MÉTODOS ÁGEIS E SCRUM NA DISCIPLINA DE ENGENHARIA DE SOFTWARE

Mônica Mara da Silva
monica@aedb.br
AEDB

Vitor Ferreira Horácio
vitor.horacio00@gmail.com
AEDB

Célia Maria Cerantola de Mattos
cel.eng.mattos@uol.com.br
AEDB

Resumo:RESUMO: aprender da maneira tradicional através de aulas expositivas está ficando cada vez mais em desuso. O perfil atual do aluno deseja aulas mais dinâmicas e interativas. Diante deste novo cenário o professor precisa adequar suas aulas para atender esse novo perfil do estudante, por isso o presente artigo traz uma pesquisa bibliográfica sobre os métodos ágeis, sobre Srum, como esta metodologia ágil pode auxiliar no desenvolvimento de softwares e finaliza descrevendo como este conteúdo foi ensinado na disciplina de engenharia de software. Considerando ser essa disciplina extremamente teórica e densa, decidiu-se por flexibilizar o conteúdo de uma maneira mais concreta e utilizando-se de aulas invertidas e dinâmicas. Após avaliação dos resultados, verificou-se um resultado expressivamente mais positivo, com participação de todos e envolvimento pleno. A construção coletiva, em times favoreceu os resultados

Palavras Chave: Métodos ágeis - Scrum - Engenharia de Softwa - -

1. INTRODUÇÃO

Por muito tempo Métodos Ágeis estava ligado a gestão de projetos, porém com os avanços tecnológicos, essa metodologia foi adaptada para ser utilizada em vários setores da tecnologia da informação – TI, principalmente no desenvolvimento de software.

No início da década de 2000, foi criado o Manifesto Ágil que tinha como conceitos básicos no desenvolvimento de software agregar mais valor aos indivíduos e interações do que nos processos e ferramentas; software funcionando mais que documentação abrangente; colaboração com o cliente mais que negociação de contratos; responder a mudanças mais que seguir um plano (PRIKLADNICKI, 2014).

Foram criadas e adaptadas várias metodologias ágeis e várias frameworks[1] para auxiliar no desenvolvimento de software. Métodos ágeis não pode ser considerado como único modelo que irá auxiliar na criação de software porque em alguns projetos os modelos tradicionais podem satisfazer no desenvolvimento do produto.

Uma das frameworks mais utilizadas é o Scrum. Este artigo dará enfoque nessas estruturas e como foi ensinada na disciplina de engenharia de software. O processo de ensino e aprendizagem foi dividido em etapas: conceitos teóricos do Scrum utilizando sala de aula invertida, questionamentos propostos pelos alunos, aula expositiva e finalizando foi realizada uma dinâmicas onde o aluno pode vivenciar todos as etapas do processo do framework no desenvolvimento de software.

Os capítulos do artigo descrevem sobre o Manifesto Ágil, como o início da ligação desta metodologia com o desenvolvimento ágil de software, descreve o framework Scrum, mostra como a disciplina de engenharia de software pode abordar de uma maneira mais lúdica essa metodologia e apresenta como foi ensinada na prática esse conteúdo.

2. MANIFESTO ÁGIL

Uma das mais importantes indústrias da era moderna é a indústria de software, presente em diversas atividades com níveis de complexidade baixos como controle de entrada e saída até níveis mais altos como o reconhecimento de padrões da inteligência artificial.

O ambiente das empresas de softwares é altamente competitivo e com isso existe um esforço por parte delas para o lançamento de inovações em produtos e serviços, e com essa mentalidade iniciou-se nos anos 90 formas de desenvolvimento conhecidas como Métodos Ágeis que trouxe uma nova forma de enxergar o desenvolvimento de softwares, mudando o foco dos processos para as pessoas e o seu conjunto de princípios e práticas possibilitando a rápida resposta às mudanças constantes do mercado (WILLIAMS E COCKBURN, 2013).

O Manifesto Ágil para o desenvolvimento de software foi criado em 2001. A partir deste anos foram introduzidos várias métodos, técnicas e melhores práticas para o desenvolvimento.

O Manifesto Ágil não rejeita ferramentas nem os processos já definidos, apenas indica sua importância secundária quando comparados com indivíduos e interações, softwares e a rápida resposta às mudanças. O objetivo não é a mudança propriamente dita, pois ela é inevitável em qualquer projeto mas sim como encarar, avaliar e responder a elas (AGILEMANIFESTO, 2001).

Abaixo estão descritos os 12 princípios definidos pelo Manifesto Ágil:

[1] Framework – é uma estrutura de suporte em torno da qual algo pode ser construído

1. Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
2. Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
3. Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
4. Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
5. Construir projetos em torno de indivíduos motivados, dando a eles o ambiente e o suporte necessário e confiando neles para fazer o trabalho.
6. O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é por meio de conversa face a face.
7. Software funcionando é a medida primária de progresso.
8. Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
9. Contínua atenção a excelência técnica e bom design aumenta a agilidade.
10. Simplicidade: a arte de maximizar a quantidade de trabalho não realizado é essencial.
11. As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis.
12. Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

3. MÉTODOS ÁGEIS EM DESENVOLVIMENTO DE SOFTWARE

As metodologias ágeis propõem formas de desenvolvimento iterativo, disciplinado e criativo, visando entregas rápidas e frequentes de versões. Isso possibilita ao cliente ter e fornecer feedback com frequência, permitindo o aumento de sua satisfação, já que ele passa a ter melhor visão do andamento do projeto e do sistema em desenvolvimento (PRIKLADNICKI, 2014).

Para elaboração de um sistema é fundamental seguir um roteiro que auxilie a criar um produto de qualidade e dentro do prazo definido. Este roteiro é chamado “processo de software”. Para projetos pequenos pode ser utilizado o Modelo Cascata embora ele está cada vez mais em desuso. Ele foi o primeiro modelo de processo de desenvolvimento de software a ser publicado, ~~a~~onde deve-se planejar e agendar as atividades dos processos antes de executá-las.

As etapas do modelo cascata refletem diretamente as atividades do desenvolvimento. Basicamente, o desenvolvimento é dividido em algumas etapas, que começam assim que a anterior termina (VETORAZZO, 2018):

- **Especificação:** é feita a análise e captação das necessidades do cliente e definidos os “*requisitos do sistema*” com suas funcionalidades;
- **Projeto do software:** é elaborado o desenho da arquitetura geral do sistema, seguindo as informações coletadas na etapa anterior. Tudo é separado em partes chamadas de *“unidades de programa”* para poder agilizar o desenvolvimento. Cada unidade terá seu próprio “*requisito de unidade*”;
- **Implementação:** nesta etapa, com o projeto definido, cada “*unidade de programa*” é implementada, podendo alocar vários programadores simultaneamente.
- **Teste unitário:** no final da implementação, são feitos os “*testes de unidade*” para certificar (validação) que as implementações atendem aos “*requisitos de unidade*”;

- **Integração:** com as unidades testadas, a próxima tarefa é integrar todas elas para compor um sistema completo.
- **Teste de Sistema:** no final da integração, é feito um teste geral para certificar (validação) que os "*requisitos do sistema*" foram completamente atendidos. Por fim, o software é entregue ao cliente;
- **Operação e manutenção:** Ao operar, ou seja, utilizar o sistema, o cliente poderá encontrar erros ou necessidades que não foram identificadas anteriormente. Esta fase se responsabiliza pela correção dos erros ou melhorias do sistema (evolução).

Os métodos ágeis atuais tentam diminuir o risco no desenvolvimento de software com curtos períodos que são chamados de iterações, cada iteração é basicamente um mini projeto e inclui todas as tarefas necessárias para implantar uma nova função. Um projeto de software ágil busca a capacidade de implantar uma nova versão do software ao fim de cada iteração, etapa a qual a equipe responsável deverá reavaliar as prioridades do projeto.

3.1 SCRUM

Scrum é uma das ferramentas ágeis mais conhecidas e utilizadas no mercado. O Scrum é considerado um framework, e não uma metodologia, uma das suas grandes vantagens é a sua fácil aplicação em projetos. Resumidamente, no Scrum, é elaborado uma lista de funcionalidades que devem ser desenvolvidas, criando a seguir os famosos sprints, que são as divisões onde cada funcionalidade deve ser criada (RUBIN, 2017).

O Scrum nasceu nos 90 mas ganhou o mundo apenas da década seguinte e chegou desbancando métodos tradicionais e se tornou a forma mais comum de se trabalhar em projetos de desenvolvimento de software.

No Scrum existem os Sprint, que são reuniões, que ocorrem periodicamente para saber que os resultados foram alcançados, se precisa haver alguma melhoria ou acrescimento no projeto. Além dos *sprints*, existem reuniões diárias e semanas para discutir a evolução do projeto, dificuldades e novos prazos, se necessário. As reuniões diárias são feitas em um curto período e é recomendado que sejam feitas de maneira informal.

O Scrum ajuda a reduzir os riscos de insucesso e ajuda a entregar valor mais rápido, lidar com as inevitáveis mudanças de escopo e transformando-as em vantagem competitiva. Seu uso pode também aumentar a qualidade do produto entregue e melhorar a produtividade das equipes (COHN, 2011).

Scrum é aplicado em projetos com características igualmente variadas. Em projetos críticos de centenas de milhares de dólares e em projetos internos simples. Em projetos para produção de softwares comerciais, de sites da Internet, de softwares embarcados, de aplicativos para dispositivos móveis, de softwares financeiros e de jogos (SABBAGH, 2018).

É importante lembrar que não existe uma solução única para todos os problemas. Scrum é um framework simples e pequena e, assim, funciona bem em cada contexto se for utilizado em conjunto com outras técnicas e práticas a serem experimentadas e adaptadas.

Os benefícios no uso do Scrum incluem:

• Entregas frequentes de retorno ao investimento dos clientes;

• Redução dos riscos do projeto;

• Maior qualidade no produto gerado;

• Mudanças utilizadas como vantagem competitiva;

• Visibilidade do progresso do projeto;

• Redução do desperdício;

• Aumento de produtividade.

Scrum possibilita que se entreguem, desde cedo no projeto e frequentemente, partes do produto funcionando. Cada uma dessas entregas proporciona retorno ao investimento realizado pelos clientes do projeto, além de possibilitar o seu feedback rápido sobre o produto para que se realizem as mudanças ou adições necessárias (VETORAZZO, 2018).

Scrum visa à redução dos riscos de negócios do projeto pela colaboração com os clientes e demais partes interessadas durante todo o seu decorrer. Os riscos também são reduzidos com a produção em ciclos curtos e entregas frequentes de partes prontas do produto, partindo-se das mais importantes em direção às menos importantes.

4. ENGENHARIA DE SOFTWARE E MÉTODOS ÁGEIS

A engenharia de software fornece uma visão geral das atividades, técnicas, métodos e ferramentas que auxiliam no processo de desenvolvimento de software. Acompanha todo o ciclo de vida do software, desde o levantamento de requisitos, avaliação dos riscos, classificação e análise das ameaças de um projeto, prevenindo ou corrigindo cada problema que possa surgir no desenvolvimento e manutenção do software. A engenharia de software ensina os conceitos de gerência de configuração de sistemas de software e capacita na utilização de métodos, técnicas e ferramentas utilizadas para tal finalidade (ENGHOLM, 2018).

O processo de desenvolvimento de software foi por muito tempo definido como algo complexo e de difícil gerenciamento. Quando o programa é pequeno, a ausência de um plano formal pode até dar certo, mas mesmo os projetos pequenos ficam em riscos quando novas funcionalidades são requisitas para inserção no software. Quanto maior e mais complexo for o programa, mais imprescindível é o conhecimento da engenharia de software. Ela auxiliar na escolha dos métodos e técnicas que serão utilizadas para minimizar, corrigir e manter o produto (PRESSMAN, 2006).

Escolher uma metodologia para ajudar na produção e gerenciamento do software não é uma tarefa fácil, deve-se levar em conta os processos de desenvolvimento dos envolvidos. Porém existem técnicas que facilitam a criação e elaboração dos programas e aplicativos atuais. Essas técnicas e métodos tem o objetivo de tornar a criação do programa o mais previsível e eficiente possível (SBROCCO, 2012).

Desenvolver um software usando métodos ágeis requer uma equipe de desenvolvimento eficaz, com integrantes qualificados e que sejam um time de verdade, tendo comprometimento na elaboração do projeto e com o cliente. Uma das premissas dos métodos ágeis é que as pessoas envolvidas no desenvolvimento do produto devem aceitar o processo e que não se deve impor este. Quando uma tarefa for aceita pela equipe, os envolvidos tem mais comprometimento e envolvimento na realização das tarefas, o que muitas das vezes não é visto em processos impostos.

5. A DISCIPLINA DE ENGENHARIA DE SOFTWARE E O SCRUM

Levando em consideração o que foi apresentado até agora, a disciplina de engenharia de software precisa fazer esse link entre a teoria e a prática sobre desenvolvimento de software e métodos ágeis.

O que foi questionado era como fazer com que os alunos aprendessem os conceitos e também assimilassem o espírito de equipe e que trabalhassem como um time?

Tornar o aluno agente do seu aprendizado não é uma tarefa das mais fáceis de ser realizada, depois de trabalhar por algum tempo com aulas expositivas sobre esse assunto, decidiu-se que este modelo tradicional de ensino não estava sendo adequado para ministrar esta parte do conteúdo para os alunos. Fez-se uma reformulação na metodologia, que antes era utilizada com métodos tradicionais e aulas expositivas e com uma avaliação no final. Atualmente são utilizados métodos mais ativos, onde o foco são as pessoas (no caso o aluno), transparência nos processos, engajamento dos envolvidos e feedback constante.

Entretanto nas aulas tradicionais utiliza-se no processo de ensino a seguinte sequência: em um primeiro momento o aluno recebe o conteúdo sobre o assunto através da plataforma utilizada pelo professor. O conteúdo ~~foi~~ disponibilizado em formato de texto e em vídeo. O aluno precisava ler o conteúdo ou assistir os vídeos. O conteúdo escrito não era muito extenso e os vídeos com duração de no máximo 5 minutos cada. Depois de assistir os vídeos ou ler o conteúdo, o aluno precisa fazer uma pergunta no fórum. Todos os alunos tinham que postar uma pergunta e responder uma questão feita por outra pessoa. Esse questionário criado pelos discentes precisava ser sempre analisado pelo professor e as respostas também precisam ser acompanhadas. Deve haver *feedback* periodicamente e avaliação da participação da turma constantemente.

Mesmo utilizando a sala de aula invertida que consiste em fazer com que o aluno já chegue em sala com algum conhecimento sobre o tema, é preciso que o professor faça um fechamento falando sobre o que o aluno precisava ter assimilado. Porém o professor não deve se estender na aula expositiva.

Neste tipo de interação deve haver comprometimento de ambas as partes – alunos e professor, caso o aluno deixe de cumprir algumas das tarefas dadas até o momento o professor terá que explicar tudo novamente em sala de aula e o objetivo não é este.

Utilizando metodologias inovadoras para a dinâmica sobre o framework Scrum, a turma foi dividida em equipes, cada uma com os papéis recomendados no framework Scrum. Abaixo será relatado cada função exercida dentro do framework (SUTHERLAND, 2020):

O Product Owner (PO) é o ponto central do projeto ágil e é quem exerce a liderança sobre o produto que está sendo desenvolvido. É ele quem diz o que precisa e o que não precisa ser feito em relação ao produto que está sendo desenvolvido.

O Scrum Master é que está sempre à disposição da equipe auxiliando quando necessário e garantindo a eficiência para a realização do projeto. É ele quem facilita e potencializa o trabalho de todos, sempre mantendo a comunicação entre os outros integrantes e deve ter um sólido conhecimento em Scrum.

O Scrum Team (Development Team) é a equipe de desenvolvimento. Não existem papéis tradicionais definidos na framework. As funções de programador, designer, analista de testes ou arquiteto é feita por todos. Sempre valorizando o trabalho em equipe com o qual todos estão comprometidos. Um Scrum Team típico tem em torno de 6 a 10 pessoas na equipe de desenvolvimento.

5.1 O PROBLEMA

Foi apresentado o problema para a turma fazer um projeto para construção de um terminal rodoviário. O PO que foi representado pelo professor definiu o que deveria ser feito

para a construção do terminal. Os alunos foram divididos e equipes. Cada equipe deveria escolher um membro para ser o Scrum Master. Este deveria cumprir as funções designadas para este cargo.

Para realizar a dinâmica, foi delimitado com fita adesiva o espaço representando a área de construção conforma a Figura 1.

Figura 1 Área delimitada para a dimâmica

Fonte: Os autores

O professor, exercendo a função de PO, fez uma lista com os itens que constavam para serem feitos:

1. Construção das rampas de acesso aos ônibus;
2. Bilheteria para a venda das passagens;
3. Banheiros masculinos e femininos;
4. Telhado em todo o terminal;
5. Sala de espera para os passageiros;
6. Praça de alimentação;
7. Muro em torno do terminal;
8. Espaço destinado a lojas;
9. Trocador infantil;
10. Calçamento da área externa do terminal;
11. Calçamento da área interna do terminal;
12. Paisagismo na área externa;
13. Sistema de som;
14. Rampas de acessos para pessoas portadoras de necessidades especiais.

Quando a lista dos itens foi divulgada os alunos deveriam colocar as prioridades para execução de cada item como foi mostrado na figura 2. Eles poderiam sugerir novos itens, retirar ou agrupar.

Foi distribuído peças de montagens para que os alunos pudessem executar a construção do terminal.

No quadro da sala as tarefas foram distribuídas em *postit* para que todos pudessem visualizá-las e aloca-las em outro lugar conforme a necessidade. Conforme as tarefas iam sendo executadas mudavam de lugar para a parte concluída.

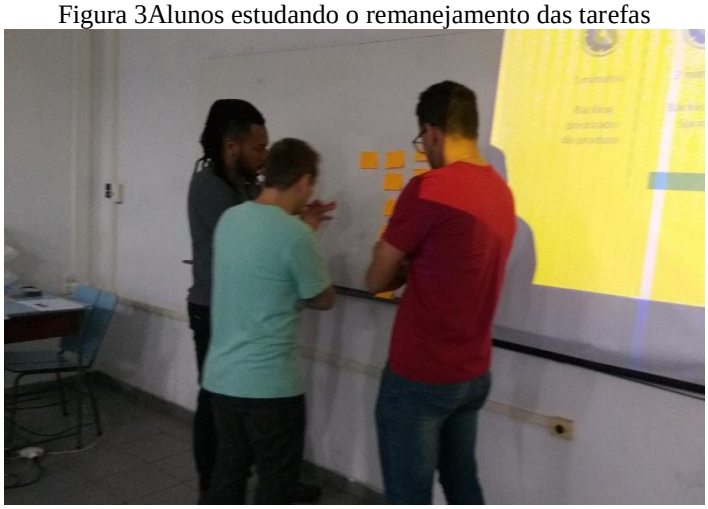
Figura 2 Distribuição das tarefas

Fonte: Próprios autores

Depois das tarefas distribuídas os alunos puderam começar a montar o terminal rodoviário. O PO marcou o tempo de cada Sprint. Indiferente da tarefa ter terminado ou não, dando o tempo pré definido os grupos se reuniam para dialogar sobre as tarefas que foram cumpridas ou não. As tarefas que não eram cumpridas deveriam voltar para o quadro e ser realocada para as equipes novamente.

Quando os banheiros masculinos e femininos formam construídos, os alunos perceberam que poderiam juntar a tarefa de construção dos trocadores infantis. Esta tarefa ficou separada justamente para ver a percepção do aluno que poderia agrupar estas tarefas e que ele ganharia tempo executando-as juntos.

Figura 3Alunos estudando o remanejamento das tarefas



Fonte: Próprios autores

Para que esta dinâmica dê certo, o PO deve propor algum tipo de modificação para ser realizada para perceber como a equipe se comporta diante de tal situação. Deve ser analisado o engajamento dos alunos e a capacidade de adaptação às mudanças.

Figura 4 Terminal rodoviário ficando pronto

Ao final da dinâmica deve haver um fechamento onde o professor irá indagar o que foi aprendido, o que não deu certo, o que poderia melhorar. Falar de cada papel desempenhado e como essa metodologia ágil ajuda no processo de criação do software.

6. CONCLUSÃO

Nesses tempos contemporâneos não basta o desejo de mudar, é necessário buscar meios de mudanças. Faz-se necessário trabalhar em conjunto, conviver e construir novos conhecimentos e mecanismos de avaliação. O pensamento avaliativo começa com perguntas em determinada situação em que os alunos possam se reunir, pensar, debater, criar processos inovadores, criativos e baseados na concepção de que em todas as interações e atividades que nos envolvemos há aprendizado. O professor deve ir ajustando o processo e as condutas cada vez que seus "olhares conceituais" – por meio dos quais observa a realidade, visualizarem suas necessárias intervenções.

A escola tem dificuldade de acompanhar as mudanças no mundo, nas pessoas e a possibilidade de acesso rápido a qualquer informação. O Professor precisa alterar seu papel de *transmissor* para o de *dinamizador* da aprendizagem, para que a relação interpessoal seja verdadeira e aberta entre os aprendizes.

A capacidade de facilitação de aprendizagem não se resume a dominar um conteúdo, organizar currículos e cronogramas, dar aulas expositivas, usar livros e recursos áudio visuais – apesar de tais elementos poderem servir didaticamente. A facilitação da aprendizagem não

pode se limitar ao formalismo educacional, pois as mesmas acontecem no relacionamento pessoal entre o professor e o aluno. Uma turma na qual o professor privilegie uma interação positiva e harmoniosa com os alunos, estes poderão desenvolver competências relacionadas ao respeito e interações necessárias ao convívio com outros humanos.

O aluno precisa ter a percepção de que o que ele está fazendo está dando certo. Chamar atenção para os acertos e comportamentos desejáveis, investindo em uma estratégia metodológica de motivação e de incentivo, é mais eficiente do que corrigir o aluno quando ele erra. O trabalho em times propicia uma complementação de saberes entre os participantes sendo aconselhável que o professor esteja atento para manter o aluno motivado a aprender. O aluno deve perceber que a aprendizagem está ao seu alcance através do esforço que ele dedica a ela.

É evidente que quando planejamos, não podemos prever todas as ocorrências. É preciso deixar a porta entreaberta para a indeterminação, a flexibilidade para incluir novos elementos, novas práticas para contemplar as situações inéditas

REFERÊNCIAS

AGILEMANIFESTO Manifesto para desenvolvimento ágil de software, 2001 disponível em: - http://agilemanifesto.org/iso/ptbr/manifesto.html, acessado em 16/06/2020

COHN, Mike. Desenvolvimento de software com Scrum. Porto Alegre Bookman, 2011, 1 recurso online ISBN 9788577808199.

DICTIONARY CAMBRIDGE, disponível em: https://dictionary.cambridge.org/pt/dicionario/ingles/framework, acessado em: 07/07/2020

ENGHOLM, Hélio. Engenharia de Software na Prática. Novatec Editora. São Paulo, Brasil, 2010.

PRESSMAN, Roger. S. Engenharia de Software, 6ª Edição. McGrawHill, Nova York, EUA, 2006

PRIKLADNICKI, Rafael. Métodos ágeis para desenvolvimento de software. Porto Alegre Bookman 2014 1 recurso online ISBN 9788582602089.

RUBIN, Kenneth S. Scrum Essencial: um Guia Prático Para o Mais Popular Processo ágil, Editora Alta Books; Edição: 1, 2017

SABBAGH, Rafael, Scrum: Gestão ágil para projetos de sucesso, Casa do Código, 2018

SBROCCO, José Henrique Teixeira de Carvalho. Metodologias ágeis engenharia de software sob medida. São Paulo Erica 2012 1 recurso online ISBN 9788536519418.

SUTHERLAND, J. J., SCRUM: guia prático - Maior produtividade. Melhores resultados. Aplicação imediata, Editora Sextante, 2020

TELES, Vinícius Manhães Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade, Novatec Editora; Edição: 2, 2017

VETORAZZO, Adriana de Souza. Engenharia de software. Porto Alegre SAGAH 2018 1 recurso online ISBN 9788595026780.

WILLIAMS E COCKBURN, Agile software development: it's about feedback and change, 2003, disponível em: http://comum.rcaap.pt/bitstream/10400.26/31932/1/Leandro_Sousa.pdf, acessado em: 25/06/2020