvimento completo, com metodologia e ferramentas utilizadas pela indústria aplicadas em ambiente acadêmico, proporcionou aos estudantes a experiência de dirigir um grupo de trabalho, conduzir reuniões, administrar crises e conflitos interpessoais.

No ambiente acadêmico, as inspeções periódicas previstas no *Scrum* servem também como uma boa técnica de avaliação contínua de cada estudante, utilizando ferramentas que funcionam na nuvem, como o GIT, para repositório de código fonte centralizado, e Jenkins (ARMENISE, 2015), para a compilação, teste e implantação da aplicação. Graças a todo esse ferramental, para cada artefato produzido foi possível rastrear onde este foi implantado, por quem e quando foi implementado, além de avaliar os resultados dos testes executados para cada artefato, dando ao professor visibilidade completa sobre as atividades e user stories desenvolvidas na *Sprint* e a possibilidade

de avaliar o código construído e ajudar na correção de problemas.

O esforço do professor em avaliar o código de cada time foi diluído dentro todo o semestre, até a entrega final dos projetos. Isso proporciona um ciclo de *feedback* mais rápido para os estudantes na aplicação das técnicas de desenvolvimento e engenharia de *software* e alguns benefícios, como o aumento da produtividade.

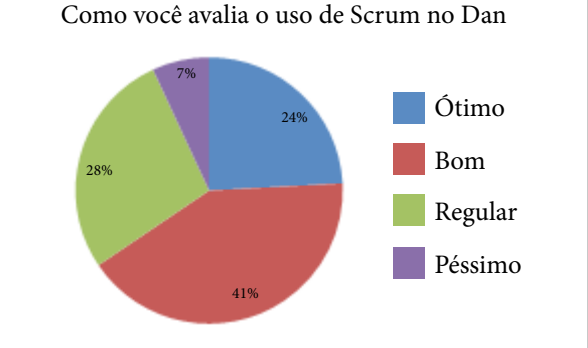
4 RESULTADOS

Esta seção fornece uma compreensão da relevância da metodologia mencionada ao longo deste artigo. A partir dos resultados obtidos diante da aplicação na prática da metodologia mencionada na seção 3, aplicamos um questionário com o objetivo de capturar a opinião dos estudantes sobre a utilização do *Scrum* como metodologia para o ciclo de desenvolvimento de *software*.

Com o intuito de avaliar a aceitação dos estudantes à metodologia proposta e a utilização do conjunto de ferramentas, foi aplicado um questionário com todos os alunos de duas turmas da disciplina Desenvolvimento de Sistemas para WEB dos cursos de Sistemas de Informação e Análise e Desenvolvimento de Sistemas. O questionário foi aplicado ao final do semestre, após a apresentação dos *softwares* desenvolvidos para toda a turma.

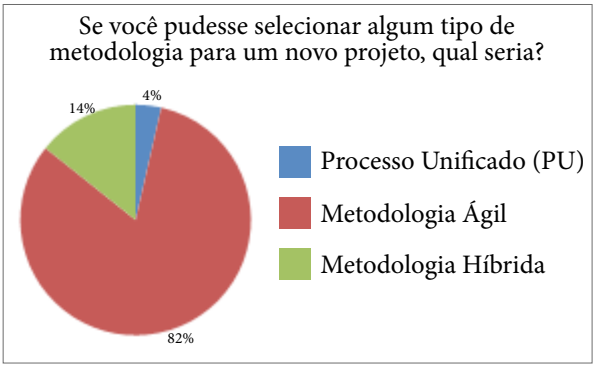
A seguir são apresentados os resultados da aplicação dos questionários.

Figura 01 - Gráfico de utilização do *Scrum* no Dan[1].



Fonte: Autores.

Figura 02 - Gráfico de aceitação do *Scrum*.



Fonte: Autores.

A adoção de uma metodologia ágil de desenvolvimento de *software* em substituição à metodologia tradicional de execução de exercícios contribuiu de forma significativa na antecipação dos alunos no contato com atividades profissionais. Com a adoção do *Scrum* foi possível avaliar na prática o domínio técnico dos estudantes sobre as boas práticas de Engenharia de *Software*, geração de documentação, gerência de projetos e programação, além das competências pessoais e profissionais, tais como elaborar e negociar em um projeto e administrar os recursos. Com a metodologia tradicional de exercícios, não era possível avaliar essas competências de forma conjunta. Ao final do semestre, os alunos avaliaram como muito positiva a adoção de uma metodologia ágil na disciplina de desenvolvimento de sistemas para WEB, conforme pode ser visto na figura 2.

A comunicação via mensagem instantânea (*Whatsapp*) mostrou-se o meio mais eficiente para realização das reuniões dos times *Scrum*, conforme figura 3. Essa estratégia tornou o trabalho no desenvolvimento do *software* mais dinâmico, já que a barreira da distância foi quebrada para a realização dos encontros. Cada equipe pôde se reunir em horários definidos, de forma remota, compartilhando as atividades realizadas em cada *Sprint*, tornando o processo de inspeção mais eficiente.

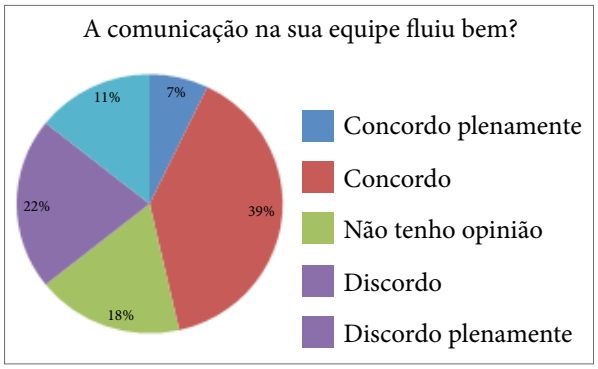
[1] Desenvolvimento de aplicações nas nuvens.

Figura 03 - Comunicação entre as equipes.

Fonte: Autores.

A validação das atividades realizadas e confirmação dos artefatos produzidos são feitas no ambiente de nuvem. Através dessas reuniões remotas, os estudantes desenvolveram a habilidade de trabalhar em equipe durante todos o semestre, se adaptando ao ambiente de trabalho, exercitando a habilidade de comunicação de forma oral e escrita. Além disso, os estudantes também puderam exercitar a competência de colaboração, no apoio ao membros da equipe no desenvolvimento do *software*.

Figura 04 - Interação entre os membros das equipes.



Fonte: Autores.

A interação entre os membros das equipes foi um dos principais pontos a ser avaliado por esta pesquisa. Na figura 4, podemos perceber que a maioria dos estudantes concorda com a aplicação da metodologia de desenvivemento de *software* nos moldes sugeridos neste artigo. Essa aceitação pode ser explicado devido ao fato de que os alunos são incentivados a buscar solução para os problemas, de forma conjunta e colaborativa. Para a realização de uma tarefa, eles são estimulados a ouvir, compreender, negociar. Os alunos, além de ter conhecimento técnico, devem ser capazes de aplicar este conhecimento em problemas do mundo real. Essa pergunta teve como objetivo avaliar o processo de comunicação em todas as fases do processo de desenvovimento. Assim, concluímos que a entrega contínua e metodologias ágeis formaram uma dupla processual eficaz na preparação dos estudantes de cursos de graduação de TI para o mercado de trabalho.

5 CONCLUSÕES

Em um ambiente acadêmico, a aplicação de uma estratégia de aprendizagem baseada na experiência, em que os alunos participam no desenvolvimento de vários projetos de software do mundo real, se mostrou bastante eficaz de acordo com avaliação feita neste artigo. Cada estudante ganhou participação ativa e experiência no ciclo de vida de desenvolvimento de um *software*.

Apesar das dificuldades, todas as equipes conseguiram finalizar seus projetos no prazo previsto, produzindo projetos de boa qualidade.

Um dos grandes benefícios dessa metodologia é que as equipes de desenvolvimento mantêm contato direto com o cliente, que como professor, orienta como as atividades devem ser conduzidas e, como cliente, repassa os requisitos para a construção do software, além de inspecionar o que está sendo desenvolvido. Isso auxilia no processo de maturidade profissional dos alunos, no que se refere aos prazos, responsabilidades, artefatos entregues e expectativas do cliente.

As principais conclusões obtidas, após a aplicação de processos ágeis de desenvolvimento como formas de aproximar os estudantes de graduação ao mercado de trabalho, podem ser resumidas conforme segue:

- Utilização de metodologias ágeis para o desenvolvimento

Metodologias ágeis estão cada vez mais presentes na indústria. Com a adoção do *Scrum* para o desenvolvimento de *software* em

um ambiente acadêmico, os alunos tiveram a oportunidade de exercitar os conhecimentos vistos na disciplina e em outras disciplinas do curso em um único projeto, de forma contextualizada, como ocorre em uma empresa de tecnologia.

- Dificuldade de comunicação entre os membros do time *Scrum*.

A pesquisa demonstrou que os estudantes tiveram alguma dificuldade na comunicação e interação humana dentro dos times Scrum. Mesmo com a participação do professor em algumas reuniões, essa dificuldade se tornou bastante evidente na análise dos resultados.

Trabalhar em equipe, mesmo que geograficamente distribuída, é um cenário comum nas empresas, pois facilita o compartilhamento de tarefas, papéis e responsabilidades. Porém, podem surgir problemas de interação humana que precisam ser adequadamente abordados e gerenciados pelos líderes. Além disso, serve como alvo de aprendizagem para o mercado de trabalho. Dentro do ambiente acadêmico, visando minimizar esses conflitos, podemos incluir um monitor específico, dentro de um projeto de monitoria acadêmica da disciplina, para acompanhar as equipes de perto e apoiar nas dificuldades encontradas.

REFERÊNCIAS

ARMENISE, V. Continuous delivery with Jenkins. **3rd International Workshop on Release Engineering**. Firenze: 2015

BORGES, K. S.; CARVALHO, T. P.; MORAIS, M. A. C. . Programa de Extensão 'Fábrica de Software Acadêmica': contribuindo para a formação profissional na área de informática. In: CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 23., 2012. **Anais**... Curitiba: Sociedade Brasileira de Computação, 2012.

DOS SANTOS, S. C.; PINTO, A. Assessing Pbl with software factory and agile processes: a case study to develop mobile softwares engineers. In: INTERNATIONAL CONFERENCE ON COMPUTERS AND ADVANCED TECHNOLOGY IN EDUCATION (CATE 2012), 15., 2012. **Anais**... Nápoles: 2012

HUMBLE, J.; FARLEY, D. **Entrega contínua**: como entregar software de forma rápida e confiável. Porto Alegre: Bookman, 2014.

INTERNATIONAL BUSINESS MACHINES (IBM). **IBM Bluemix**. Disponível em: <https://www.ibm.com/cloud-computing/bluemix/pt>. Acesso em: 27 de jan. 2017.

KROPP, M.; MEIER, A. Teaching agile software development at university level: values, management, and craftsmanship. In: CONFERENCE ON SOFTWARE ENGINEERING EDUCATION AND TRAINING, 26., 2013. **Anais**... San Francisco: 2013.

SCHWABER, K.; SUTHERLAND J. **O guia do Scrum**. [*S.l.*]: [*S.d.*]. Disponível em: <www.scrum.org>. Acesso em: jan. 2017.

ZOEZO, S.D., DE PONTE, L.; LUCREDIO, D., Using scrum to teach software engineering: a case study. In: FRONTIERS IN EDUC. CONF., 2013. **Proceedings**... Oklahoma City: 2013.