INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO – IF SERTÃO-PE
LICENCIATURA EM COMPUTAÇÃO

EMANUELLA BEZERRA DOS SANTOS

DIFICULDADES DE ADAPTAÇÃO DAS EQUIPES NO USO DE METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE NA DISCIPLINA DE PROJETO DE DESENVOLVIMENTO DE SOFTWARE NO IF SERTÃO – PE

Petrolina – PE
2014

INSTITUTO FEDERAL DE EDUCAÇÃO CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO – IF SERTÃO-PE
LICENCIATURA EM COMPUTAÇÃO

EMANUELLA BEZERRA DOS SANTOS

DIFICULDADES DE ADAPTAÇÃO DAS EQUIPES NO USO DE METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE NA DISCIPLINA DE PROJETO DE DESENVOLVIMENTO DE SOFTWARE NO IF SERTÃO – PE

Monografia apresentada ao Instituto Federal de Educação Ciência e Tecnologia do Sertão Pernambucano, como requisito para obtenção do grau em Licenciatura em Computação.

Orientadora: Prof. Esp. JUSSARA ADOLFO MOREIRA

Petrolina – PE
2014

EMANUELLA BEZERRA DOS SANTOS

DIFICULDADES DE ADAPTAÇÃO DAS EQUIPES NO USO DE METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE NA DISCIPLINA DE PROJETO DE DESENVOLVIMENTO DE SOFTWARE NO IF SERTÃO – PE

Monografia apresentada ao Instituto Federal de Educação Ciência e Tecnologia do Sertão Pernambucano, como requisito para obtenção do grau em Licenciatura em Computação.

Aprovada em _____/_____/ 2014

BANCA EXAMINADORA

_____
Profa.Esp. Jussara Adolfo Moreira
Instituto Federal do Sertão Pernambucano

_____
Profa.Esp. Rossana de Paula Junqueira Almeida
Instituto Federal do Sertão Pernambucano

_____
Prof. Msc. Glécio Rodrigues Albuquerque
Instituto Federal do Sertão Pernambucano

Petrolina – PE

2014

Dedico este trabalho aos meus pais, que, com todo carinho e amor, me mostraram que, com esforço e dedicação, é possível se chegar aonde quiser.

Dedico também à minha Vó Júlia (in memoriam), pois mesmo não estando mais aqui, consigo sentir sua presença constante ao meu lado.

Agradecimentos

Agradeço a Deus, primeiramente, por me dar a oportunidade de ter conhecido pessoas incríveis durante esses quatro anos e, principalmente, por me mostrar o lindo caminho da docência.

Agradeço aos meus pais Rocilda Bezerra e Edivaldo Lopes por todo o carinho, amor e dedicação, por me darem todo apoio quando decidi ser professora, acreditando no meu sonho junto comigo, por me mostrarem os valores mais preciosos das pessoas. Vocês são a minha vida, muito obrigada.

Agradeço aos meus irmãos John Williams e Allan Bezerra que, mesmo com as nossas "brigas", se importaram sempre comigo, me ajudando e estando sempre ao meu lado.

Agradeço ao meu amor, meu noivo, companheiro e amigo Raphael Vidal, por ser paciente comigo durante os 7 anos, estando comigo nos momentos mais difíceis da faculdade e nos momentos mais felizes também. Quero passar o resto da minha vida ao seu lado, conquistando mais e mais sonhos.

Agradeço à minha sogra Maria de Lourdes, por ser, para mim, uma mãe, me tratando sempre como uma filha querida.

Agradeço à minha "PrimaAmigaIrmã", Shena Gonçalves, pelos artigos revisados, todas as vibrações e orações. Mesmo distante, sinto você sempre perto. Obrigada por sempre me acalmar quando eu estive tão aflita.

Agradeço à minha tia linda, Apolônia Bezerra, por tudo que faz por mim, por todo o amor, carinhos e abraços apertados.

Agradeço a todos os meus tios e primos que estiveram sempre torcendo por mim.

Agradeço, especialmente, à professora Rossana Junqueira, por ser, para mim, uma referência como pessoa e profissional, sendo sempre justa, batalhadora e correta. Quando eu crescer, quero ser igualzinha a você! Obrigada por cada conselho, incentivo, puxão de orelha e

principalmente pela sua amizade.

Agradeço à minha professora e orientadora Jussara Moreira, por percorrer comigo esse caminho me mostrando sempre o lado correto a seguir. Obrigada por cada ensinamento, por cada oportunidade, cada e-mail e ligação. Sem você, não tenho dúvidas do quanto seria difícil.

Agradeço aos meus amigos, em especial, a Bruno Teles, Joana D'arque, Vanessa Souza e Raphael Vidal, por cada manhã, tarde e noite de estudo, por cada trabalho, prova e não só por isso, mas também por cada sorriso, abraço apertado, festas, aniversários, cinemas e muitas outras coisas que passamos nesses 4 anos juntos. Quero levar vocês comigo para sempre, em cada lembrança, em cada encontro, para sempre.

Agradeço aos meus queridos amigos do “Grupo da Alegria”, pessoas que quero sempre ao meu lado. Obrigada por cada almoço, cada ida a lanchonete verde e pela preciosa amizade de vocês. Sintam-se todos abraçados nesse momento de alegria para mim.

Agradeço também a todos os meus professores, pela imensa bagagem fornecida de um conhecimento que é inesgotável.

Resumo

Para o sucesso no desenvolvimento de software vários processos devem ser seguidos tanto para pequenas ou grandes soluções. É necessário um bom planejamento para que se tenha bons resultados na utilização das várias metodologias que estão disponíveis no mercado. Partindo desse principio, nesse trabalho ocorrerá uma analise em busca dos principais motivos dos alunos da disciplina de desenvolvimento de projetos de software do IF Sertão - PE, não executarem, como proposto pelos mesmos, os procedimentos das metodologias escolhidas por cada equipe. No final do estudo será proposta um possível solução para melhorar as dificuldades apresentadas pelos alunos durante a disciplina.

Palavras-chave: Metodologias tradicionais. Metodologias ágeis. Engenharia de software. Projetos de software.

Sumário

Lista de Figuras

1. INTRODUÇÃO

A pesquisa está dividida em 6 capítulos, organizados da seguinte forma: no Capítulo 1 está descrito o tema, a motivação para o desenvolvimento da pesquisa, a problematização, os objetivos gerais e específicos, e, por fim, a justificativa. No Capítulo 2 está descrito o referencial teórico, onde começo abordando Engenharia de Software com seus conceitos, logo após as metodologias de desenvolvimento onde abordo sobre as tradicionais e ágeis mostrando suas diferenças e descrevendo algumas das metodologias, suas estruturas e funcionamento. No Capítulo 3 estão descritas as características do estudo explicativo, o percurso metodológico e a descrição da pesquisa. No Capítulo 4 apresento a análise dos resultados, a explanação do questionário 1 e a do questionário 2, os resultados encontrados e as soluções propostas com o planejamento atual e o proposto para a disciplina de Engenharia de Software. No Capítulo 5 apresento as considerações finais da pesquisa realizada. O capítulo 6 traz as referências utilizadas para a produção deste trabalho.

Com o avanço tecnológico cada dia mais veloz, as pessoas buscam se atualizar e utilizar o melhor que a tecnologia traz com ela, ficando cada vez mais dependente das grandes empresas que fabricam essas novas tecnologias. Grandes e pequenas empresas buscam sistemas seguros e com informações confiáveis, mas para que isso ocorra, os softwares desenvolvidos para tal finalidade devem ser bem planejados e projetados para que no futuro grandes problemas não venham a acontecer. Os novos softwares devem ser mais ágeis e de simples funcionamento, agradando o cliente e agregando valor à empresa. A grande procura por software nas grades empresas aumentou de acordo com a evolução das tecnologias e com o aumento das possibilidades que elas oferecem em diversas áreas como administração, planejamento e gerenciamento da empresa.

O que leva a confirmar essa hipótese é Oliveira (2004, p.172):

> A tecnologia hoje é fundamental nas organizações de grande porte e "é inegável que as inovações tecnológicas introduzidas nas organizações aumentaram sua produtividade, seja pelas melhorias que incorporam aos processos produtivos, seja pela racionalização da mão de obra".

E para que as tecnologias dentro das empresas deem resultado é necessário um bom planejamento no desenvolvimento das soluções. A área de engenharia de software tem produzido vários modelos de melhoria, processos, métodos e ferramentas para aumentar a

probabilidade de que os projetos de software tenham sucesso (PRESSMAN, 1997; SOMMERVILLE, 1996).

No desenvolvimento de software existe a preocupação com a qualidade e a entrega do produto, porém, é necessário existir um planejamento eficiente para que se tenha a confiança do cliente e várias metodologias e ferramentas para o gerenciamento de projetos.
Um conceito sobre gerenciamento de projeto é o de Martins (2007 p. 3):

> Genericamente "projeto" significa "empreendimento", e como tal é um trabalho que visa a criação de um produto ou a execução de um serviço específico, temporário, não repetitivo e que envolve certo grau de incerteza na realização. O trabalho normalmente é executado por pessoas que vão consumir horas, estão limitadas no prazo, custo e escopo. Como em qualquer empreendimento, as atividades precisam ser planejadas, programadas e, durante a execução, precisam ser controladas.

No gerenciamento de projeto existem vários processos e métodos que são usados no desenvolvimento de software pelas equipes, o foco nesse trabalho são as metodologias de desenvolvimento de software que podem ser tradicionais e ágeis.

Jalote (1997, p. 23), apresenta o conceito de processo como sendo o coração da engenharia de software. E conclui que um processo de software é:

> Um conjunto de atividades, ligadas por padrões de relacionamento entre elas, pelas quais se as atividades operarem corretamente e de acordo com os padrões requeridos, o resultado desejado é produzido. O resultado desejado é um *software* de alta qualidade a baixo custo. Obviamente, um processo que não aumenta a produção (não suporta projetos de *software* grandes) ou não pode produzir *software* com boa qualidade não é um processo adequado.

No curso de Licenciatura em Computação do Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano – IF Sertão – PE é ministrada a disciplina de Projeto de Desenvolvimento de Software, que utiliza em seu desenvolvimento a maioria dos conceitos vistos em Engenharia de Software e Programação Web, para que seja produzido pelos alunos um sistema utilizando os processos de desenvolvimento como nas fábricas de software, com processamento rápido do produto de software.

O problema identificado na disciplina de projeto fomentou a proposta deste projeto de pesquisa, para identificar as dificuldades vivenciadas pelos alunos desta disciplina em aplicar uma metodologia de desenvolvimento, para atender um cliente e elaborar um software, tentando com os resultados facilitar a aprendizagem dos mesmos.

A proposta da disciplina é o desenvolvimento de soluções de software para o setores do IF-Sertão pelas equipes formadas facilitando o trabalho dos funcionários do Instituto e gerando experiências para os alunos, já que são realizados todos os processos para o desenvolvimento de software formando melhor o aluno para o mercado de trabalho. Partindo desta explanação, este trabalho levanta o seguinte problema: Por que as equipes formadas para o desenvolvimento das soluções possuem tantas dificuldades na utilização das metodologias?

Com base neste questionamento, este trabalho busca subsídios para tentar identificar esse problema e melhorar o andamento da disciplina, já que com as dificuldades vivenciadas pelos alunos, o professor não consegue obter um retorno do conhecimento que foi passado na disciplina anterior, e assim as equipes no decorrer de toda a disciplina ficam bastante confusas e realizando tarefas desnecessárias perdendo tempo na realização da atividade proposta.

O objetivo geral do trabalho é identificar as dificuldades de uso das metodologias de desenvolvimento de software pelos grupos formados na disciplina de desenvolvimento de software, através de uma pesquisa realizada com os alunos, nos processos de planejamento, desenvolvimento e execução do projeto escolhido, para propor melhorias na disciplina do curso de Licenciatura em Computação.

Os objetivos específicos são:

- Acompanhar e analisar o funcionamento das fábricas de software durante o desenvolvimento de suas atividades, fazendo um comparativo com as respectivas metodologias escolhidas;
- Realizar uma pesquisa bibliográfica das metodologias escolhidas pelas equipes;
- Aplicar um questionário para os grupos;
- Identificar o nível de conhecimento dos membros das fábricas em relação as metodologias aplicadas em seus projetos;
- Propor possíveis soluções para minimizar a dificuldade destas equipes.

Um dos objetivos desta pesquisa é propor soluções para minimizar os problemas vividos pelas equipes e formar melhor o aluno para a aplicação correta dos conceitos aprendidos em Engenharia de Software.

Através desta discussão, alguns fatores serão explanados a respeito da forma de ensino em sala de aula, onde propõe a quebra do paradigma do ensino convencional já tão enraizado

em nossa cultura, que exige mudança de postura tanto do professor como do estudante, trabalhando uma forma de aprendizagem altamente investigativa e questionadora, além de uma certa maturidade profissional.

Segundo Ribeiro (2003):

> "Na educação em engenharia ainda predominam currículos tradicionais, a fraca interdisciplinaridade e a integração tardia, quando presente, entre os diferentes componentes curriculares, entre a teoria e a prática e entre o mundo escolar e o mundo profissional […] na sala de aula convencional, os alunos são vistos como receptáculos vazios a serem preenchidos por conhecimentos validados pela teoria e distribuídos pelo professor.". (Ribeiro, 2003)

Desenvolvido na educação médica na década de 70, de acordo com Ning (1995) o método PBL (*Problem Based Learning*) tem sido adaptado em um número crescente de áreas de atuação, incluindo a Engenharia, e em diferentes níveis educacionais (RIBEIRO; MISUKAMI, 2005), (GUZELIS,2006). Sob uma ótica bem simples, PBL pode ser definido como um método instrucional que usa um problema para iniciar, direcionar e motivar o aprendizado. Entretanto, o desenvolvimento de um processo efetivo para resolução de problemas é só um dos objetivos do PBL. Ribeiro e Misukami enfatizam:

> "Este método também pretende apoiar os estudantes na aquisição de uma base de conhecimento estruturada em torno de problemas da vida real e no desenvolvimento de competências e atitudes, incluindo trabalho em equipe e habilidades de auto-aprendizado, cooperação, ética e respeito aos pontos de vista de outras pessoas." (Ribeiro e Misukami, 2005)

A adoção do PBL é justificada por seus idealizadores como uma resposta à percepção dos professores de que os alunos estavam saindo do curso com muitos conceitos, mas pouca capacidade de utilizá-los e integrá-los à prática cotidiana (Barrows, 1996).

É na disciplina de Projeto de Desenvolvimento de Software que os alunos colocam em prática as teorias vistas na disciplina de Engenharia de Software. Visando entender as dificuldades vividas pelos alunos na aplicação das metodologias de desenvolvimento, esse trabalho busca expor possíveis soluções para ajudar o professor no processo de ensino-aprendizagem dos alunos.

2. REFERENCIAL TEÓRICO

2.1. ENGENHARIA DE SOFTWARE

A engenharia de software é a área que abrange vários conhecimentos a serem aplicados no desenvolvimento de software, vários conceitos sobre ela são apresentados, mas sempre se fala em qualidade e produtividade como principais características da engenharia de software, segundo Rezende (2005), Engenharia de software é metodologia de desenvolvimento e manutenção de sistemas modulares, com as seguintes características: processo (roteiro) dinâmico, integrado e inteligente de soluções tecnológicas; adequação aos requisitos funcionais do negócio do cliente e seus respectivos procedimentos pertinentes; efetivação de padrões de qualidade, produtividade e efetividade em suas atividades e produtos; fundamentação na Tecnologia da informação disponível, viável, oportuna e personalizada; planejamento e gestão de atividades, recursos, custos e data.

É possível entender com isso que a engenharia de software envolve todo o processo de desenvolvimento, desde o levantamento dos requisitos até a entrega do produto.

Para o *Institute of Eletrical and Eletronic Engineering* a engenharia de software é a aplicação sistemática, disciplinada e com abordagem quantitativa para o desenvolvimento, operação e manutenção de software (IEEE, 1990).

Segundo Martin e McClure (1991) a engenharia de software é: “o estudo dos princípios e sua aplicação no desenvolvimento e manutenção de sistemas de software [...] tanto engenharia de software como as técnicas estruturadas são coleções de metodologias de software e ferramentas...”.

Para Sommerville (2004) a engenharia de software envolve questões técnicas e não-técnicas, tais como a especificação do conhecimento, técnicas de projeto e implementação, conhecimentos dos fatores humanos pelo engenheiro de software e ainda, gestão de projetos.

Segundo Maffeo (1992) a engenharia de software é:

> “a área interdisciplinar que engloba vertentes tecnológicas e gerencial visando a abordar de modo sistemático (modular), os processos de construção, implantação e manutenção de produtos de software com qualidade assegurada por construção segundo cronogramas e custos previamente definidos”. (Maffeo)

Para vários autores a engenharia envolve também as características da gestão de projetos e não apenas os processos de desenvolvimentos que devem ser seguidos.

Para Paula Filho (2001) a engenharia de software é conexa, porém distinta, envolve múltiplas variáveis, tais como arte, atendimento das necessidades humanas, conhecimentos científicos, conhecimentos empíricos, habilidades específicas, recursos naturais, formas adequadas, dispositivos, estruturas e processos. Não deve ser confundida com a Ciência da Computação como um todo, pois ela usa resultados da ciência e fornece problemas para seus estudos.

Outras definições de Engenharia de Software costumam omitir a vertente gerencial. Concentram-se apenas nos aspecto tecnológico do problema. Os aspectos gerenciais do desenvolvimento de software devem receber uma atenção cada vez maior nessa disciplina (PRESSMAN, 2002).

2.2. METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE

As metodologias de desenvolvimento são técnicas criadas para facilitar e nortear o desenvolvedor durante todo o processo de criação do software, com isso podemos dizer que são métodos que devem ser seguidos para a produção do produto. Cada metodologia possui métodos diferentes e com isso a equipe de desenvolvimento pode escolher a que mais se adeque às suas características.

Segundo Caetano (2009, p. 1) metodologia de desenvolvimento de software é:

> As metodologias de desenvolvimento de software servem para não tornar a tarefa, complexa por natureza, um verdadeiro caos. O problema é que, dependendo do projeto, os métodos tradicionais podem deixar os desenvolvedores amarrados a requisitos desatualizados, que não correspondem às reais necessidades do cliente. Em mercados altamente competitivos, ou em momentos de crise econômica, a flexibilidade e a facilidade de mudar o rumo são qualidades muito valiosas para serem deixadas de lado. (Caetano, 2009)

Para Cisneiros et al. (2009), no decorrer de vários anos o desenvolvimento de software cresceu de forma muito acelerada, principalmente pelo fato de que as ferramentas da informática se tornaram presentes em grande parte das áreas de trabalho das pessoas. Quanto mais a informática penetrava no mercado, mais tipos de softwares tinham de ser feitos. Esta crescente demanda fez surgir uma variedade de estratégias de desenvolvimento de software. Estas estratégias têm como objetivo organizar o projeto de software e o time de

desenvolvimento, aumentando a eficácia e diminuindo os prazos e custos destes mesmos projetos de software. As metodologias de desenvolvimento trouxeram uma maior qualidade para o produto final: algo muito importante principalmente quando os softwares são mais complexos e precisam de muita organização para não se tornarem um "monstro" incontrolável. Alguns dos principais problemas encontrados no processo de desenvolvimento de software (CISNEIROS et al. (2009)):

- Caos pela mudança constante de requisitos;
- Estimativas de tempo, custo e qualidade do produto totalmente irreais;
- Os desenvolvedores não sabem exatamente como está indo o andamento do projeto e acabam mentindo ou se enganando sobre seus próprios resultados.

Metodologias de desenvolvimento de Software, Processo de Software, é o conjunto de atividades, ordenadas ou não, com a finalidade de produzir um software de qualidade. O resultado do processo é um produto que reflete a forma como o processo foi realizado. Embora existam vários processos de desenvolvimento de software, existem atividades comuns a todos eles (SOMMERVILLE, 2004):

- **Especificação do software:** definição do que o sistema deve fazer; definição dos requisitos e das restrições do software;
- **Desenvolvimento do software:** projeto e implementação do software, o sistema é desenvolvido conforme sua especificação;
- **Validação do software:** o software é validado para garantir que as funcionalidades implementadas estejam de acordo com o que especificado;
- **Evolução do software:** evolução do software conforme a necessidade de cliente.

Dentre as metodologias utilizadas nas organizações, onde há padrões próprios para o desenvolvimento de sistemas, é usual a adoção de uma metodologia que aborde o desenvolvimento de software por meio de suas etapas. Apresenta de uma maneira bastante abrangente, que poderá ser utilizado em diferentes ambientes, sistemas e metodologias, incluindo sistemas especialistas, sistemas de tempo real, protótipos etc (REZENDE, 2005).

Entre as metodologias temos duas linhas distintas, as tradicionais e as ágeis. Enquanto

as tradicionais prezam por uma quantidade excessiva de documentação as ágeis prezam por ter o software funcionando com o mínimo de documentação necessária. Portanto, adotar processos mais simplificados, como as metodologias ágeis, tem despertado um grande interesse entre as comunidades de desenvolvimento de software. Enquanto que as metodologias tradicionais têm seus processos baseados na produção de grande quantidade de documentação e de modelos para guiar o processo de desenvolvimento (FOWLER, 2001).

De acordo com Mainart e Santos (2010, p. 1):

> Com o surgimento recente de novas Metodologias de Desenvolvimento de Software – MDS, que dividiram o processo de desenvolvimento de Software, para organizá-lo, planejá-lo e facilitar o entendimento do mesmo, existem agora duas principais áreas de atuação, o Desenvolvimento chamado tradicional, com embasamento na Análise e Projeto, mantendo tudo em documentos, porém se tornando lento para mudanças, e o Desenvolvimento Ágil, baseado em código, totalmente adaptável a mudanças nos requisitos, entretanto, fraco na parte contratual e de documentos. (Mainart e Santos, 2010)

2.2.1. TRADICIONAIS

As metodologias tradicionais são divididas em partes, onde em cada uma delas existem tarefas definidas que abrangem diversas atividades a serem seguidas pelos desenvolvedores. Em cada fase da metodologia existe sempre algum material que é gerado, seja documentação ou versão do sistema, mas sempre tendo algum resultado ao final das fases.

Muitas “metodologias” pesadas são desenvolvidas no modelo em cascata o que dificulta o controle do projeto, pois a cada alteração em determinado ponto do projeto, como os requisitos, será necessário uma volta ao início do mesmo para alteração de documentação ou outro marco (PRESSMAN, 2002).

É um modelo em que as fases definidas são seguidas de maneira linear (LARMAN, 2002). Cada fase tem associado ao seu término uma documentação padrão que deve ser aprovada para que se inicie a etapa imediatamente posterior. Cada fase concluída gera um marco, que geralmente é algum documento, protótipo do software ou mesmo uma versão do sistema (NETO, 2004).

As organizações devem buscar alternativas sobre como gerenciar seus projetos de

software, visando à diminuição dos fracassos e a melhoria na qualidade de seus produtos e serviços. Segundo (Augustine et. al., 2005), os processos tradicionais de desenvolvimento e gerência de projetos (GP) têm sido caracterizados como uma descrição linear de um processo sequencial. Estes métodos podem ser efetivos para requisitos estáveis, conhecidos e consistentes. Porém, Nerur et. al., 2005, a metodologia tradicional sente falta de flexibilidade para dinamicamente ajustar-se ao processo de desenvolvimento.

Lehman (2001) diz que uma metodologia tem que atender a um ambiente complexo, imprevisível e de frequente mudança nos requisitos. Já Boehm e Turner (2004), afirmam que as organizações precisam desenvolver-se para o melhor equilíbrio entre agilidade e processos tradicionais que melhor se adaptam à sua situação.

A abordagem tradicional de gerência de projetos baseia-se em processos definidos e documentados que passam por melhorias contínuas nas diversas organizações (BOEHM & TURNER, 2003).

O planejamento detalhado e o processo disciplinado que orientam a gerência de projetos tradicionais na engenharia de software permitem a medição e controle de todas as etapas do desenvolvimento de software e da equipe do projeto, onde cada membro tem o seu papel claramente definido e os artefatos gerados em cada fase são os registros da evolução do projeto (Boehm & Turner, 2003).

2.2.1.1. RATIONAL UNIFIED PROCESS - RUP

Rational Unified Process, RUP, é um processo proprietário de desenvolvimento de software criado pela *IBM Rational Software Corporation*. O RUP é um processo bem estruturado para desenvolver software com alta qualidade de modo repetível e previsível (Kruchten, 2003).

Segundo Hermano (2003), o RUP é um conjunto de métodos e práticas de desenvolvimento voltadas a papéis e responsabilidades. Define quem faz o quê, quando, como e onde. Segundo ele, as atividades dentro do processo são bem definidas, com artefatos de entrada e saída de um processo para o outro, com dependência de ordem de execução das mesmas e com uma descrição sistemática de como devem ser realizadas. Ele diz ainda que o RUP é, na verdade, uma junção de processos, métodos e a linguagem UML[1] (*Unified*

1 Uma linguagem universal de modelagem usada para diagramar os processos de negócios e do sistema.

Modeling Language).

De acordo Kruchten (1999), o RUP é um processo de Engenharia de Software, que possui o objetivo de garantir a produção de um software de alta qualidade que satisfaça as necessidades dos usuários finais dentro de um orçamento de tempo pré-determinados. De uma maneira mais usual, o RUP é um framework de processo que pode ser adaptado de acordo com as necessidades de cada projeto.

Uma das principais características do RUP é o alto teor de customização. Porém, é algo complexo, sendo apenas recomendado para grandes equipes e grandes projetos (KRUCHTEN, 1999).

O RUP possui cinco figuras cruciais: papéis, atividades, artefatos, fluxos de trabalho (*workflows*) e disciplinas.

2.2.1.1.1. PAPÉIS

Um papel conceitua e controla o comportamento e as responsabilidades de cada um dos indivíduos ou um conjunto deles na equipe (KRUCHTEN, 2001 *apud* PINHEIRO, 2005).

Figura 1: Papéis, responsabilidades e habilidades necessárias.

Papel	Responsabilidades	Habilidades Necessárias
Analista de Sistemas[2]	Identificação de requisitos e modelagem de casos de uso.	1. Facilidade de expressão e comunicação; 2. Facilidade em construir relacionamentos; 3. Facilidade de adaptação a mudanças; 4. Iniciativa na solução de problemas e desenvolvimento de alternativas criativas; 5. Tolerância a pressão; 6. Presteza e Iniciativa; 7. Organização; 8. Proatividade e objetividade.
Arquiteto de Software	Responsável pela estrutura geral de cada visão de arquitetura. Trabalha em sintonia com o gerente do projeto.	1. Conhecimento geral e Maturidade 2. Visão e opiniões sensatas e criteriosas na falta de informações completas; 3. Liderança para conduzir o esforço técnico entre as várias equipes; 4. Tomar decisões importantes sob pressão; 5. Fazer com que essas decisões sejam cumpridas à risca; 6. Comunicação para conquistar confiança; 7. Poder de persuasão e motivação; 8. Orientação por metas, pró-atividade e foco nos resultados.
Gerente de Projeto	Aloca recursos, ajusta as prioridades, coordena interações com clientes e usuários e geralmente mantém a equipe do projeto concentrada na meta certa.	1. Análise de decisões; 2. Habilidades de apresentação, comunicação e negociação; 3. Liderança e desenvolvimento do espírito de equipe; 4. Gerenciamento de tempo; 5. Capacidade de decisão em situações de stress; 6. Bom relacionamento interpessoal; 7. Objetividade na definição e avaliação do trabalho, assegurando a participação de toda a equipe; 8. Honesto na avaliação dos resultados;
Implementador[3]	Desenvolver e testar componentes.	1. Atenção a detalhes e boa memória; 2. Capacidade de concentração; 3. Capacidade de resolver problemas práticos; 4. Disciplina; 5. Facilidade para matemática; 6. Paciência; 7. Perseverança; 8. Raciocínio lógico desenvolvido.

Fonte: http://www.oremi.com.br/artigo/arquivos/080715130256_2007WOSES.pdf

2.2.1.1.2. ATIVIDADES

Uma atividade é uma parte do trabalho que um membro da equipe executa após estar ciente de qual papel deve exercer inserido no contexto do projeto (KRUCHTEN, 2001 *apud* PINHEIRO, 2005).

2.2.1.1.3. ARTEFATO

O artefato é o produto do projeto, produzido durante o desenvolvimento do projeto. É um espaço de informação criado, modificado ou utilizado em um dado processo. (KRUCHTEN, 2001 *apud* PINHEIRO, 2005).

2.2.1.1.4. FLUXO DE TRABALHO

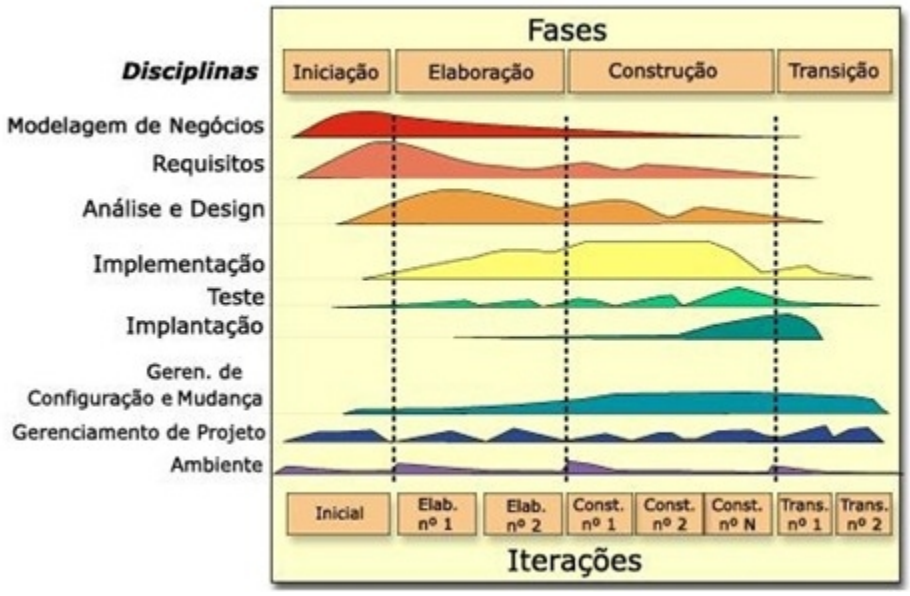
O fluxo de trabalho, por sua vez, é a determinação da sequência do desenvolvimento das atividades, após a atribuição de papéis, para que se possa ser produzido um artefato (KRUCHTEN, 2001 *apud* PINHEIRO, 2005).

2.2.1.1.5. DISCIPLINAS

A disciplina nada mais é que um conjunto de atividades inter-relacionadas que fazem parte do mesmo contexto. Ou seja, as disciplinas proporcionam uma melhor compreensão do projeto sob a visão tradicional, tornando o entendimento de cada atividade mais fácil (KRUCHTEN, 2001 *apud* PINHEIRO, 2005).

O RUP demonstra-se como uma metodologia de desenvolvimento de software que valoriza a atividade de gerenciamento. Apesar de, em algumas vezes, se tornar muito burocrático e repetitivo, o processo unificado tem por objetivo garantir o sucesso do projeto com o cumprimento de várias atividades e artefatos relacionados com a gerência de projetos. Tais atividades e artefatos são construídos por vários papéis que foram definidos com o mesmo desejo: dar suporte para realização da gerência de projetos (PINHEIRO, 2005).

Figura 2: Gráfico das baleias RUP.



Fonte: http://www.comp.ita.br/~marin/ProjetoFinal_CE235_V3.htm

2.2.2. ÁGEIS

À medida que as organizações tornam-se cada vez mais dependentes da indústria de *software*, ficam mais evidentes os problemas relacionados ao processo de desenvolvimento de sistemas: alto custo, alta complexidade, dificuldade de manutenção, e uma disparidade entre as necessidades dos usuários e o produto desenvolvido (SOMMERVILLE, 2004)

Acreditando que o processo utilizado é um dos motivos para a ocorrência desses problemas, um segmento crescente da Engenharia de Software vem defendendo a adoção de processos mais simplificados conhecidos como métodos ágeis, que visam a desburocratização das atividades associadas ao desenvolvimento (FOWLER, 2001).

A partir da década de 90, começaram a surgir novos métodos sugerindo uma abordagem de desenvolvimento ágil em que os processos adotados tentam se adaptar às mudanças, apoiando a equipe de desenvolvimento no seu trabalho. Esses novos métodos surgiram como uma reação aos métodos tradicionais de desenvolvimento de sistemas, ganhando com o passar dos anos um número cada vez maior de adeptos (BECK, 1999).

A partir de fevereiro de 2001, desenvolvedores de software de várias partes do mundo, insatisfeitos com as técnicas e métodos de desenvolvimento de sistemas usados até o momento, resolveram criar uma aliança chamada de *Agile Software Development Alliance*, mais conhecida como *Agile Alliance* (AMBLER, 2004). De acordo com Fagundes (2005), para auxiliar as pessoas a compreender o enfoque do desenvolvimento ágil, os membros da Aliança Ágil refinaram as filosofias contidas em seu manifesto em uma coleção de doze princípios, aos quais os métodos ágeis de desenvolvimento de software devem se adequar. Estes princípios são (COCKBURN, 2001):

1. A prioridade é satisfazer ao cliente através de entregas de software de valor contínuas e frequentes;
2. Entregar softwares em funcionamento com frequência de algumas semanas ou meses, sempre na menor escala de tempo;
3. Ter o software funcionando é a melhor medida de progresso;
4. Receber bem as mudanças de requisitos, mesmo em uma fase avançada, dando aos clientes vantagens competitivas;
5. As equipes de negócio e de desenvolvimento devem trabalhar juntas diariamente

durante todo o projeto;

6. Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessário para a realização do trabalho;
7. A maneira mais eficiente da informação circular dentro da equipe é através de uma conversa face a face;
8. As melhores arquiteturas, requisitos e projetos provêm de equipes organizadas;
9. Atenção contínua a excelência técnica e um bom projeto aumentam a agilidade;
10. Processos ágeis promovem o desenvolvimento sustentável. Todos envolvidos devem ser capazes de manter um ritmo de desenvolvimento constante;
11. Simplicidade é essencial;
12. Em intervalos regulares, a equipe deve refletir sobre como se tornar mais eficaz e então se ajustar e adaptar seu comportamento.

2.2.2.1. SCRUM

O *Scrum* é uma das metodologias ágeis mais usadas entre os desenvolvedores de software, é bastante conhecida por ter a intensa participação do cliente, ciclos rápidos, reuniões rápidas em todo o decorrer do projeto.

Teoricamente, a melhor maneira de acelerar o processo de desenvolvimento é construir apenas o que o cliente valoriza e nada mais. O uso exagerado de formalização – planejamento extensivamente detalhado das tarefas e documentação em excesso – pode impedir o andamento do projeto, mas o contrário, ou seja, sem a utilização de algum modo de gerenciamento, pode gerar um cenário em que os objetivos do projeto não irão ser alcançados. O uso do *Scrum* – Metodologia Ágil para Gestão de Projetos – permite desenvolver projetos bem mais adaptados à nova realidade das organizações de forma rápida. O foco do *Scrum* é descobrir uma forma que os membros da equipe trabalhem para produzir um software flexível em um ambiente de constantes mudanças (SCHWABER, 2004).

O *Scrum* orienta-se por três princípios: a visibilidade, a inspeção e a adaptabilidade (SCHWABER, 2004, p. 3). As coisas devem estar visíveis a todos os envolvidos no desenvolvimento, a inspeção deve ser uma ação corrente e, consequentemente, as ações para adaptação do produto de software devem ser realizadas.

Scrum é um processo baseado em práticas, artefatos e algumas regras simples e fáceis de aprender. Não é um processo prescritivo e não descreve o que deve ser feito em cada uma

das situações. *Scrum* é usado para trabalhos complexos, onde é impossível predizer tudo que irá acontecer. Ele oferece um *framework* e uma série de práticas que mantém tudo bem claro, trazendo para os seus praticantes o conhecimento exato do andamento do projeto e a possibilidade de realizar ajustes de forma correta para manter o projeto no objetivo desejado (SCHWABER, 2004).

Segundo Schwaber e Sutherland (2004), os principais termos da metodologia *Scrum* são:

- ***Backlog:*** Lista de todas as funcionalidades a serem desenvolvidas durante o projeto completo, sendo bem definido e detalhado no início do trabalho, deve ser listado e ordenado por prioridade de execução;

- **Sprint:** O coração do *Scrum* é a *Sprint*, um time-box de um mês ou menos, durante o qual um "Pronto", versão incremental potencialmente utilizável do produto, é criado. *Sprint*s tem durações coerentes em todo o esforço de desenvolvimento. Uma nova *Sprint* inicia imediatamente após a conclusão da *Sprint* anterior.

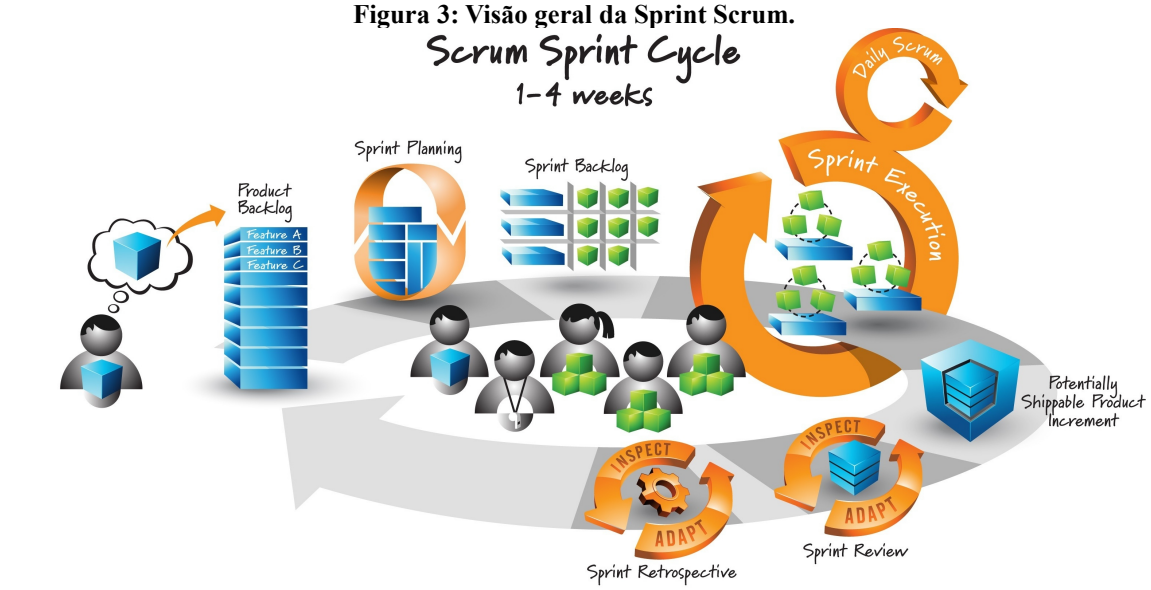
As *Sprint*s são compostas por uma reunião de planejamento da *Sprint*, reuniões diárias, o trabalho de desenvolvimento, uma revisão da *Sprint* e a retrospectiva da *Sprint*. Durante a *Sprint*:

- Não são feitas mudanças que podem afetar o objetivo da *Sprint*;
- A composição da Equipe de Desenvolvimento permanecem constantes;
- As metas de qualidade não diminuem; e,
- O escopo pode ser clarificado e renegociado entre o *Sprint* e a Equipe de Desenvolvimento quanto mais for aprendido.
-

Cada *Sprint* pode ser considerada um projeto com horizonte não maior que um mês. Como os projetos, as *Sprint*s são utilizadas para realizar algo. Cada *Sprint* tem a definição do que é para ser construído, um plano projetado e flexível que irá guiar a construção, o trabalho e o resultado do produto. *Sprint*s são limitadas a um mês corrido. Quando o horizonte da *Sprint* é muito longo, a definição do que será construído pode mudar, a complexidade pode

aumentar e o risco pode crescer. *Sprint*s permitem previsibilidade que garante a inspeção e adaptação do progresso em direção a meta pelo menos a cada mês corrido. *Sprint*s também limitam o risco ao custo de um mês corrido.

Figura 3: Visão geral da Sprint Scrum.



Fonte: http://www.mpug.com/articles/scrum-first-establish-a-firm-foundation/

- **Scrum *Master:*** O *Scrum Master* é responsável por garantir que o *Scrum* seja entendido e aplicado. O *Scrum Master* faz isso para garantir que o Time *Scrum* venha a aderir à teoria, práticas e regras do *Scrum*. O Scrum Master é um servo-líder para o Time *Scrum*. O *Scrum Master* ajuda aqueles que estão fora do Time *Scrum* a entender quais as suas interações com o Time *Scrum* são úteis e quais não são. O *Scrum Master* ajuda todos a mudarem estas interações para maximizar o valor criado pelo Time *Scrum.*

- ***Scrum Team*:** A Equipe de Desenvolvimento consiste de profissionais que realizam o trabalho de entregar uma versão usável que potencialmente incrementa o produto "Pronto" ao final de cada *Sprint*. Somente integrantes da Equipe de Desenvolvimento criam incrementos. As Equipes de Desenvolvimento são estruturadas e autorizadas pela organização para

organizar e gerenciar seu próprio trabalho. A sinergia resultante aperfeiçoa a eficiência e a eficácia da Equipe de Desenvolvimento como um todo. As Equipes de Desenvolvimento tem as seguintes características:

- Eles são auto-organizados. Ninguém (nem mesmo o *Scrum* Master) diz a Equipe de Desenvolvimento como transformar o *Backlog* do Produto em incrementos de funcionalidades potencialmente utilizáveis;
- Equipes de Desenvolvimento são multifuncionais, possuindo todas as habilidades necessárias, enquanto equipe, para criar o incremento do Produto.
- O *Scrum* não reconhece títulos para os integrantes da Equipe de Desenvolvimento que não seja o Desenvolvedor, independentemente do trabalho que está sendo realizado pela pessoa; Não há exceções para esta regra.
- Individualmente os integrantes da Equipe de Desenvolvimento podem ter habilidades especializadas e área de especialização, mas a responsabilidade pertence à Equipe de Desenvolvimento como um todo; e,
- Equipes de Desenvolvimento não contém sub-equipes dedicadas a domínios específicos de conhecimento, tais como teste ou análise de negócios.

- **Product Owner:** O *PO*, ou dono do produto, é o responsável por maximizar o valor do produto e do trabalho da equipe de Desenvolvimento. Como isso é feito pode variar amplamente através das organizações, Times *Scrum* e indivíduos. O *PO* é a única pessoa responsável por gerenciar o *Backlog* do Produto. O gerenciamento do *Backlog* do Produto inclui:
  - Expressar claramente os itens do *Backlog* do Produto;
  - Ordenar os itens do *Backlog* do Produto para alcançar melhor as metas e missões;
  - Garantir o valor do trabalho realizado pelo Time de Desenvolvimento;
  - Garantir que o *Backlog* do Produto seja visível, transparente, claro para todos, e mostrar o que o Time *Scrum* vai trabalhar a seguir e;

- Garantir que a Equipe de Desenvolvimento entenda os itens do *Backlog* do Produto no nível necessário.

O *PO* pode fazer o trabalho acima, ou delegar para a Equipe de Desenvolvimento fazê-lo. No entanto, o *Sprint* continua sendo o responsável pelos trabalhos.

O *PO* é uma pessoa e não um comitê. O *PO* pode representar o desejo de um comitê no *Backlog* do Produto, mas aqueles que quiserem uma alteração nas prioridades dos itens de *Backlog* devem convencer o *PO*.

Para que o *PO* tenha sucesso, toda a organização deve respeitar as suas decisões. As decisões do *PO* são visíveis no conteúdo e na priorização do *Backlog* do Produto. Ninguém tem permissão para falar com a Equipe de Desenvolvimento sobre diferentes configurações de prioridade, e o Time de Desenvolvimento não tem permissão para agir sobre o que outras pessoas disserem.

- **Dayling Scrum:** Reunião diária onde são avaliados os progressos do projeto e as barreiras encontradas durante o desenvolvimento;

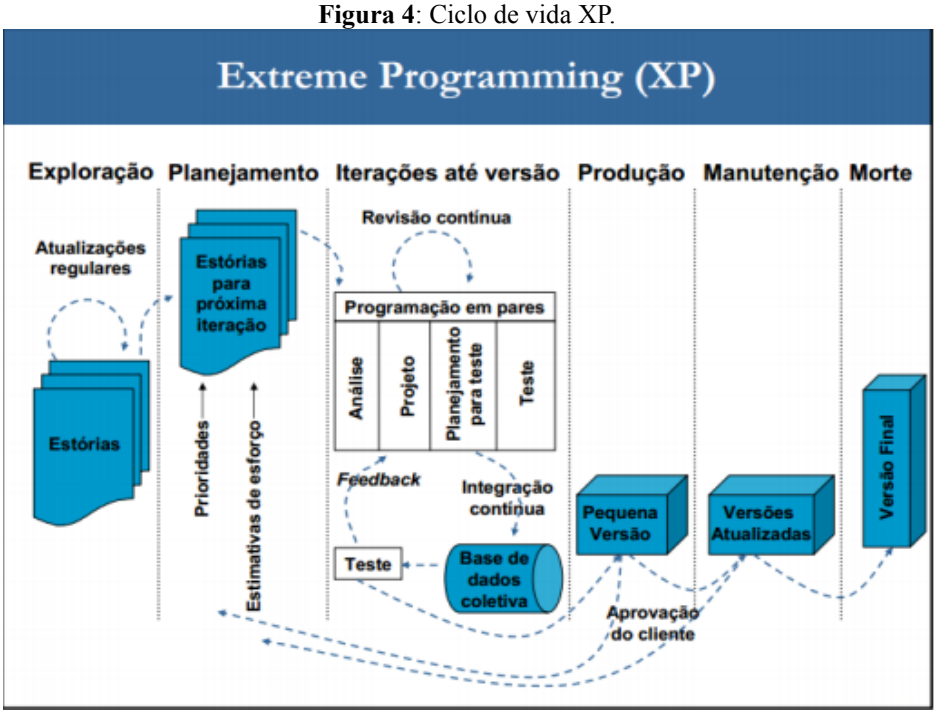
- **Sprint Backlog:** Trabalho a ser desenvolvido num *Sprint* de modo a criar um produto a apresentar ao cliente. Deve ser desenvolvido de forma incremental, relativa ao *Backlog* anterior (se existir);

- **Product Backlog:** Produção do trabalho executado.

2.2.2.2. XP

Para Kent Beck (2004), criador da *Extreme Programming*, XP é uma metodologia ágil para pequenas e médias equipes desenvolvendo software com requisitos vagos e em constante mudança. *Extreme Programming* usa times integrados de programadores, clientes, e gerentes para desenvolver software de alta qualidade em velocidade alta. Reúne também um conjunto de práticas de desenvolvimento de software já testadas, que quando estimuladas as sinergias entre elas, gerarão vastas melhorias em produtividade global e satisfação do cliente (JEFRIES, 2001).

Figura 4: Ciclo de vida XP.



Fonte: Abrahamsson (2002)

2.2.2.2.1. VALORES

A metodologia ágil XP é composta de valores que fundamentam e sustentam as boas práticas de desenvolvimento, são elas: comunicação, *feedback*, simplicidade e coragem.

- **Comunicação**

Na metodologia XP, a comunicação é um elemento fundamental no desenvolvimento de softwares (TELES, 2006).

No desenvolvimento de software um fator importante é que exista comunicação entre os desenvolvedores e todos os envolvidos no projeto para que informações não sejam passadas de maneira errada, ocasionando assim problemas futuros.

Na visão de Cavalcante (2008), os principais problemas encontrados dentro das organizações têm, como principal fator, um sistema de comunicação mal definido. Este, quando bem definido, trará maiores chances de obtenção dos resultados esperados, pois evitará possíveis desvios que dificultam o alcance dos objetivos estabelecidos.

- **Feedback**

O *feedback* é a realimentação que o cliente fornece à equipe de desenvolvimento, quando aprende algo novo sobre o sistema, quando aprende mais sobre os requisitos, ou quando aprende mais sobre a forma como foram implementados (TELES, 2006).

Para Minicucci (1995), usar o feedback é o mesmo que verificar o desempenho através da comunicação com outras pessoas e, dentro do possível, poder modificá-lo.

- **Simplicidade**

Koscianski e Soares (2007) afirmam que, na XP, a simplicidade permite a criação de softwares que não possuam funções desnecessárias, evitando-se adicionar funcionalidades que, talvez, só venham a ser importantes no futuro.

A simplicidade se torna essencial nas soluções que evitam esforços desnecessários. É mais válido gastar menos esforço produzindo uma solução simples e, depois, sofisticá-la do que criar, de imediato, implementações complexas, com funcionalidades extras (BASSI FILHO, 2008).

- **Coragem**

Ter coragem em XP significa ter confiança nos mecanismos de segurança utilizados para proteger o projeto. Ao invés de acreditar que os problemas não ocorrerão e fazer com que a coragem se fundamente nessa crença, os projetos XP partem do princípio de que problemas irão acontecer, inclusive aqueles mais temidos. Entretanto, a equipe utiliza redes de proteção que possam ajudar a reduzir ou eliminar as consequências desses problemas (TELES, 2005).

2.2.2.2.2. PRINCÍPIOS

A metodologia XP também tem definida um conjunto de princípios que a equipe de desenvolvimento deve seguir. Serão esses princípios que irão ajudar a equipe nas escolhas das soluções para possíveis problemas durante a execução do projeto. Os princípios são: retorno rápido, simplicidade, mudanças incrementais, aceitar mudanças, trabalho de qualidade.

- **Retorno rápido**

Feedback rápido (*Fast feedback*): O tempo decorrido após a implantação de uma nova funcionalidade e o *feedback* do usuário é fundamental para a aprendizagem (CASADEI; LODDI; PEREIRA; SOUZA, 2010), facilitando a correção de problemas e ajustes necessários para melhorar o uso do software.

- **Simplicidade**

Presumir simplicidade (*Assuming simplicity*): Um sistema grande e complexo é formado por partes pequenas e simples, desenvolvedores tradicionalmente são orientados a planejar para o futuro visando o reuso, a XP orienta que o esforço de desenvolvimento seja direcionado para resolver o problema atual, da forma mais rápida e simples possível (BECK, 1999).

- **Mudanças incrementais**

Mudanças incrementais *(Incremental changes*): A busca pela solução mais simples força a equipe a estar preparada para mudanças constantes. A evolução do projeto e entendimento do problema acabam gerando alterações nos requisitos e escopo, a equipe deve estar preparada para estas mudanças, executando as mesmas no menor tempo e custo possível (MACEDO, 2009).

- **Aceitar mudanças**

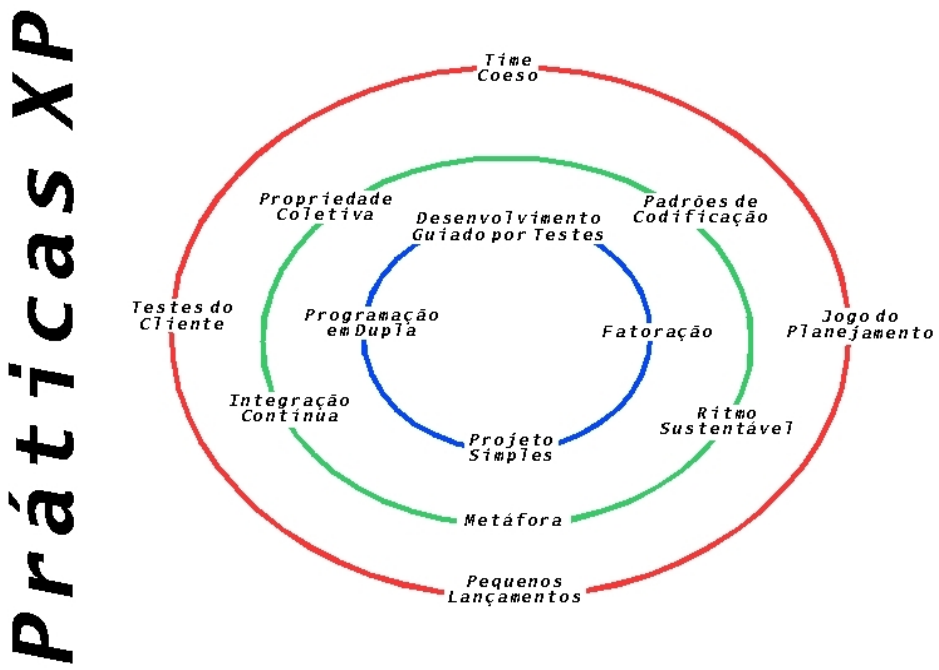
Abraçar mudanças (*Embracing changes*): Desenvolvedores geralmente não lidam bem com mudanças de requisitos ou escopo, porém estas mudanças devem ser encaradas como a oportunidade de tornar o cliente mais competitivo, entregando software que será útil e que atende as necessidades atuais.

- **Trabalho de qualidade**

Trabalho de alta qualidade (*High quality work*): Qualidade não é um fator negociável, ela deve ser uma meta, o aumento da qualidade traz mais motivação para os membros da equipe, satisfaz o cliente e facilita novas implementações (BASSI FILHO, 2008).

2.2.2.2.3. PRÁTICAS

Figura 5: Práticas XP.



Fonte: http://www.devmedia.com.br/integrando-xp-as-principais-metodologias-ageis/30989

- **Jogo de planejamento**

O objetivo do jogo de planejamento é que o cliente priorize aquilo que é importante para ele no momento. O mesmo descreve de forma sucinta as funcionalidades que ele deseja no sistema. O mesmo fica ciente do tempo e custo necessário para desenvolver cada nova funcionalidade, esta prática assegura que o trabalho seja direcionado para aquilo que é mais importante para o cliente (ASTELS; MILLER; NOVAK, 2002).

- **Pequenas versões**

Ao desenvolver um software, a equipe envolvida, geralmente, divide o projeto em partes, que ficarão sob a responsabilidade de alguns desenvolvedores. Dessa forma, um desenvolvedor se preocupa apenas com uma parte do sistema, reduzindo a complexidade a qual é exposto (TELES, 2006).

Segundo Teles (2005), com o *release*[2] curto, o cliente tem a chance de receber um feedback sobre o potencial do projeto. Dessa forma, tem a opção de decidir se continua ou

2 Um *release* representa um marco no tempo no qual um conjunto coeso de funcionalidades é finalizado e avançado para consumo de seus usuários (Teles, 2005 p. 73)

não com o projeto e se continua investindo na equipe de desenvolvimento.

- **Metáforas**

De acordo com Bassi Filho (2008), os participantes do projeto devem encontrar uma linguagem comum, que facilite o entendimento do sistema, relacionando suas abstrações com elementos do mundo real.

- **Projeto simples**

O sistema deve ser desenvolvido o mais simples quanto possível, em qualquer momento, pois a complexidade extra é removida assim que é descoberta (BECK, 1999).

Bassi Filho (2008) comprova esta situação ao dizer que as funcionalidades extras ou a flexibilidade desnecessária irão atrasar o cumprimento das tarefas essenciais e aumentarão a complexidade do sistema, dificultando, assim, a inclusão das próximas funcionalidades.

- **Testes**

Teste é a parte do desenvolvimento de software que todos sabem da importância de ser feita, porém ninguém quer fazer. Testar costuma ser considerada uma tarefa chata, que consome tempo e só é valorizada depois que o sistema entra em produção e vários problemas começam a surgir (TELES, 2006).

- **Refactoring**

Para Sato (2007), cada incremento na qualidade oferece reflexos de melhoria nas demais áreas do projeto, como a produtividade, a eficiência e a motivação. Essa prática, trazendo para a área administrativa, pode estar relacionada diretamente à qualidade. Toda vez que surgir uma nova ideia que possa vir a melhorar uma ideia antiga, trata-se do refactoring. Este permite a eficiência nos processos e nos resultados.

- **Programação em pares**

A programação em par é uma técnica, na qual, dois programadores trabalham em um mesmo problema, ao mesmo tempo, e em um mesmo computador. Dessa forma, enquanto uma pessoa (o condutor) assume o teclado e digita os comandos que farão parte do

programa, a outra (navegador) a acompanha fazendo um trabalho de estrategista. A programação em par tem o objetivo de fazer com que as correções sejam aplicadas imediatamente, assim que os erros apareçam (TELES, 2006).

- **Propriedade coletiva**

A equipe como um todo é responsável pelos códigos gerados, não possuindo donos.

Nessa mesma linha de pensamento, Teles (2006) trava uma batalha permanente a esta forma de trabalho. Não existe uma pessoa responsável pelo código. Cada desenvolvedor tem acesso a todas as partes do código e total liberdade para alterar o que necessitar, ou seja, o código é coletivo e pertence a todas as pessoas que fazem parte da equipe.

- **Cliente presente**

Na tecnologia, de acordo com Koscianski e Soares (2007), é fundamental a participação do cliente durante todo o processo de desenvolvimento de um projeto. Ele deve estar sempre disponível para tirar suas dúvidas, evitando assim atrasos e construções errôneas.

- **Padrão de código**

Os diferentes estilos de códigos costumam estar presentes entre os programadores de qualquer equipe de desenvolvimento e tais diferenças dificultam a compreensão do código por todos os membros da equipe. Isso pode ser evitado, se todos os programadores adotarem um mesmo estilo de codificação (TELES, 2006).

- **Integração contínua**

Logo após terminar determinada rotina, o programador deve integrá-la ao projeto principal. Isso deve ser feito várias vezes ao dia, visando a sincronização das atividades individuais (BECK, 2004).

- **Semana de 40 horas**

Pessoas cansadas não conseguem aplicar a metodologia XP, o respeito pelos limites físicos e necessidades individuais é essencial para a produção de um bom trabalho. A XP recomenda que a carga horária de trabalho não ultrapasse oito horas diárias e quarenta horas semanais (GONÇALVES JR., 2009).

2.3. GERENCIAMENTO DE PROJETOS

O gerenciamento de um projeto de software é uma tarefa complicada e envolve condições adversas, além de fatores externos que podem ser imprevisíveis. Podem-se citar algumas entre essas condições como mudanças de tecnologias e as constantes renovações dos requisitos do produto por parte do cliente (SCHWABER, 2004).

Segundo o *Project Management Institute*, PMI (2001) um projeto pode ser definido como um esforço temporário para criar um produto ou serviço único e o gerenciamento de projetos pode ser definido como a arte de coordenar atividades com o objetivo de atingir as expectativas dos stakeholders.

Para DINSMORE (1992), Gerência de Projetos é a combinação de pessoas, técnicas e sistemas, necessários à administração dos recursos indispensáveis ao objetivo de atingir o êxito final do projeto.

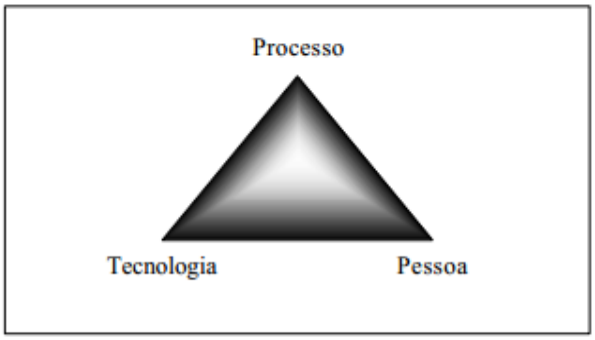
Segundo LEWIS (2000), as organizações que praticam um gerenciamento de projetos com métodos sólidos detêm uma vantagem competitiva sobre as outras.

A falha nos projetos é discutida em algumas pesquisas, como a apresentada por Curtis e Statz (1996) onde, em 1992 e 1993, mais de 60% dos projetos pesquisados nos Estados Unidos (EUA) estavam atrasados e mais da metade ultrapassava em 50% o prazo planejado. Um estudo conduzido em 1999 por Gordon (1999) indicou que somente 37% dos Sistemas de Informação (SI) foram finalizados no prazo estipulado. Adicionalmente, dos 63% dos projetos que atrasaram, 42% seriam finalizados acima do orçamento. O Standish Group, através de um estudo chamado de relatório do “Chaos” (STANDISH, 1995), identificou que as empresas dos Estados Unidos gastaram US $81 milhões em projetos de software que foram cancelados em 1995, sendo que 31% dos projetos de software estudados foram cancelados antes de estarem concluídos, 53% excederam mais do que 50% a sua estimativa de custo e somente 9% dos projetos, em grandes empresas, foram entregues no tempo e orçamento previstos; para empresas de pequeno e médio porte, os números melhoraram de 50% para 28% e de 9% para 16%.

O problema pode estar no fato que modelos de melhoria estão baseados no Triângulo Mágico da Força do Desenvolvimento de Software (*Magic Triangle of Software Development Greatness*) apresentado na Figura 6 que são importantes, mas não são determinantes, para o

sucesso do desenvolvimento de software. O sucesso do projeto está baseado em oportunidades e benefícios e em riscos. Oportunidades e benefícios tratam o valor do produto a ser entregue; e riscos tratam das incertezas de se obter o produto dentro do custo, tempo, esforço e qualidade estimado. Qualquer deficiência em alguma área do triângulo (Figura 6) se manifestará nos riscos do projeto (LISTER, 1997).

Figura 6: Triângulo mágico da força do desenvolvimento de software.



Fonte: http://www.inforede.net/Technical/Business/IT/PM%20Risco%20em%20Desenvolvimento.pdf

Pelo exposto anteriormente, fica evidente que projetos de software lidam com um alto nível de incertezas, oriundas das mais diversas fontes, dentre elas, a mudança de tecnologia, a imaturidade nos processos, a adequação do perfil técnico para exercer uma atividade e a mudança contínua de requisitos. As incertezas são uma fonte de risco em potencial.

Segundo BOENTE (2003), tratando-se do assunto projetos ou processos, tanto o planejamento quanto o gerenciamento são fundamentais. No mundo globalizado as empresas necessitam de projetos para alcançar ou superar suas expectativas.

3. PESQUISA EXPLICATIVA

A pesquisa do ponto de vista dos objetivos trata-se de um estudo explicativo. Segundo Gil (2010) a pesquisa é explicativa quando o pesquisador procura explicar os porquês das coisas e suas causas, por meio do registro, da análise, da classificação e da interpretação dos fenômenos observados. Visa identificar os fatores que determinam ou contribuem para a ocorrência dos fenômenos; “aprofunda o conhecimento da realidade porque explica a razão, o

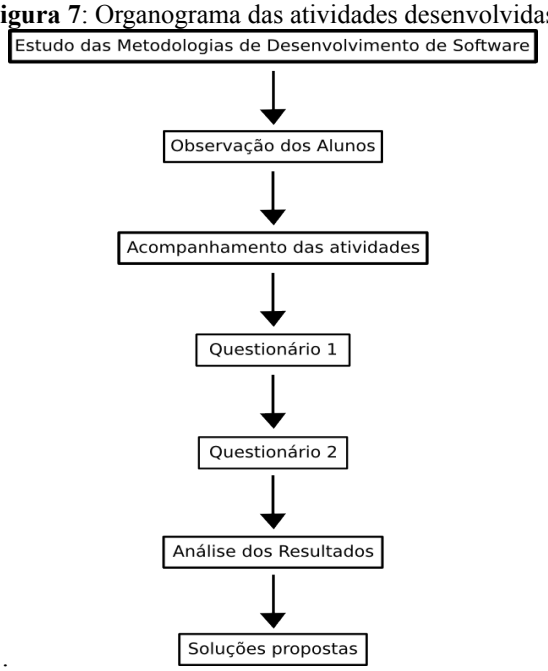
porquê das coisas.” (GIL, 2010, p. 28).

Do ponto de vista dos procedimento a pesquisa se enquadra no estudo de caso, quando envolve o estudo profundo e exaustivo de um ou poucos objetos de maneira que permita o seu amplo e detalhado conhecimento (YIN, 2001).

De acordo com isso, na pesquisa primeiramente foi realizado um estudo sobre as metodologias de desenvolvimento de software e realizado também juntamente com os alunos da disciplina de projeto de desenvolvimento de software uma pesquisa para levantamento de dados. Logo após, a obtenção e análise dos resultados foram propostos uma nova opção de metodologia para a realização da disciplina de Engenharia de Software, que é fundamental para o domínio das metodologias de desenvolvimento de software.

3.1. PERCUSO METODOLÓGICO

Figura 7: Organograma das atividades desenvolvidas



Fonte: Elaborada pelo autor

1 – Estudo das Metodologias de Desenvolvimento de Software: O estudo das metodologias é necessário para a obtenção de um maior de conhecimento sobre as mesmas, e suas diversas práticas, facilitando assim o acompanhamento dos alunos, onde é analisar melhor as ações tomadas por eles.

2 – Observação dos alunos: Ocorre para que seja possível observar de vários ângulos

os diversos fatores que ocorrem nas ações das equipes ao escolher as metodologias de desenvolvimento e sua utilização. É possível com a observação chegar a várias conclusões através da análise das equipes.

3 – Acompanhamento das atividades: Com o acompanhamento das atividades realizadas pelas equipes é possível identificar a maneira que os alunos estão produzindo o produto a eles proposto e de que forma utilizam as metodologias de desenvolvimento, acompanhando cada prática que a equipe desenvolve.

4 – Questionário 1: Foi desenvolvido para avaliar o conhecimento dos alunos em relação as metodologias escolhidas, o mesmo foi realizado com a intenção de obter dados qualitativos e quantitativos.

5 – Questionário 2: Foi desenvolvido para avaliar como foi o uso das metodologias pelas equipes, o mesmo foi realizado na intenção de obter dados quantitativos e qualitativos quanto a utilização das metodologias no decorrer do projeto pelas equipes.

6 – Análise dos Resultados: Foi possível coletar e analisar os resultados obtidos com os alunos das equipes formadas na disciplina durante todos os processos anteriormente realizados.

7 – Considerações finais: Após todos procedimentos de observação, acompanhamento, aplicação dos questionários e análise dos resultados foram propostas possíveis soluções para amenizar as dificuldades vividas pelos alunos.

3.2. DESCRIÇÃO DA PESQUISA EXPLICATIVA

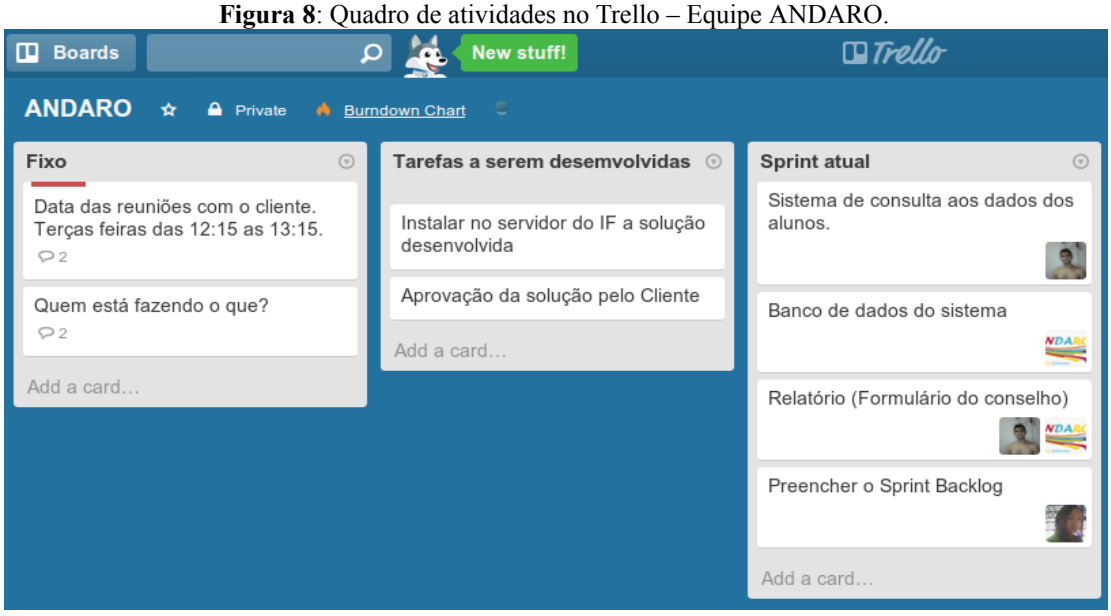
Após a escolha do tema de pesquisa e área a ser explorada foi realizado um estudo sobre as Metodologias de Desenvolvimento para que fosse possível ter um maior entendimento dos processos, maneiras de utilização e principais características. Nesse estudo foi possível conhecer mais a fundo as metodologias tradicionais e ágeis, mas para isso foi realizada uma pesquisa entre essas para identificar quais eram as mais utilizadas pelas equipes de desenvolvimento evitando assim que houvesse o estudo de uma metodologia ultrapassada. O foco foi maior nas metodologias ágeis, que são as mais utilizadas pelas equipes nas fábricas criadas na disciplina de projeto de desenvolvimento de software.

Através disso foi realizado um estudo Bibliográfico, conhecendo os principais autores e melhores conceitos das metodologias, gerenciamento de projetos e Engenharia de software.

Posteriormente foi realizada a observação com as equipes em sala de aula, as aulas ocorriam todas as sextas-feiras por um período de 3 horas, dessa maneira era possível observar melhor a equipe já que com todo esse tempo a maioria das atividades eram realizadas em sala. Na observação foi analisado o comportamento dos grupos e as maneiras utilizadas por eles para a escolha das metodologias a serem usadas no desenvolvimento dos projetos. Dessa forma foi possível observar quais os critérios que alguns utilizavam para encontrar a melhor metodologia que se adaptasse ao desenvolvimento da sua equipe.

Durante todo o período de observação foi realizado também o acompanhamento das atividades junto as equipes. Após a escolha do cliente que receberá o sistema desenvolvido pelos alunos, existem reuniões para um melhor entendimento do que deve ser criado. E de acordo com a metodologia escolhida pelas equipes, o cliente participava ativamente no processo de desenvolvimento, com isso várias reuniões aconteciam apenas com a equipe, dessa forma era possível realizar o acompanhamento das atividades. Para que a professora da disciplina pudesse acompanhar também o desenvolvimento das equipes era utilizado a ferramenta on-line Trello. O Trello é uma plataforma de gestão de projetos e/ou tarefas com diversos recursos que facilitam a interação dos participantes e aumentam a produtividade do trabalho em equipe.

Figura 8: Quadro de atividades no Trello – Equipe ANDARO.



Fonte: https://trello.com/

Para avaliar o conhecimento dos alunos em relação as metodologias escolhidas, e

quais os critérios utilizados por eles em realizar essas escolhas, foi desenvolvido e realizado um questionário quantitativo e qualitativo para a obtenção de dados, pois através dele era possível identificar se os alunos obtinham o conhecimento teórico e mais profundo das metodologias escolhidas. A pesquisa foi realizada com os alunos do 7º período no segundo semestre de 2013 e no primeiro semestre de 2014. Ao realizar a aplicação do questionário era solicitado aos alunos que respondessem sem o auxilio de nenhum material (internet, livros, cadernos) que pudessem ajudá-los a desenvolver sua resposta, pois o que seria analisado na pesquisa seria justamente o nível de conhecimento dos mesmos.

Após esse procedimento ocorreram 5 semanas de observação onde as equipes realizaram o desenvolvimento do projeto, logo, foi realizado um segundo questionário que tinha como objetivo identificar se as equipes utilizaram de maneira correta as metodologias escolhidas, e avaliar se realmente as definições colocadas por eles no questionário anterior foram compatíveis com as informações respondidas.

Com todos os dados já disponíveis, foi possível realizar a análise e obter os resultados das pesquisas.

Com os resultados gerados, foram pensadas em possíveis soluções para os problemas encontrados.

4. ANÁLISE E RESULTADOS

Foram coletadas 20 respostas nos dois questionários propostos aos alunos. O questionário 1 foi aplicado na turma após as equipes terem escolhido as metodologias que iriam utilizar no desenvolvimento, já o questionário 2 foi realizado no período de finalização do produto gerado, dessa maneira foi possível analisar o porque da escolha deles já que escolheram as metodologias que melhor se adaptavam e se conseguiram utilizá-las da forma que a mesma é descrita.

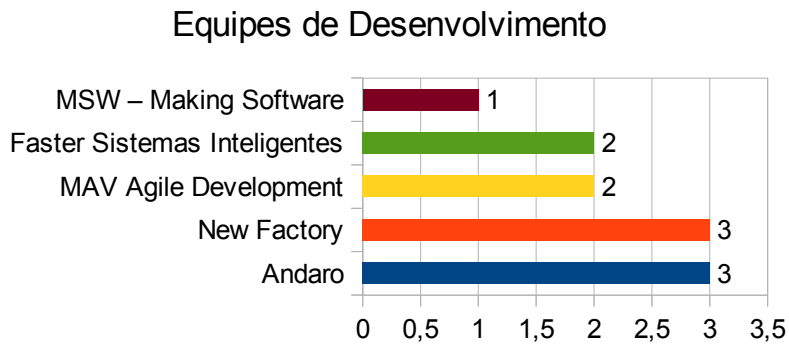
A primeira pergunta refere-se ao nome da fábrica que o aluno pertence, foi criada na intenção de identificar qual metodologia foi escolhida por cada equipe e assim conseguir acompanhar os alunos nas atividades de acordo com a metodologia escolhida, podendo facilitar a identificação do nível de conhecimento de todos da equipe, já que será possível separar as respostas pelas fábricas.

4.1. QUESTIONÁRIO 1

A aplicação do questionário 1 ocorreu antes das equipes iniciarem o desenvolvimento, acontecendo no momento das escolhas das metodologias de desenvolvimento dos seus projetos.

1 - Qual nome da sua empresa?

Figura 9: Gráfico da relação das equipes entrevistadas e a quantidade média de alunos por equipe



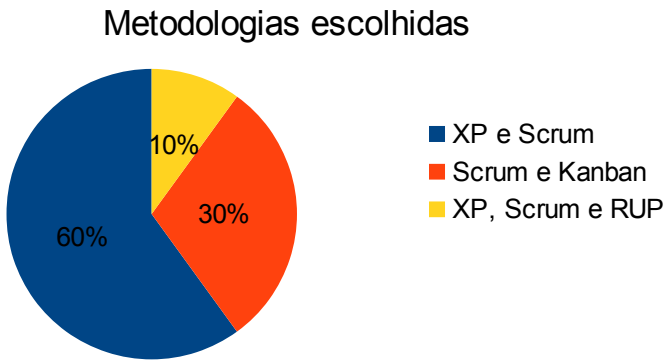
Fonte: Elaborado pelo autor

Segundo Cohn (2011), uma pesquisa realizada em mais de sete mil projetos aponta que o tamanho ideal para uma equipe para obter maior produtividade é de cinco a sete pessoas. Logo, este também é um dos princípios do *Scrum*, que recomenda equipes pequenas com sete pessoas, podendo variar em mais ou menos duas pessoas. As pequenas equipes maximizam a comunicação e minimizam a supervisão. Com a figura 9 é possível ver que a média de alunos formadas na disciplina de projeto eram pequenas, facilitando assim o desenvolvimento dos processos.

As equipes formadas na disciplina de projeto foram formadas por no máximo 4 alunos, o que garante um melhor desenvolvimento do sistema, garantindo uma maior interação entre todos facilitando o planejamento das atividades e a organização do projeto.

2 - Quais as metodologias de desenvolvimento foram escolhidas pela sua equipe para o projeto?

Figura 10: Metodologias utilizadas pelas equipes



Fonte: Elaborado pelo autor

Na figura 10 as equipes foram questionadas em relação a quais metodologias de desenvolvimento escolheram para o uso no projeto.

Com as informações vistas na figura 10 que com 60% das escolhas nas metodologias XP e *Scrum* é possível afirmar que estas são as mais usadas, Painka e Marchi (2013) também afirmam isso. As metodologias ágeis surgiram como uma alternativa as metodologias tradicionais, visando a agilidade no desenvolvimento de softwares, onde é possível fazer a incorporação de modificações que venham a ocorrer no decorrer do projeto. As metodologias ágeis mais usadas atualmente são a *Scrum* e o XP (*Extreme Programming*).

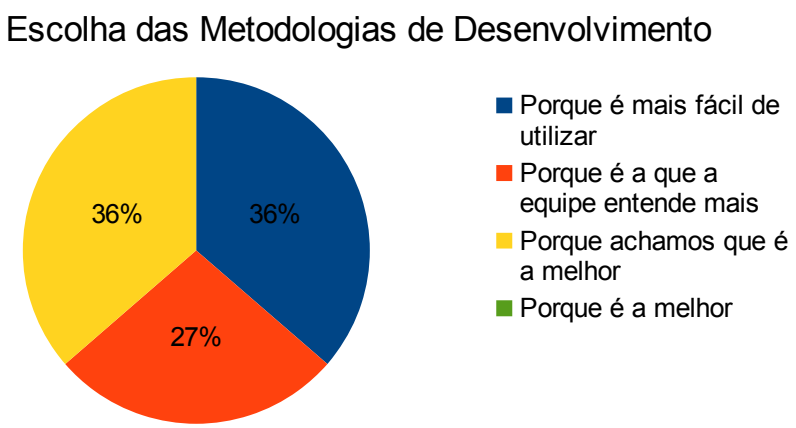
Logo após com 30% a escolha do *Scrum* e Kanban[3], sendo utilizado o Trello como quadro de atividades pela equipe.

Com 10% o uso do XP, *Scrum* e RUP que é uma metodologia tradicional de desenvolvimento de software.

3 É uma metodologia de sinalização para controle de fluxo de operação aplicada em indústria de forma geral. Stefanini (2013).

3 - Porque essas metodologias foram escolhidas?

Figura 11: Escolha das metodologias



Fonte: Elaborado pelo autor

Essa questão tem como objetivo entender os critérios utilizados nas escolhas das metodologias pelas equipes.

Dos alunos entrevistados 36% afirmam que as metodologias foram escolhidas "porque são mais fáceis de utilizar", outros 36% "porque são melhores no desenvolvimento do projeto" e por fim 27% "devido ao domínio" da metodologia escolhida.

Na resolução do questionário quando a resposta sobre a escolha das metodologias fosse "Porque achamos que é a melhor" ou "Porque é a melhor", foi obrigatória a explicação da opção escolhida. Das duas opções apenas a "Porque achamos que é a melhor" foi escolhida, obtendo um total de 36%. Apenas 4 alunos responderam sobre esta opção, veremos a seguir duas delas:

Resposta do aluno A ao achar que a metodologia era a melhor alternativa: "Eram as mais faladas por pessoas que participavam de fabricas de softwares.".

O aluno B afirma que: "De acordo com o que foi visto na disciplina de engenharia de software o *Scrum* é uma metodologia ágil que pode ser bem adaptados a vários projetos."

Em outras respostas no mesmo questionário foram encontradas respostas diferentes de alunos da mesma equipe, afirmando a falta de consenso em relação as escolhas.

Partindo dos resultados das figuras 10 e 11 é possível concluir que 90% das escolhas foram focadas nas metodologia ágeis que tem como definição, o desenvolvimento rápido dos projetos e comprovando assim o que diz (COCKBURN, 2001) referenciado na seção 2.2.2. deste trabalho.

5 - Tiveram dificuldade na escolha das mesmas? Porque?

A questão acima busca identificar possíveis dificuldades vividas pelos alunos em cada equipe na escolha das metodologias que seriam utilizadas.

No momento de reunião com o grupo para a decisão de um passo tão importante no projeto, várias dúvidas surgem devido ao grande número de metodologias que existem e com estruturas bem diferenciadas, nessa questão tiveram várias contradições nas respostas das equipes.

Segundo Perboni:

> "Ao adotar uma nova metodologia, uma organização deverá enfrentar os mesmos desafios associados à adoção de novas tecnologias. Em outras palavras, ao adotar uma metodologia ágil, a organização deve se adaptar aos novos princípios e práticas ao mesmo tempo em que a metodologia também é customizada para se adaptar à organização. Dentre as principais observações nesse sentido, quanto "menor a distância" entre a cultura organizacional e os pressupostos culturais implícitos no conjunto de práticas de uma metodologia, mais fácil será o processo de adoção.". Perboni (2013, p. 01)

O aluno C responde sobre as dificuldades em escolher a melhor metodologia:

> "Sim, a equipe teve dificuldade em escolher a metodologia e definir como seria o processo. Acredito que isso ocorreu devido a pouco entendimento prático e pouca assimilação de como funciona o processo de desenvolvimento de software de uma fábrica de software. Apesar do conhecimento teórico, compreendemos que aplicar as metodologias envolve muita organização e conhecimento das mesmas por parte de toda a equipe.".

"Não. Porque os conceitos são simples e de fácil entendimento, bastando apenas poucas horas de estudo.". Resposta do aluno D, que afirma que a metodologia escolhida pela equipe é de fácil utilização.

Resposta do aluno E sobre a escolha da sua equipe:

> "Em parte, a escolha da metodologia que a empresa irá usar é algo que irá influenciar na produtividade da mesma e na forma da equipe trabalhar, sendo que precisa ser algo objetivo gerando, relatórios, produtividade e que tenha também reuniões para poder ser discutido alguma dificuldade ou o andamento do trabalho.".

Com a variação das respostas doa alunos, é possível perceber as dúvidas existentes em todo o processo de escolha das metodologias.

6 - Como tiveram conhecimento sobre a metodologia escolhida?

Nessa questão teve-se a intenção de identificar de que forma os alunos conheceram as metodologias.

Na seção 2.2. deste trabalho Martin e McClure (1991) expõem os conceitos da Engenharia de Software e mostram a estruturação da mesma.

A disciplina de Projeto de Desenvolvimento de Software acontece no 7º período do curso, anteriormente no 5º período é cursada pelos alunos a disciplina de Engenharia de Software onde são destacadas as principais metodologias de desenvolvimento e suas características.

As maioria das respostas dos alunos para esse questionamento foi de justamente ter conhecimento sobre as metodologias por ter aprendido na disciplina de engenharia de software, segue abaixo algumas respostas.

"Através da disciplina de engenharia de Software tivemos que pesquisar acerca desta metodologia.". Resposta do aluno F em relação a questão 6.

"Através das aulas de engenharia de software, sendo assim estamos colocando boa parte do que aprendemos nas aulas de engenharia em pratica. Sendo também discutida entre a equipe." Resposta do aluno G.

7 - Fale um pouco sobre elas.

Essa pergunta te por objetivo analisar o que os membros das equipes conhecem de maneira teórica sobre as metodologias, dessa forma é possível compreender o real conhecimento dos alunos.

Os resultados dessa questão que foram explanados pelos alunos mostravam sempre as características básicas das metodologias, em algumas respostas era perceptível a falta de conhecimento sobre as escolhas realizadas por eles.

Resposta do aluno H a respeito das metodologias escolhidas:

> "A XP é uma metodologia ágil para pequenas e médias equipes. O ponto chave é desenvolver softwares com especificações abertas, e que deve sofrer várias mudanças. Trabalha em cima de 4 pontos: Planejamento, projeto, codificação e testes. O *Scrum* necessita, sempre que possível, de reuniões diárias com os stakeholders (todos os envolvidos no processo). Cada ação não vai ser descrita nessa metodologia. Se enquadra melhor com trabalhos complexos que seria impossível descrever todo o processo. Acho que é isso".

Resposta do aluno I:

> "O *Scrum* é uma metodologia bastante utilizada atualmente é um método ágil, todas as opiniões dos integrantes da equipe são levadas em consideração e onde todos os membros da equipe podem ter uma visualização de como está o projeto através das reuniões dos *Sprint*s que são feitos a cada 30 dias e o daily *Scrum* feito em 15 dias e com uma reunião de uma hora. No final deste ciclo será entregue uma implementação do sistema, vale lembrar que estes prazos podem ser adaptados para aumentar ou para diminuir, isto irá depender totalmente do empenho de nossos programadores que se empenharão ao máximo em cada projeto por nós desenvolvido.".

"Sinceramente no momento não consigo falar sobre as metodologias pois não estou lembrando sobre seus procedimentos.". Resposta do aluno J, o mesmo não consegue responder sobre a metodologia utilizada em sua equipe.

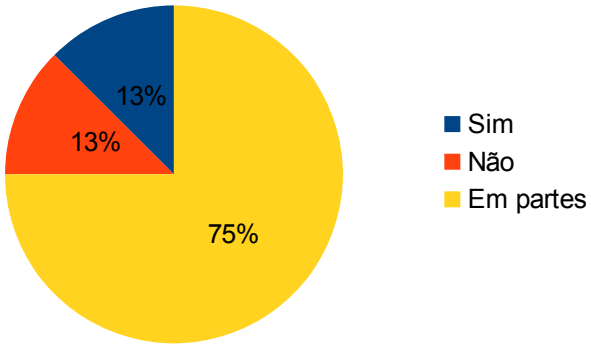
"Srcum - Reuniões de 15 minutos em pé, divide o problema em partes e vai incrementando até o resultado. XP - não sei. ". Resposta do aluno K, informando apenas poucas características de uma das metodologias.

4.2. QUESTIONÁRIO 2

O questionário 2 foi aplicado após o desenvolvimento das soluções pela equipes, tentando identificar se a utilização das metodologias ocorreu da maneira correta.

1) As metodologias escolhidas pela equipe, foram utilizadas da maneira correta?

Figura 12: Uso correto das metodologias



Fonte: Elaborado pelo autor

Essa questão busca saber se os alunos das equipes utilizaram as metodologias de desenvolvimento de maneira correta, cumprindo os processos propostos pela metodologias.

Nesse quesito 75% dos 11 alunos entrevistados informaram que o uso ocorreu em partes, onde na observação foi possível perceber que muitos fatores importantes acabaram não sendo cumpridos, deixando as regras propostas realmente de lado.

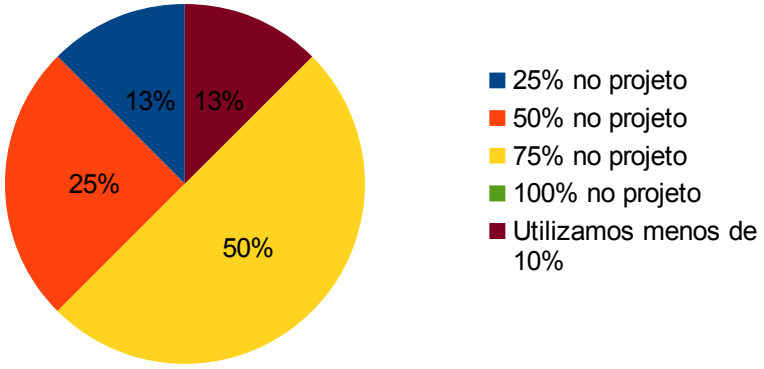
Em seguida, 13% informou que utilizou da maneira correta seguindo todos os métodos e processos e 13% informou que não utilizou de maneira correta.

Segundo Bassi (2008) Quando o desenvolvimento do software não cumpre o plano inicial, uma cadeia de eventos pode ser prejudicada e isso se torna motivo de estresse e pressão sobre a pressão sobre a equipe de desenvolvimento.

Pode-se afirmar a partir desse conceito que o replanejamento feito após a definição dos processos pode atrapalhar o desenvolvimento das equipes gerando assim as dificuldades a durante o projeto.

2) Em relação ao uso total das metodologias escolhidas pela sua equipe, você classificaria que utilizou:

Figura 13: Uso total das metodologias escolhidas



Fonte: Elaborado pelo autor

Ao utilizar metodologias de desenvolvimento em um projeto, muitas coisas podem ser adaptadas de acordo com a necessidade do cliente ou até mesmo da equipe, dessa maneira é possível identificar na figura 13 que nenhum dos entrevistados afirmou usar totalmente as metodologias como elas são, mas 50% dos alunos garantem utilizar grande parte das mesmas, adaptando assim a metodologia para o uso da equipe.

Outros 25% dos alunos afirmaram utilizar metade da metodologia definida durante o seu projeto. 13% afirmou utilizar 25% da metodologia em seu desenvolvimento e outros 13% afirmou utilizar menos de 10% da metodologia escolhida durante todo o projeto.

3) Durante a execução do projeto houve algum problema no uso das metodologias?

Essa questão tem como intenção saber dos alunos se problemas ocorreram no decorrer do projeto e identificar de que forma podem ser sanados.

As equipe são pequenas e com isso existe uma maior comunicação e a probabilidade de erros e problemas são menores pois ao estarem todos sempre a parte do que está ocorrendo no projeto a resolução acontece de maneira mais rápida.

Alguns dos problemas relatados pelos alunos fora os descritos abaixo:

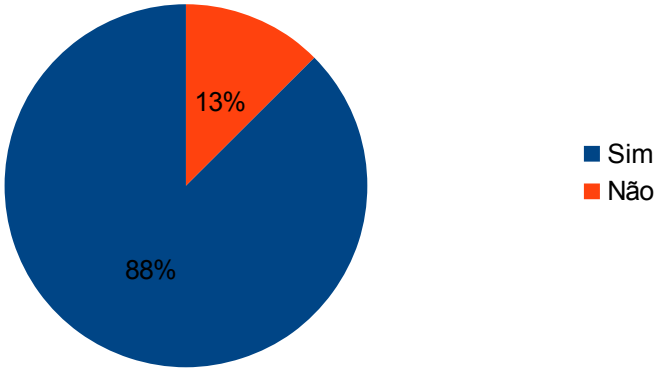
“O problema foi a não aplicação das metodologias corretamente.”. Resposta do aluno L ao ser questionado sobre os problemas no uso das metodologias.

“De inicio as metodologias não estavam sendo seguidas, mas, depois foi visto a importância de seguir os métodos para facilitar e agilizar o projeto e ficar atualizado com o cronograma.” Resposta do aluno M, afirmando a importância do uso correto das metodologias.

“Houve sim, em relação a confusão que fizemos, pois escolhemos uma metodologia e aplicamos em parte outra. Não estava condizente”. Resposta do aluno N.

4) Houve a adaptação das metodologias no decorrer do projeto?

Figura 14: Adaptação das metodologias de desenvolvimento

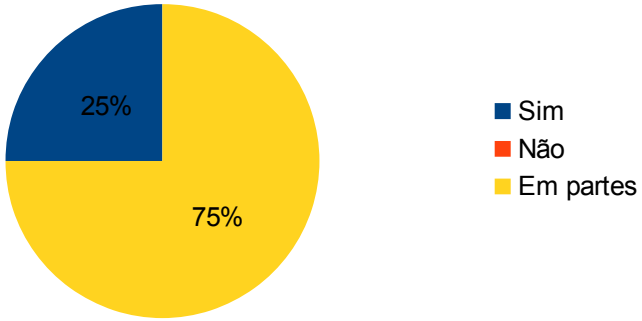


Fonte: Elaborado pelo autor

Ao iniciar o projeto de desenvolvimento os alunos escolhem as metodologias que irão utilizar durante todo toda a realização do produto. Ao escolher a metodologia, a equipe concorda em utilizar os métodos e processos definidos pela mesma, mas a figura 14 mostra que 88% dos entrevistados fizeram a adaptação das mesma no uso com as equipes, utilizando apenas as características que achavam mais importantes. Outros 13% afirmam o uso completo e sem adaptações dos processo escolhidos.

5) O processo definido pela equipe para o desenvolvimento do projeto foi seguido?

Figura 15: Uso correto do processo de desenvolvimento



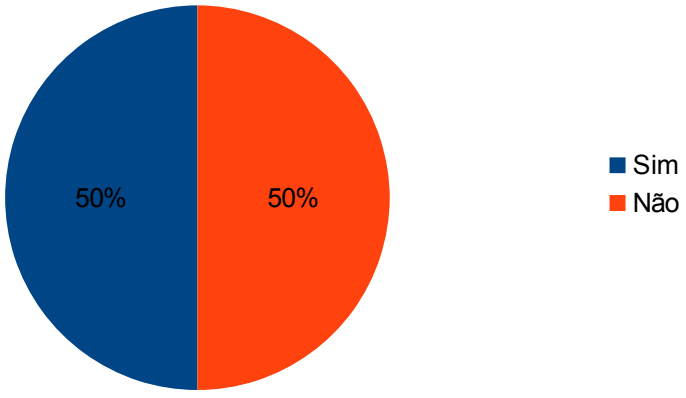
Fonte: Elaborado pelo autor

Um processo de software pode ser definido como um conjunto coerente de atividades, políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, dispor e manter um produto de software (FUGGETTA, 2000).

A questão busca saber se os processos definidos pelas equipes após a escolha das metodologias foram utilizadas da maneira correta. Antes de iniciar o desenvolvimento os alunos listam as atividades, procedimentos e ferramentas que irão utilizar. De acordo com o questionamento feito, é possível analisar na figura 15 que apenas 25% afirmam ter seguido de forma correta os processos definidos. Os outros 75% afirmam que seguiram em parte os processos definidos.

6) Houve a mudança de metodologia no decorrer do projeto?

Figura 16: Mudança de metodologia



Fonte: Elaborado pelo autor

De acordo com os resultados da figura 15 é possível identificar que várias mudanças inesperadas ou até mesmo esperadas podem ocorrer no projeto, dessa maneira muitas equipes escolhem as metodologias e acabam não utilizando o que a mesma define e pontua, ocorrendo a necessidade de adaptação de alguns pontos ou ate mesmo a mudança para outra metodologia.

A figura 16 mostra perfeitamente que 50% dos entrevistados afirma que houve a mudança na escolha das metodologias, e outros 50% afirmam que continuaram com a mesma do início ao fim do projeto.

7) Caso a resposta anterior tenha sido SIM, responda. Qual metodologia foi escolhida para substituir a outra?

A questão busca identificar caso a resposta anterior sobre a mudança de metodologia tenha sido positiva, qual metodologia foi inserida no projeto após algum tempo de desenvolvimento.

O aluno O respondeu da seguinte maneira: “Retiramos o XP, e trabalhamos com Kanban e *Scrum*”.

Comentários!!

O espaço de comentário foi deixado caso o aluno se sentisse a vontade para explanar qualquer opinião sobre a pesquisa, o projeto ou a disciplina. Alguns comentários importantes fora:

“O maior problema de fato foi a perda de tempo que se deu em readequações das metodologias. Até de fato depois de muito caminhar chegar a um resultado.”. O aluno P, relata as dificuldades encontradas no uso das metodologias.

“A disciplina é excelente para adquirir experiência sobre a forma e como um projeto deve ser feito, com todas as vantagens e desvantagens. ”. O aluno Q, cita a importância da disciplina na obtenção de experiência para o mercado de trabalho.

“No início estávamos perdidos, porém começamos a trabalhar ordenadamente com as tecnologias.”. O aluno R, mostra a dificuldade de organização da equipe no início do projeto.

4.3. RESULTADO ENCONTRADOS

No desenvolvimento de software é necessário organização, planejamento e entendimento sobre os método que devem ser seguidos.

Durante todo o período de observação, acompanhamento das atividades e análise dos questionários realizado com alunos, foi possível identificar as grandes dificuldades enfrentadas por eles nos processos de desenvolvimento. Essa afirmação pode ser confirmada observando os resultados das questões 4, 5 e 7 do questionário 1, onde foram questionados os porquês de alguns alunos definirem as metodologias como melhores, se possuíram dificuldades na escolha das mesmas e explanar um pouco sobre as mesma, respectivamente. Na questão 4 e 5 é visível a falta de argumento dos alunos ao tentar explicar o porque da escolha das metodologias, mostrando a falta de conhecimento em relação a elas. Na questão 7, ao serem questionados a respeito do conhecimento sobre as metodologias escolhidas por eles para o desenvolvimento dos projetos, foram expostos apenas os conceitos básicos de uso dos processos, nenhum aluno mostrou conhecimento mais profundo a respeito.

O questionário 1 teve o objetivo de colher dados mais qualitativos, e através dele foi possível identificar o nível de conhecimento teórico dos alunos.

Após a aplicação dos processos definidos, foi visto com o questionário 2 que as metodologias não foram aplicadas da maneira correta, os alunos não tinham conhecimento dos procedimentos a serem seguidos, dessa maneira utilizavam como justificativa a adaptação das metodologias, onde no planejamento inicial propuseram realizar apenas pequenas adaptações no decorrer do projeto.

4.4. SOLUÇÃO PROPOSTA

Com todo o levantamento teórico realizado nesta pesquisa, é possível perceber a grande importância do estudo sobre a Engenharia de Software, onde a mesma tem um papel fundamental no sucesso do desenvolvimento de software.

A disciplina de Engenharia de Software ocorre no 5º período do curso de Licenciatura em Computação, por um período de aproximadamente 4 meses e meio. Nela, são vistos os conceitos e principais teorias a respeito das metodologias de desenvolvimento, gerência de projetos, qualidade de software e outros diversos fatores que facilitam ao aluno o entendimento sobre todos os procedimentos para se ter um bom produto.

Com a pesquisa foi possível identificar que a maioria das metodologias utilizadas pelos alunos nos projetos são as metodologias ágeis, dessa maneira levando em consideração o problema descrito na seção 3.3.3. deste trabalho, e analisando as dificuldades de entendimento da metodologias vividas pelas equipes, propõe-se que uma das formas de melhoria do rendimento dos alunos na disciplina de Projeto de desenvolvimento de software no conhecimento das metodologias ágeis, seria a utilização de procedimentos práticos na disciplina de Engenharia de software.

A proposta seguiria os seguintes passos:

1. Explanação acerca dos conceitos de Engenharia de software, sendo eles qualidade de software, gerencia de projetos, processos da engenharia, etc;
2. Realizar o levantamento dos principais processos ágeis de desenvolvimento utilizados atualmente, como Scrum e XP;
3. Divisão da turma em pequenas equipes, podendo variar de acordo com a metodologia escolhida;
4. Definição de um produto a ser desenvolvido por todas as equipes ao mesmo tempo, onde a metodologia utilizada seria a mesma, analisando assim o desenvolvimento de cada equipe na entrega do produto final;
5. Definição de pequenos ciclos dentro da disciplina, onde as equipes trabalhariam o mesmo produto em metodologias diferentes;
6. Ao final da disciplina fazer um comparativo entre as vantagens e desvantagens de cada metodologia aplicada.

4.4.1. PLANEJAMENTO ATUAL DA DISCIPLINA DE ENGENHARIA DE SOFTWARE

Na disciplina de engenharia de software são realizados seminários pelos alunos para a explanação a cerca das metodologias de desenvolvimento, mas a disciplina acontece sem nenhum tipo de prática para a fixação dos assuntos pelos aluno. Segue abaixo o cronograma atual da disciplina.

Figura 17: Planejamento atual de Engenharia de Software

IF - Sertão Pernambucano

CRONOGRAMA DE AULAS - 2013

CURSO: LICENCIATURA EM COMPUTAÇÃO		**TURNO: Vespertino**
DISCIPLINA ENGENHARIA DE SOFTWARE	**CARGA HORÁRIA** 60h / 80 aulas	**PROFESSOR** JUSSARA ADOLFO MOREIRA

No. De Aulas	**Unidade**	**Data**	**Conteúdo Programático**	**METODOLOGIA E RECURSOS MATERIAIS**
2		01/08/2013	Apresentação professor, disciplina e aluno	Exposição Oral
2		02/08/2013	Introdução em Engenharia de software	Exposição Oral / Quadro Branco / Projetor Multimídia
2		08/08/2013	Ciclo de vida e de desenvolvimento - Processos - Vídeo Metodologia tradicional (ambiente de desenvolvimento da Gonow)	Exposição Oral / Quadro Branco / Projetor Multimídia
2		09/08/2013	Processos, metodologias de desenvolvimento (Cascata)	Exposição Oral / Quadro Branco / Projetor Multimídia
2		16/08/2013	Processos, metodologias de desenvolvimento (Espiral - prototipação - 4 geração - CMMi	Exposição Oral / Quadro Branco / Projetor Multimídia
2		22/08/2013	Processos / modelos de processos- CMMi, MPS.br	Exposição Oral / Quadro Branco / Projetor Multimídia
2		23/08/2013	Métodos Ágeis - LEAN, XP, SCRUM, Open UP, FDD Visão Geral	Exposição Oral / Quadro Branco / Projetor Multimídia
2		29/08/2013	Prototipação em papel - Elaborar a proposta do jogo e entregar protótipo	Atividade avaliativa
2		30/08/2013	Ciclo PDCA e Conceitos de Qualidade de Sw	Exposição Oral / Quadro Branco / Projetor Multimídia
2	1ª	05/09/2013	Conceitos de Qualidade de Sw	Exposição Oral / Quadro Branco / Projetor Multimídia
3		06/09/2013	Prototipação em papel - Entregar protótipo ATÉ 27/09	Exposição Oral / Quadro Branco / Projetor Multimídia
3		12/09/2013	Métricas de Software e Teste de software ferramentas	Exposição Oral / Quadro Branco / Projetor Multimídia
3		13/09/2013	Gerenciamento de Projetos areas de conhecimento PMI	Exposição Oral / Quadro Branco / Projetor Multimídia
2		19/09/2013	Seminário - RUP e OpenUP,	Atividade em equipe
3		20/09/2013	Seminário - XP e Kanbam	Atividade em equipe
2		26/09/2013	Seminário - Espiral, prototipação, MPS.br e CMMi - SCRUM	Atividade em equipe
2		27/09/2013	**ATIVIDADE AVALIATIVA**	Atividade avaliativa
2		03/10/2013	Técnicas criativas - Brainstorm clássico, Brainstorm construtivo e destrutivo e método 635	Exposição Oral / Quadro Branco / Projetor Multimídia
3		04/10/2013	Definição do jogo - uso de técnicas criativas - Brainstorm clássico, Brainstorm construtivo e destrutivo e método 635	Atividade em equipe
3		10/10/2013	Especificação de requisitos	Exposição Oral / Quadro Branco / Projetor Multimídia

Fonte: Elaborado pelo autor

4.4.2. PLANEJAMENTO PROPOSTO PARA A DISCIPLINA DE ENGENHARIA DE SOFTWARE

O planejamento proposto visa uma maior participação dos alunos, tornando a disciplina mais prática fazendo com que os mesmos tenham um melhor entendimento das metodologias aplicadas, com o planejamento proposto será possível focar mais nos métodos ágeis já que foi possível identificar na pesquisa que na maioria dos projetos são utilizados os métodos ágeis.

Os seminários serão realizados pelos alunos como uma simulação de uso das metodologias, essa aplicação ocorrerá com as outras equipes tentando passar aos outros alunos os principais aspectos e características que compõe a metodologia estudada e de que maneira a mesma é aplicada, fazendo com que a aula se torne mais atrativa e os mesmos saibam de forma prática como a metodologia é realmente utilizada.

Figura 18: Planejamento proposto a disciplina de Engenharia de Software

IF - Sertão Pernambucano

CRONOGRAMA DE AULAS – 2014

CURSO: LICENCIATURA EM COMPUTAÇÃO		**TURNO: Vespertino**
DISCIPLINA	**CARGA HORÁRIA**	**PROFESSOR**
ENGENHARIA DE SOFTWARE	60h / 80 aulas	JUSSARA ADOLFO MOREIRA

No. De Aulas	**Unidade**	**Data**	**Conteúdo Programático**	**METODOLOGIA E RECURSOS MATERIAIS**
2	1ª	15/09/2014	Apresentação professor, disciplina e aluno	Exposição Oral
2		16/09/2014	Introdução em Engenharia de software	Exposição Oral / Quadro Branco / Projetor Multimídia
2		22/09/2014	Ciclo de vida e de desenvolvimento - Vídeo Metodologia tradicional	Exposição Oral / Quadro Branco / Projetor Multimídia
2		23/09/2014	Qualidade de software	Exposição Oral / Quadro Branco / Projetor Multimídia
2		30/09/2014	Gerência de projetos	Exposição Oral / Quadro Branco / Projetor Multimídia
2		06/10/2014	Conceito geral sobre os Processos de desenvolvimento (Metodologias) tradicionais	Exposição Oral / Quadro Branco / Projetor Multimídia
2		07/10/2014	Escolha das principais metodologias de desenvolvimento ágil, definição dos produtos a serem desenvolvidos pelas equipes, definição das equipes (Podem ser modificadas no decorrer do projeto)	Exposição Oral / Quadro Branco / Projetor Multimídia
2		13/10/2014	Métricas de Software, Teste de software e ferramentas	Exposição Oral / Quadro Branco / Projetor Multimídia
2		14/10/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		20/10/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		21/10/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		27/10/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		28/10/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		03/11/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		04/11/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		10/11/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		11/11/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		17/11/2014	Aplicação da metodologia com os alunos em sala no desenvolvimento do produto	Seminário
2		18/11/2014	Elaboração de jogos educativos	Exposição Oral / Quadro Branco / Projetor Multimídia
2		24/11/2014	Prototipação em papel - Entregar protótipo	Atividade avaliativa
2		25/11/2014	**ATIVIDADE AVALIATIVA**	Atividade avaliativa

Fonte: Elaborado pelo autor

5. CONCLUSÃO

Através da realização dessa pesquisa foi possível constatar a real dificuldade da utilização das metodologias de desenvolvimento pelas equipes, sejam elas tradicionais ou ágeis.

Durante a realização de todo o percurso metodológico foi possível perceber que a maioria dos alunos, apesar de não dominar a metodologia escolhida pela sua equipe, são relutantes em afirmar suas reais dificuldades no desenvolvimento do projeto.

Na escolha das metodologias, foi percebido que a falta de conhecimento das equipes em relação aos vários processos existentes e mais utilizados atualmente, culminam na escolha da metodologia em que os integrantes tiveram mais contato desprezando assim, a análise que cada projeto, com suas variáveis diferentes, tem para a escolha da metodologia mais adequada.

Como origem dos problemas em relação ao entendimento das metodologias foi percebido que a disciplina de Engenharia de Software tem um papel fundamental para o sucesso das equipes na disciplina de Projeto de Desenvolvimento de Software, visando que, é na primeira disciplina citada onde a maioria dos alunos tem o primeiro contato com os processos de desenvolvimento. Demonstrando assim, a necessidade da utilização de uma metodologia de ensino que vise a união linear das duas disciplinas.

A possível solução que foi proposta visa contribuir para um melhor desenvolvimento dos projetos propostos na disciplina de Projeto de Desenvolvimento, facilitando todo o processo percorrido pela equipe para a obtenção de um projeto bem planejado, organizado e com o cliente satisfeito.

Como trabalhos futuros, podem ser realizados experimentos na turma de engenharia de software para identificar a viabilidade do uso da solução proposta. Podendo ainda em outro momento, realizar análises similares nas demais disciplinas do curso de Licenciatura em Computação, aplicando a mesma metodologia deste trabalho.

6. REFERÊNCIAS

AMBLER, S. **Modelagem ágil:** práticas eficazes para Programação Extrema e Processo Unificado. Porto Alegre: Bookman, 2004.

ASTELS, D.; MILLER G.; NOVAK, M. **Extreme Programming - Guia prático.** Campos, 2002.

AUGUSTINE, S.; PAYNE, B.; SENCINDIVER, F.; WOODCOCK, S. (2005). **Agile Project Management: Steering from the Edges. Communications of the ACM**, v. 48, n. 12, pp. 85 – 89.

BARROWS, H. S. **Problem-based learning in medicine and beyond: a brief overview**. In: WILKERSON, L.; GIJSELAERS, W. H. (Eds.). Bringing problem-based learning to higher education: theory and practice. San Francisco: Jossey-Bass, 1996. p. 3-12

BASSI FILHO, Dairton Luiz. **Experiências com desenvolvimento ágil.** 2008. 154 f. Dissertação (Mestrado em Ciências) – Instituto de Matemática e Estatística da Universidade de São Paulo, São Paulo, 2008.

BASSI FILHO, D. L. **Experiências com desenvolvimento ágil.** 170 pág. Dissertação (Mestrado) - Universidade de São Paulo, São Paulo, 2008. BECK, Kent, ANDRES, Cynthia. **Extreme Programming Explained.** Publisher: Addison-Wesley Professional; 2 edition, 2004.

BECK, K. Extreme **Programming Explained: Embrace change**. Addison-Wesley Professional, 1999.

Belbin, M.R. (1993) "**Team Roles at Work**", Elsevier Butterworth-Heinemann Ltd.

BOEHM, B.; TURNER, R. (2003). **Balancing agility and discipline: a guide for the perplexed.** Addison Wesley.

BOEHM, B.; TURNER R. (2004). **Balancing agility and discipline: evaluating and integrating agile and plan-driven methods.** In Proceedings of the 26th IEEE International Conference on Software Engineering (ICSE), pp. 718 – 719.

CAETANO, Rodrigo. **Metodologias de desenvolvimento: qual a mais adequada?.** Computerword, 2009. Disponível em: http://computerworld.com.br/gestao/2009/08/05/metodologias-de-desenvolvimento-qual-a-mais-adequada/. Acesso em 10 março de 2014.

CAVALCANTE, Shirley Maria. **Gestão da Comunicação organizacional: conhecendo as ferra- mentas e suas aplicabilidades.** 2008.

<http://www.aberje.com.br/monografias/MONOGRAFIA%20Shirley%20Cavalcante%20PDF.pdf> Acesso em 10 de novembro de 2013.

CISNEIROS, H. JACOB, M. S. MAZZA, S. PEREIRA, T. Engenharia de Software – **Modelo de Desenvolvimento ágil Scrum**. São Paulo, 2009.

COCKBURN, Alistair. **Agile Software Development.** Adisson-Wesley, 2001.

DINSMORE, Paul Campbell. **Gerência de Programas e Projetos.** São Paulo: Pini, 1992.

FAGUNDES, Priscila Bastos. **Framework para Comparação e análise de Métodos Ágeis.** Dissertação de Mestrado. Universidade Federal de Santa Catarina. Florianópolis: 2005.

FOWLER, M. **The New Metodology.** 2001. Disponível em: <http://www.martinfowler.com/articles/newMetodology.html> Acesso em 08 de abril de 2014.

FRAME, J. Davison – **Managing projects in organization**, São Francisco – Jossey – ass Inc., 1995

FUGGETTA, A., Software **Process: A Roadmap**, In: Proc. of The Future of Software Engineering, ICSE'2000, Limerick, Ireland, 2000, pp. 25-34.

GIL, A. C. **Métodos e técnicas de pesquisa social.** 6. ed. São Paulo: Atlas, 2010.

GÜZELIS, C. "**An Experience on Problem Based Learning in an Engineering Faculty**", In: Turk J Elec Engin, Vol.14, No., p. 67-76. 2006

HERMANO. **Visão Geral do RUP**. 2003. Disponível em : <http://www.cin.ufpe.br/~if717/slides/visao-geral-do-rup.ppt.>. Acesso em 13 de abril de 2014.

IEEE - Institute of Eletrical and Eletronics and Engineers**. IEEE Std 610.12-1990 - Standard glossary of software engineering terminology**. Piscataway: IEEE, 1990.

JALOTE, P. **An integrated approach to software engineering.** 2 ed. New York: Springer-Verlag,1997.

JEFRIES, R. **What is Extreme Programming**. 2001. Disponível em: <http://www.extremeprogramming.com>. Acesso em 13 abril de 2014.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** 2. ed. São Paulo: Novatec, 2007.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software: aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software.** 2. ed. São Paulo: Novatec, 2007.

KRUCHTEN, P. **The Rational Unified Process: an introduction**. Boston. Addison-Wesley, 1999.

KRUCHTEN, P. **The Rational Unified Process: an introduction**. 3Th Edition. Addison-Wesley, 2003.

LEWIS, James P. **Como gerenciar projetos com eficácia.** Tradução Gilberto de Araújo Brandão Couto. Rio de Janeiro: Campus, 2000.

MACEDO, O. A. C. **Diretrizes para desenvolvimento de linhas de produtos de software com base em Domain-Driven Design e métodos ágeis.** 153 pág. Dissertação (Mestrado) - Universidade de São Paulo, São Carlos, 2009.

MAFFEO, Bruno. **Engenharia de Software e Especificação de Sistemas**. Rio de Janeiro: Campus, 1992.

MAINART, Domingos de A.; SANTOS, Ciro M**. A Importância da Tecnologia no Processo Ensino Aprendizagem**. VII Convibra Administração. Anais. 2010.

MARTIN, J. McCLURE, C., **Técnicas estruturadas e Case**. São Paulo: Makron Books, 1991

MARTINS, J. C. C. **Gerenciando projetos de desenvolvimento de Software com PMI, RUP e UML**. 4 ed. Ver. – Rio de Janeiro: Brasport, 2007.

MINICUCCI, Agostinho. **Psicologia aplicada à administração**, São Paulo, Atlas, 1995.

NERUR S.; MAHAPATRA, R.; MANGALARAJ, G. (2005). **Challenges of Migrating to Agile Methodologies. Communications of the ACM**, v. 48, n. 5.

NETO, Oscar Nogueira de Souza. **Análise Comparativa das Metodologias de Desenvolvimento de Softwares Tradicionais e Ágeis**. Trabalho de Conclusão de Curso. Universidade da Amazônia. Belém: 2004.

NING, C. **"Undergraduate academic programme: planning, development, implementation and evaluation"**, In: J. Engng. Educ., v.11, n.3, p.175-84, 1995.

OLIVEIRA, Otavio J, et AL. **Gestão da Qualidade – Tópicos Avançados**. São Paulo:

Thompson pioneira, 2004.

PAINKA, Marcelo Augusto Lima; MARCHI, Késsia Rita da Costa. **Utilização das Metodologias Ágeis Xp e Scrum para o desenvolvimento rápido de Aplicações.** 2003. Disponível em: <http://web.unipar.br/~seinpar/2013/artigos/Marcelo%20Augusto%20Lima%20Painka.pdf>. Acessado em 20 de abril de 2014.

PAULA FILHO,W.P. **Engenharia de Software: fundamentos, métodos e padrões**. Rio, LTC, 2001

PERBONI, Marcos. **Adoção de metodologias ágeis, PMI-ACP, curso preparatório, certificação.** 2013. Disponível em: < http://marcosvperboni.wordpress.com/2013/02/15/adocao-de-metodologias-ageis/ > Acessado em 20 de junho de 2014.

PINHEIRO, M. R. e SILVA-DE-SOUZA, T. **Rational Unified Process: uma abordagem gerencial**. Dissertação de Mestrado, Instituto Militar de Engenharia, Rio de Janeiro, Brasil, 2005.

PMI, PROJECT MANAGEMENT INSTITUTE. **A Guide to the Project Management Body of Knowledge (PMBok)**. Maryland: Project Management Institute Inc., 2001.

PRESSMAN, R. S. **Software engineering: a practitioner's approach**. New York: McGraw-Hill, 1997.

PRESSMAN, R. S. **Engenharia de Software.** 5a Ed., Makron Books, 2002.

REZENDE, D. A. **Engenharia de software e sistemas de informação.** Rio de Janeiro: Brasport, 2005.

RIBEIRO, L. R. C. **Radiografia de uma aula de engenharia**. São Carlos: EDUFSCar, 2007.

SATO, Danilo Toshiaki. **Uso eficaz de métricas em métodos ágeis de desenvolvimento de software.** 2007. 139 f. Dissertação (Mestrado em Ciências) – Instituto de Matemática e Estatística da Universidade de São Paulo-SP, São Paulo, 2007.

SCHWABER, Ken. **Agile Project Management with** *Scrum*, Washington: Microsoft Press, 2004.

SOMMERVILLE, I. **Software Engineering**. New York: Addison-Wesley, 1996.

SOMMERVILLE, I. **Engenharia de Software.** São Paulo: Addison-Wesley, 2004.

STEFANINI. **Ferramenta visual de gestão de fluxo de desenvolvimento, o kanban é uma das mais conhecidas metodologias de Gerenciamento Ágil de Projetos.** 2013. Disponível em <http://stefanini.com/br/2013/11/kanban-gerenciamento-agil-projetos/>. Acessado em 15 de agosto de 2014.

TELES, Vinicius Manhães. **Um estudo de caso da adoção das práticas e valores do Extreme Programming.** (Dissertação de Mestrado). 2005. Disponível em: <http://www.improveit.com.br/xp/dissertacaoXP.pdf> Acesso em 15 março de 2014.

TELES, Vinícius Manhães. **Extreme Programming: aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade.** São Paulo: Novatec, 2006.

YIN, R. K. **Estudo de caso: planejamento e métodos.** 2. ed. Porto Alegre: Bookman, 2001.

ANEXO A – QUESTIONÁRIO 1

Questionário

*Obrigatório

Qual nome da sua empresa? *

Quais as metodologias de desenvolvimento escolhidas pela sua equipe para o projeto? *

Porque essas metodologias foram escolhidas? *

- a) Pois é mais fácil de utilizar
- b) Porque é a que a equipe entende mais
- c) Porque achamos que é a melhor
- d) Porque é a melhor

Caso a resposta anterior tenha sido as opções C ou D, explique.

Tiveram dificuldade na escolha das mesmas? Porque? *

Como tiveram conhecimento sobre a metodologia escolhida? *

Fale um pouco sobre elas. *

Enviar

ANEXO B – QUESTIONÁRIO 2

Questionário 2

*Obrigatório

1) As metodologias escolhidas pela equipe, foi utilizada da maneira correta? *

- Sim
- Não
- Em partes (De maneira adaptada)

2) Em relação ao uso total das metodologias escolhidas pela sua equipe, você classificaria que utilizou: *

- 25 % no projeto
- 50 % no projeto
- 75 % no projeto
- 100 % no projeto
- Utilizamos menos de 10 %

3) Durante a execução do projeto houve algum problema no uso das metodologias? *

4) Houve a adaptação das metodologias no decorrer do projeto? *

- Sim
- Não

5) O processo definido pela equipe para o desenvolvimento do projeto foi seguido? *

- Sim
- Não
- Em partes

6) Houve a mudança de metodologia no decorrer do projeto? *

- Sim
- Não

7) Caso a resposta anterior tenha sido SIM, responda. Qual metodologia foi escolhida para substituir a outra? *

Comentários!!