Implementação de metodologias ágeis para o gerenciamento de projetos em uma indústria

Implementation of Agile Methodologies for Project Management in an Industry

Implementación de metodologías ágiles para la gestión de proyectos en una industria

Daniel Luis Notari
Universidade de Caxias do Sul
dlnotari@gmail.com

Géssica Grolli
Universidade de Caxias do Sul
ggrolli3@ucs.br

Resumo

Uma empresa de grande porte do ramo metalúrgico desenvolve os softwares administrativos internamente com um processo de desenvolvimento inadequado as suas necessidades. Este trabalho descreve um estudo de caso aplicado a partir da análise da cultura organizacional. Foi criada e utilizada uma metodologia baseada no Design Thinking combinada com o método ágil SCRUM. O estudo de caso teve dois experimentos com equipes pequenas para validar a metodologia. O resultado mostrou-se adequado às necessidades da empresa. Ajustes foram feitos na metodologia e, posteriormente, aplicado para toda a TI. O estudo auxiliou no planejamento das demandas, apoiando o desenvolvimento e gerenciamento de projetos de software.
Palavras-chave: Design Thinking. SCRUM. Métodos Ágeis. Gerenciamento de Projeto.

Abstract

A large-sized company in the metallurgical industry develops its own corporate software using an inadequate software process development. This paper describes an applied case study based on the analysis of the company's cultural environment. A methodology based on Design Thinking combined with SCRUM Agile Method was created and used. The case study included two experiments with small teams to validate the methodology. The outcome showed to be appropriate to the company's needs. Some adjustments were made to the methodology, and after that, it was applied to the entire IT sector. The study aimed to plan the demands by supporting software development and project management.
Keywords. Design Thinking. SCRUM. Agile Method. Project Management.

Resumen

Una gran empresa de la industria metalúrgica desarrolla internamente software administrativo con un proceso de desarrollo inadecuado para sus necesidades. Este trabajo describe un estudio de caso aplicado desde el análisis de la cultura organizacional. Se creó y utilizó una metodología basada en Design Thinking combinada con el método ágil SCRUM. El caso de estudio contó con dos experimentos con pequeños equipos para validar la metodología. El resultado fue adecuado a las necesidades de la empresa. Se realizaron ajustes a la metodología y, posteriormente, se aplicaron a toda el área de tecnología de información (TI). El estudio ayudó en la planificación de las demandas, apoyando el desarrollo y la gestión de proyectos de software.

Palabras clave: Design Thinking. SCRUM. Métodos ágiles. Gestión de proyectos.

Introdução

O cliente quem dita o sucesso dos produtos de uma empresa (XAVIER, 2016). Com isso, as organizações buscam se reinventarem e, aceitar a permanente mudança. Exemplos envolvem o lançamento de novos produtos, modificação de linhas de produção ou mudanças administrativas. Cada um desses exemplos deve ser tratado como um projeto visando gerar um produto ou serviço.

Um sistema em que os requisitos não mudam é muito raro de ser encontrado (GALLOTTI, 2016). A indústria de software tem adequado os seus processos para realizar entregas de produtos, com o menor tempo possível aos seus clientes, visando não perder a qualidade, economia e satisfação.

Um processo de desenvolvimento de software precisa estar em constante evolução buscando atualizar-se e seguir as tendências de aproximação com o gerenciamento de projetos de forma ágil (FOGGETTI, 2014). É difícil mudar de uma abordagem tradicional para ágil, uma vez que não é recomendado que a ruptura seja feita de um dia para o outro, mas sim feita de forma sistemática (KERZNER, 2015). Uma característica muito importante é respeitar a cultura organizacional de uma empresa fazendo alterações planejadas e com o envolvimento dos colaboradores (MASSARI, 2018).

Um processo ágil pode reduzir o custo das alterações de um software que é entregue de forma incremental (PRESSMAN e MAXIM, 2016). Com isso, as alterações podem ser mais bem controladas a partir da estruturação da comunicação e atitude das pessoas de uma equipe.

O SCRUM é um método de desenvolvimento ágil de software criado por Jeff Sutherland na década de 1990 (SCHWABER e SUTHERLAND, 2017). O SCRUM (PHAM e PHAM, 2011; PHAM e PHAM, 2019) segue os princípios do Manifesto Ágil (BECK, 2001): a. Indivíduos e interações mais que processos e ferramentas;

b. Software em funcionamento mais que documentação abrangente; c. Colaboração com o cliente mais que negociação de contratos; e, d. Responder a mudanças mais que seguir um plano. Esses princípios são usados para orientar as atividades de desenvolvimento dentro de um processo que incorpora as atividades de: requisitos, análise, projeto, evolução e entrega.

O SCRUM além de ser uma metodologia, também possui maior ênfase nas pessoas e na comunicação (PRESSMAN e MAXIM, 2016), e estes são dois aspectos muito importantes dentro de uma organização ou departamento. SCHWABER e SUTHERLAND (2017) citam que o SCRUM é um framework estrutural, que está sendo usado para gerenciar o trabalho em produtos complexos, dentro do qual pessoas podem tratar e resolver problemas complexos e adaptativos.

Além do SCRUM, outra abordagem que visa a comunicação e promove maior ênfase nas pessoas de uma organização é o Design Thinking (DT) (VIANNA, 2018). Para (CAMARGO e RIBAS, 2019), o termo DT trabalha muito com a empatia, que é a capacidade de se colocar no lugar da outra pessoa, com o objetivo de enxergar as dificuldades e necessidades pelos olhos do outro. De acordo com MAKIOSZEK (2019), o DT auxilia para averiguar se um projeto ou ideia é eficiente e importante para o cliente, ou organização. (CAMARGO e RIBAS, 2019) abordam que o DT não é uma solução padrão, ou seja, pode ser aplicada a qualquer projeto, necessidade ou oportunidade com o objetivo de buscar a melhor solução para projetos de definição e circunstâncias incertas. Outros autores também reforçam a ideia da viabilidade de se integrar o DT com métodos ágeis (STRAY et al., 2020; DOBRIGKEIT e DE PAULA, 2019; DOBRIGKEIT et al., 2019; PERREIRA e DE FSM RUSSO, 2018; CORRAL e FRONZA, 2018; DOBRIGKEIT e DE PAULA, 2017).

O objeto de estudo desse artigo é uma empresa metalúrgica de grande porte denominada ABC[1]. Essa empresa possui um departamento de TI (Tecnologia da Informação) voltado para o desenvolvimento e manutenção próprios dos seus softwares administrativos. Um dos problemas atuais é deixar o levantamento de requisitos apenas na visão do usuário. Neste contexto, alguns dos projetos entregues nem sempre atendem as

[1] Para preservar o nome da empresa, iremos identificá-la ao longo deste trabalho como a empresa ABC.

dificuldades previstas pelos solicitantes. Outro ponto diz respeito as estimativas feitas em que há grandes atrasos nas entregas. Atualmente, o departamento não realiza o uso específico de uma metodologia de gerenciamento de projetos. Sendo assim, as atividades são realizadas conforme é conveniente para a equipe de desenvolvimento naquele momento. Neste contexto, as atividades e fases dos projetos não são realizadas de forma padronizada, e as fases existentes como definição e distribuição de tarefas são tratadas com base na experiência e disponibilidade dos programadores.

Este trabalho teve como objetivo realizar um estudo de caso no departamento de TI para propor e implantar uma metodologia ágil de gerenciamento de projetos utilizando as práticas DT e SCRUM.

Esse artigo apresenta os trabalhos relacionados e ferramentas no Capítulo 2. Os materiais e métodos são descritos no Capítulo 3. Os resultados e a discussão são apresentados nos Capítulos 4 e 5. Por fim, o Capítulo 6 apresenta as conclusões.

Trabalhos Relacionados

De acordo com COHN (2011), a empresa Salesforce.com percebeu as vantagens ao adotar o SCRUM. Em 2006, a empresa percebeu que a frequência do lançamento de suas versões tinha diminuído de quatro por ano para uma por ano. Os clientes estavam recebendo menos, mas esperando mais. Para resolver esta situação a empresa decidiu realizar a transição para o SCRUM. No primeiro ano com SCRUM, a Salesforce.com teve um aumento de 94% nas versões de seu sistema, distribuiu 38% mais recursos por desenvolvedor e 500% a mais de valor a seus clientes.

Um estudo de caso descrito em ANDRADE et al. (2011) mostra como uma empresa do ramo de desenvolvimento de software utilizando SCRUM. Foi observado que o projeto foi entregue dentro do prazo e do orçamento melhoraram em relação ao que foi previsto. Os autores constataram uma maior participação e satisfação do cliente, principalmente pelo tempo fixo estimado para as Sprints. Os autores também destacaram que a equipe pode evoluir

profissionalmente, tornando-se mais autoconfiante e com autogerenciamento, e que este crescimento foi gradativo, no decorrer das Sprints.

A criação de um modelo de desenvolvimento de software baseado no DT, Lean Startup e SCRUM foi proposta por DOBRIGKEIT e DE PAULA (2017). O modelo foi nomeado de InnoDev. Esse modelo foi transformado em produto de software relatado em DOBRIGKEIT et al. (2019). No artigo, os autores descrevem o uso dos três métodos juntos. O DT auxilia no entendimento dos requisitos para o desenvolvimento do software. O Lean Startup contribui com as estratégias de negócio. Por fim, o SCRUM apresenta as práticas de gerenciamento de projetos de software. Nesse sentido, os autores propuseram um novo método ágil de processos InnoDev. Esse método está centrado no usuário para o desenvolvimento inovativo e melhoria de processos, produtos e serviços de software de empresas.

Uma revisão sistemática relacionada a integração de DT com práticas ágeis para o desenvolvimento de software foi realizada por PERREIRA e DE FSM RUSSO (2018). Foram selecionados 29 artigos para esse estudo. Os resultados demonstram que a integração dos modelos com o ciclo de vida de desenvolvimento de software acontece com mais frequência entre o DT e o SCRUM. A união dos modelos resulta em uma maior aproximação com os usuários finais e a equipe de desenvolvimento de software, melhorando a qualidade e a usabilidade do software produzido.

O artigo de DOBRIGKEIT e DE PAULA (2019) explora o uso de DT em empresas de desenvolvimento de software através de um estudo de caso em uma empresa multinacional. A aplicação do estudo de caso relata que os funcionários foram treinados no método DT e o seu impacto em uma empresa de desenvolvimento de software. Os funcionários obtiveram sucesso nas fases de desenvolvimento de software, bem como, em projetos envolvendo o desenvolvimento de processos, espaços de trabalho de equipes e o desenvolvimento do trabalho em equipe.

Um estudo de caso foi realizado por MORAES et al. (2019) em uma filial de uma grande empresa brasileira de software. No ano de 2017, passou por uma mudança de processos com a implantação do SCRUM e do DT. Para este objetivo, observaram a necessidade de uma combinação de planos, pessoas e processos para que pudesse ter uma evolução. Os benefícios percebidos com

a implantação do DT foram: conhecimento compartilhado, tempo na resolução de problemas, poder fazer mais com menos, manutenção junto de inovação e a forma de trabalho (sem retrabalho). Como desafios, citaram a cultura, papéis, dimensionamento, maturidade, planejamento, estimativa, comunicação e gerenciamento.

Uma coleção de artigos sobre o desenvolvimento de software usando métodos ágeis foi apresentada no trabalho de STRAY et al. (2020). Os artigos envolvem o desenvolvimento de casos em empresas, artigos teóricos a respeito de DT e SCRUM, abordagens práticas para equipes de desenvolvimento de software. Um dos artigos é um *survey* de pesquisadores brasileiros sobre o uso de DT combinado com práticas ágeis. Nesse *survey*, 127 profissionais da indústria brasileira de desenvolvimento de software foram entrevistados. Como resultado, os autores apresentaram a existência de uma grande variedade de cenários diferentes com o uso de DT, bem como de práticas ágeis e software para o gerenciamento do processo de desenvolvimento de software unindo DT e práticas ágeis.

A importância de ferramentas que podem ser utilizadas em conjunto com o DT e o SCRUM foram apresentadas por ÖZKAN e MISHRA (2019) visando auxiliarem no gerenciamento de projetos de forma prática e ágil. As ferramentas citadas foram escolhidas por interesse do Departamento de TI da empresa ABC após uma seleção interna das possíveis ferramentas a serem utilizadas.

A ferramenta Trello[2] (ÖZKAN e MISHRA, 2019) é um aplicativo de gerenciamento de projetos que opera um modelo gratuito com opção de assinatura para acesso a recursos avançados. O Trello utiliza o Kanban (SENAPATHI e DRURU-GROGAN, 2021), representando os projetos por quadros (*boards*) que contém conjunto de listas de tarefas (cartões) que podem ser designados a usuários de um time e movidos de um *board* para outro[3].

A ferramenta ActiveCollab[4] (ÖZKAN e MISHRA, 2019) é uma ferramenta que permite organizar os projetos em tarefas e subtarefas, e com hierarquia entre elas. Esta ferramenta oferece a possibilidade de elencar prioridades em

[2] www.trello.com

[3] Disponível em https://blog.trello.com/br/metodo-Kanban Acesso em: 21 de maio 2021.

[4] https://activecollab.com/

seus cartões e, além disso, fornece alguns controles como cronômetro, quadros e relatórios de horário, para visualizar o tempo que a equipe está gastando nas tarefas. O *ActiveCollab* também possibilita analisar a rentabilidade do projeto, atribuindo taxas por hora a todos os membros da equipe.

A ferramenta Jira Software[5] (LI, 2018) permite organizar os projetos em tarefas e subtarefas, e com dependência entre elas. Esta ferramenta oferece a possibilidade de criar versões diferentes por projetos, elencar prioridades e vários controles como tempo estimado, realizado, categorias e *branches*. Possui quadro SCRUM já desenvolvido pensando-se na divisão em Sprints, além de variados relatórios e gráficos de tempos do projeto, carga de trabalho e tarefas, *burndown* e *burnup*.

A ferramenta Service Desk[6] é uma ferramenta que visa gerenciar as tarefas (solicitações) e a lista de projetos solicitados ao setor. Com esta ferramenta, é possível que os usuários nos encaminham solicitações de incidentes e dúvidas e cadastrar tarefas de melhorias ou tarefas vinculadas a um projeto previamente cadastrado e atribuí-las aos analistas ou desenvolvedores. No Service Desk é possível priorizar as tarefas através de três prioridades: normal, alta e crítica, também é possível identificar o *status* do projeto, a quantidade de horas investidas, e a sua porcentagem de execução.

Tabela 1 - Comparação entre ferramentas de acompanhamento de projetos

	Trello	ActiveCollab	Jira Software	Ferramenta própria
Controle de Status	✓	✓	✓	✓
Dashboard	✗	✗	✓	✓
Alertas por e-mail	✓	✓	✓	✗
Armazenamento de arquivos	✓	✓	✓	✓
Kanban	✓	✓	✓	✗
Facilidade	✓	✗	✗	✓
Software Gratuito	✗	✗	✗	✓
Criação de lista de subtarefas	✓	✓	✓	✗
Identificação de prioridade	✓	✓	✓	✓
Lançamento de horas trabalhadas	✗	✓	✓	✓
Chat online para o grupo do projeto	✗	✓	✓	✗
Velocidade rápida	✓	✓	✓	✗
Possibilidade de criar etiquetas ou categorias	✓	✓	✓	✗
Gráfico de Gantt	✗	✓	✗	✗
Gráfico de Burndown	✗	✗	✓	✗

[5] https://www.atlassian.com/br/software/jira

[6] Desenvolvida internamente pelo departamento de TI da empresa ABC

A Tabela 1 apresenta uma análise das características das ferramentas apresentadas. O controle de *status*, está presente em todas as ferramentas e é uma característica indispensável na indicação das diferentes fases do projeto.

O uso de *dashboard* não está presente nas ferramentas Trello e ActiveCollab, mas é um diferencial importante para demonstrar em uma tela, visualizações rápidas dos principais indicadores de desempenho relevante aos projetos.

O alerta por correio eletrônico é uma característica que não existe na ferramenta própria, mas pode vir a ser algo vantajoso para notificar os interessados sobre o andamento e vencimentos dos prazos das tarefas.

O armazenamento de arquivos (todas as ferramentas possuem) é uma característica bem importante, pois um projeto poderá ter diferentes documentos ou imagens.

O Kanban, não está presente na ferramenta própria, porém é um aspecto bem importante para o controle do fluxo de execução das tarefas.

A facilidade de uso é um atributo importante para a operação e manutenção das informações levando a uma interface intuitiva, em comparação às demais ferramentas percebeu-se maior necessidade de adaptação na ferramenta ActiveCollab *e* Jira Software.

Ser um *software* gratuito é um atributo pertencente apenas na ferramenta própria, mas de certa forma é relevante para indicar os custos fixos que a empresa poderá vir a ter com novas ferramentas.

A criação de lista de subtarefas, atributo não identificado na ferramenta própria, mas que poderia auxiliar na identificação da hierarquia de tarefas e dependências.

A identificação de prioridade é um atributo presente em todas as ferramentas analisadas e é necessário para identificar as tarefas ou projetos mais importantes para a organização, ficando claro o que deverá ser executado por primeiro.

O lançamento de horas trabalhadas é uma característica não identificada na ferramenta Trello, porém é indispensável para dimensionar o custo do projeto.

O *chat* online está presente nas ferramentas *ActiveCollab e* Jira Software. Porém não se demonstrou ser um ponto tão importante visto que a empresa já viabiliza uma maneira de comunicação entre as equipes.

Na análise de velocidade rápida das ferramentas, percebeu-se menor eficiência na ferramenta própria. Esta característica pode ser julgada como um ponto importante, pois garante que os processos sejam realizados com maior agilidade e eficiência.

A possibilidade de criar etiquetas ou categorias, embora a ferramenta própria não disponibilize desta possibilidade, entende-se como um ponto importante na subdivisão do planejamento do projeto em diferentes formatos, podendo ser readequado de acordo com a demanda.

O gráfico de Gantt está presente apenas na ferramenta *ActiveCollab*, não é uma necessidade essencial, mas tende a ser um diferencial ao auxiliar na estimativa de prazo das demandas de uma forma visual e de fácil interpretação.

Material e Métodos

De acordo com FERRARI (1974), o método científico é um traço característico da ciência, um mecanismo básico que organiza o pensamento em sistemas e traça os procedimentos para atingir o objetivo. Já LAKATOS e MARCONI (2017) defendem que é possível utilizá-la também para a resolução de problemas do cotidiano.

Para a elaboração deste trabalho foi utilizado o método científico dialético, que de acordo com GIL (2008), busca interpretar a realidade partindo do pressuposto de que todos os fenômenos apresentam características contraditórias organicamente unidas e indissolúveis. LAKATOS e MARCONI (2017) concordam com este pensamento ao afirmar que sempre existe uma forma de se transformar, tornando o fim de um processo o início de outro.

Por se tratar de um estudo de caso com experimento, o meio de investigação adotado foi através da abordagem experimental, que segundo GIL (2008), consiste em definir e observar variáveis para então determinar formas de controle.

O método específico utilizado será o método comparativo, que se dará entre dois projetos da empresa ABC, e que conforme GIL (2008) procede pela investigação de indivíduos, classes, fenômenos ou fatos, com vistas a ressaltar as diferenças e as similaridades entre eles.

Para a elaboração da pesquisa foi utilizado a pesquisa de campo, que conforme LAKATOS e MARCONI (2017), é utilizada para obter informações sobre um problema ou hipótese, que se queira solucionar ou comprovar. O autor ainda comenta que esta pesquisa consiste na observação de fatos e fenômenos espontâneos, na coleta de dados e no registro de variáveis relevantes para a análise que se procura realizar.

Cenário Atual

O desenvolvimento desse projeto será realizado em uma empresa metalúrgica que possui um departamento de desenvolvimento de *software*. Este departamento dispõe de uma equipe de desenvolvimento formada por vinte e sete participantes que desempenham papéis variados para desenvolver o *software* ERP. O ERP é utilizado em toda a estrutura organizacional.

O departamento de TI atua junto a grupos de pessoas com conhecimento do negócio no direcionamento e especificação das funcionalidades do sistema, e pela priorização de ajustes ou desenvolvimento das demandas. Atualmente, o setor de TI, recebe projetos durante os primeiros seis meses do ano, caso houver algum projeto que demanda certa urgência e que não foi planejado no início do ano, pode-se realizar a inclusão deste projeto como Extra Plano, sempre sob aprovação do gerente de TI.

Os projetos são recebidos com um documento padrão que possui uma breve Análise de Requisitos (AR), e após aprovados são incluídos no portfólio de projetos. O fato de deixar o levantamento de requisitos apenas na visão do usuário, muitas vezes geram explicações superficiais, até porque, esses profissionais possuem conhecimento de negócio e não de sistema.

Neste contexto, alguns dos projetos entregues nem sempre atendem as dificuldades previstas pelos solicitantes. Quando esta situação ocorre, o projeto é abandonado logo após a sua entrega.

A programação e o andamento dos projetos do ano podem ser consultados em um sistema desenvolvido internamente, onde o analista do sistema sugere

uma estimativa de planejamento através da informação de ano/mês. Contudo, dificilmente esta estimativa é condizente com o ano/mês de entrega.

Atualmente, o programador analisa o documento AR anexado ao projeto, realiza um breve levantamento de requisitos de sistema, e parte imediatamente para o desenvolvimento do código. Enquanto desempenha esse papel, o programador também realiza o atendimento diário de solicitações e/ou de dúvidas sobre regras e procedimentos.

Atualmente o departamento não realiza o uso específico de uma metodologia de gerenciamento de projetos. Sendo assim, as atividades são realizadas conforme é conveniente para a equipe de desenvolvimento naquele momento. Neste contexto, as atividades e fases dos projetos não são realizadas de forma padronizada, e as fases existentes como definição e distribuição de tarefas são tratadas com base na experiência e disponibilidade dos programadores.

Metodologia Proposta

Visando a forte cultura enraizada na empresa ABC, resolveu-se criar um *framework* sob medida baseado no DT, incorporando algumas técnicas do SCRUM que possam auxiliar no gerenciamento do portfólio de projetos (STRAY et al., 2020; DOBRIGKEIT e DE PAULA, 2019; DOBRIGKEIT et al., 2019; PERREIRA e DE FSM RUSSO, 2018; CORRAL e FRONZA, 2018; DOBRIGKEIT e DE PAULA, 2017).

O Service Desk, *software* utilizado atualmente pela empresa, possui uma estruturação simples para acompanhar as fases de um projeto, que são: não iniciado, em andamento e concluído. Na Figura 1, pode-se verificar como é o funcionamento do processo destas três fases.

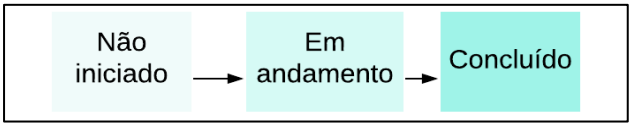


Figura 1 - Processo atual das fases de um projeto

Baseado no método DT, a proposta deste trabalho é incorporar duas novas divisões nas fases do projeto, que são: imersão de profundidade e prototipação.

Para o caso em questão, por motivo de padronização, serão utilizadas as nomenclaturas "Em planejamento" e "Em homologação" respectivamente.

A proposta implica em subdividir a fase "Em andamento" em três etapas, sendo a primeira "Em planejamento", a segunda "Em desenvolvimento" e por fim "Em homologação". A Figura 2 representa de forma ilustrativa esta proposta de adequação.

A fase de imersão preliminar será chamada de "Não iniciado", esta fase é realizada através das reuniões que ocorrem no primeiro semestre do ano e definem os projetos que farão parte do portfólio. Para que isso ocorra, o projeto precisa ser aprovado pelo setor de TI, além de possuir o documento AR preenchido, onde consta o real problema a ser resolvido, as partes interessadas, os objetivos do projeto, escopo e prioridades.

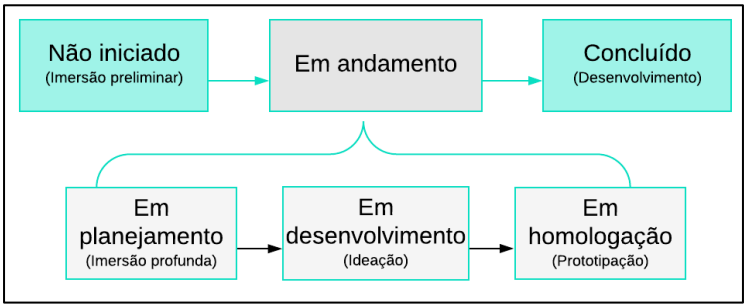


Figura 2 - Ilustração da proposta de subdivisão do processo atual

A fase de imersão profunda será chamada de "Em planejamento" e irá compreender a atividade de montar a equipe que irá trabalhar neste projeto e nivelar as informações entre eles. Para isso será realizada uma reunião para validar os objetivos, restrições, especificações, prazos, cronograma, recursos, divisão de tarefas, documentos do projeto e outras questões afins. Esse encontro será denominado reunião de iniciação.

A fase de ideação será chamada de 'Em desenvolvimento", esta fase irá realizar as etapas de monitoramento e controle do que está sendo desenvolvido, e verificará se a evolução está de acordo com o planejado. Em caso de problemas, é nesta fase que serão gerenciadas as possíveis soluções, e quando necessário serão realizadas reuniões com a equipe para manter o projeto alinhado.

A fase de prototipação será chamada de "Em homologação" e será responsável pela análise e *feedback* da entrega ao cliente. Como se trata da primeira entrega, poderá haver adequações de requisitos que não ficaram claros ou ainda ajustes de falhas do *software*. Quando isso ocorrer, será realizado um novo planejamento para desenvolvimento das premissas e entrega do projeto.

Tabela 2 - Resumo das etapas de cada fase do projeto

PROPOSTA DE FASES E ETAPAS DO PROJETO		
1ª FASE	**NÃO INICIADO**	
		Etapa A - Determinar o real problema a ser resolvido
		Etapa B - Identificar Stakeholders
		Etapa C - Definir objetivos
		Etapa D - Definir escopo, recursos e tarefas prioritárias
2ª FASE	**EM PLANEJAMENTO**	
		Etapa A - Montar a equipe do projeto
		Etapa B - Nivelar informações
		Etapa C - Reunião de iniciação
		Etapa D - Discriminar tempos
		Etapa E - Definir agenda do trabalho
3ª FASE	**EM ANDAMENTO**	
		Etapa A - Reunião frequentes com a equipe
		Etapa B - Monitorar e controlar o planejamento do desenvolvimento
		Etapa C - Reportar a evolução do projeto
		Etapa D - Gerenciar problemas
4ª FASE	**EM HOMOLOGAÇÃO**	
		Etapa A - Análise do feedback para possível replanejamento
5ª FASE	**CONCLUÍDO**	
		Etapa A - Concluir documentação
		Etapa B - Análise da performance
		Etapa C - Feedback
		Etapa D - Encerrar

A fase de desenvolvimento será chamada de "Concluído", nesta fase o projeto é encerrado e o produto é entregue para utilização do cliente. Nesta etapa também será analisado o desempenho dos envolvidos, onde o gerente poderá apresentar os indicadores de desempenho de produtividade do projeto, o *feedback* e a documentação.

Cada fase do projeto terá um conjunto de tarefas a ser seguido, estas tarefas serão monitoradas através do SCRUM *Board* ou Kanban (SENAPATHI e DRURU-GROGAN, 2021). Através da Tabela 2 identifica-se o resumo destas etapas separadas por fases, podendo ser observada através da mudança das cores, iniciando na cor mais clara até a mais escura. Já a Figura 3 está ilustrado o processo final, ao nível macro, do funcionamento da proposta de gerenciamento de projeto, nesta figura também é possível observar a evolução do processo através da mudança das cores, iniciando na cor mais clara até a mais escura.

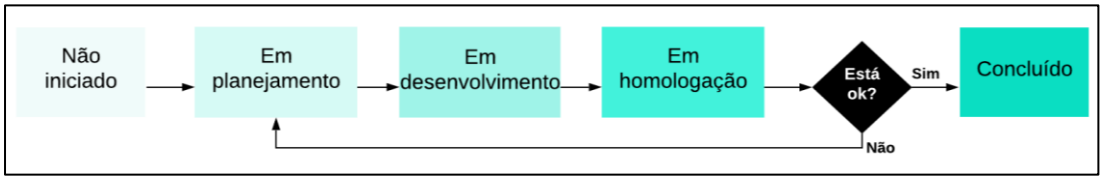


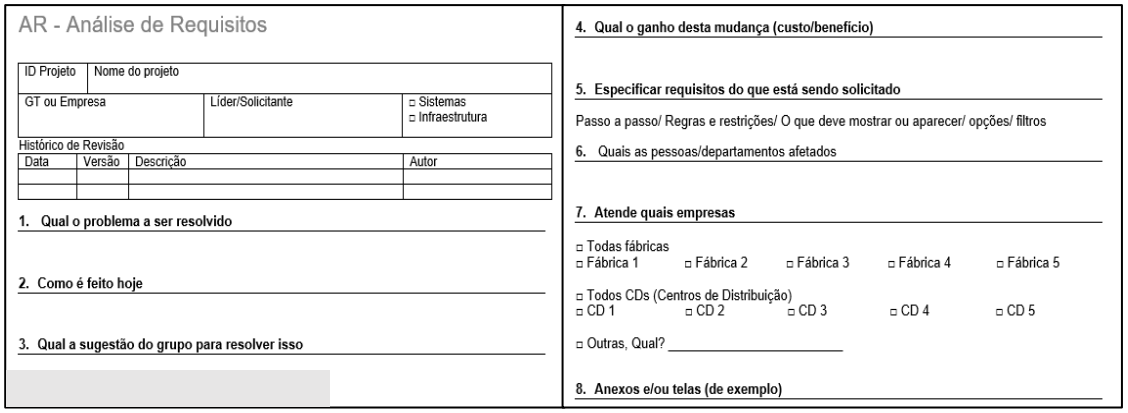
Figura 3 - Proposta final das fases de um projeto

Resultados

Nessa seção serão apresentados os experimentos realizados neste estudo de caso, em cada uma das etapas do projeto, bem como os detalhamentos a respeito da condução do experimento. Também serão apontados os resultados obtidos e as validações realizadas. Salienta-se que os dois experimentos estão apresentados de acordo com as fases da Figura 3.

Etapa Não Iniciado

Nesta fase preenche-se o documento com a análise de requisitos, que tem como objetivo atender as etapas iniciais do projeto, garantindo assim uma imersão preliminar ao assunto. A Figura 4 apresenta o modelo de documento utilizado.



AR - Análise de Requisitos

ID Projeto	Nome do projeto	
GT ou Empresa	Líder/Solicitante	▫ Sistemas ▫ Infraestrutura

Histórico de Revisão

Data	Versão	Descrição	Autor

1. Qual o problema a ser resolvido

2. Como é feito hoje

3. Qual a sugestão do grupo para resolver isso

4. Qual o ganho desta mudança (custo/benefício)

5. Especificar requisitos do que está sendo solicitado

Passo a passo/ Regras e restrições/ O que deve mostrar ou aparecer/ opções/ filtros

6. Quais as pessoas/departamentos afetados

7. Atende quais empresas

▫ Todas fábricas
▫ Fábrica 1 ▫ Fábrica 2 ▫ Fábrica 3 ▫ Fábrica 4 ▫ Fábrica 5

▫ Todos CDs (Centros de Distribuição)
▫ CD 1 ▫ CD 2 ▫ CD 3 ▫ CD 4 ▫ CD 5

▫ Outras, Qual? _____

8. Anexos e/ou telas (de exemplo)

Figura 4 - Documento criado para a análise de requisitos (AR)

Este levantamento prévio de requisitos auxiliou para que o departamento de TI identificasse qual a direção e caminho a percorrer para chegar ao que o cliente necessita. Com base neste documento, é possível identificar requisitos funcionais e não funcionais, e prever a dimensão e complexidade do projeto antes mesmo de sua análise.

Anteriormente, utilizava-se um campo de texto livre definido como 'descrição completa', onde o usuário relatava informações do projeto, e se houvesse interesse, também poderia anexar arquivos ou telas de exemplo. Porém, muitas vezes o cliente não sabia o que relatar e a descrição recebida não detinha informação suficiente para o entendimento e dimensão do assunto.

Etapa Em Planejamento

Nesta fase definiram-se os projetos a serem trabalhados, comprovou-se o recebimento do AR que detém a imersão preliminar do assunto e se definiram os integrantes que farão parte do time para o desenvolvimento do projeto.

Os projetos desenvolvidos neste experimento foram:

- Projeto 1 - Melhorias no processo de controle de documentos de prestadores de serviço, o qual terá integração com o controle de acesso às fábricas e demais unidades da empresa ABC.
- Projeto 2 - Atualização da versão do EDI (*Electronic Data Interchange*), que em português significa Troca Eletrônica de Dados

O segundo passo foi o treinamento da equipe de envolvidos no experimento nivelando-se o conhecimento sobre métodos ágeis. No treinamento, que teve duração de 30 minutos, foram repassados os ciclos do *framework* e apresentados em detalhes cada evento a ser executado, assim como as vantagens de sua utilização.

No terceiro passo realizou-se a separação do projeto em pequenas tarefas e uma reunião de iniciação. Esta reunião teve o objetivo de aprofundar o levantamento de requisitos e o entendimento da necessidade do usuário, além de realizar a estimativa de tempo para cada uma das tarefas. Também foram ordenadas as prioridades elegendo primeiramente as tarefas que agregam maior valor ao usuário, e após isso, priorizou-se a tarefa mais fácil, usando-a assim como referência na estimativa das próximas.

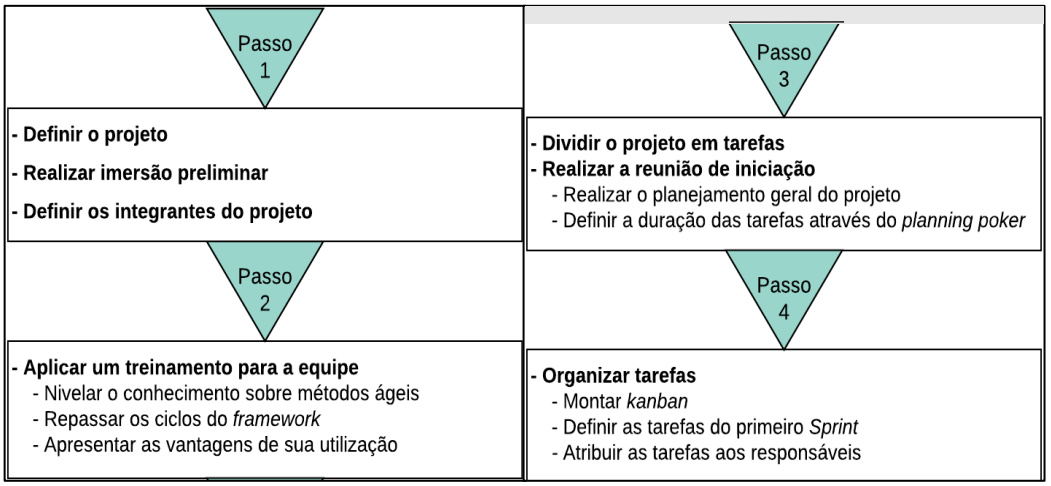


Figura 5 - Passo a passo para implementação da metodologia

Com o *backlog* ordenado, estimou-se a duração das tarefas através do método *Planning Poker* (PHAM e PHAM, 2011; PHAM e PHAM, 2019) convertido em horas, seguindo as necessidades e a realidade do setor. Após identificar os tempos de cada tarefa, estimou-se os itens que fariam parte do Sprint e definiu-se a duração do Sprint.

O primeiro *Sprint* teve a duração de uma semana, que é o período padrão de atualização de versão no departamento em questão. Já no segundo

experimento, optou-se por realizar um experimento com um Sprint quinzenal. De forma objetiva[7], a Figura 5 ilustra o passo a passo realizado.

Cada tarefa dos projetos experimentados teve a determinação de métricas e controles como a data de entrega, etiquetas que indicavam a Sprint *e* a etapa do projeto, a estimativa de horas e o total de horas trabalhadas.

Para o primeiro experimento, definiu-se uma equipe pequena, de apenas três integrantes, visando realizar o amadurecimento e o aprendizado da proposta. Esta equipe contou com um SCRUM Master, um analista de negócio, que não fazia parte do setor de desenvolvimento, e um desenvolvedor/analista. Foi realizado o levantamento dos requisitos (imersão profunda), analisou os impactos das necessidades propostas, indicou as melhores oportunidades para resolução da proposta e realizou o desenvolvimento do *software*.

Para os controles, métricas e montagem do Kanban, neste primeiro experimento utilizou-se a ferramenta ActiveCollab. Além disso, foi possível trocar mensagens através das tarefas, mantendo o histórico de todas as conversas do grupo em um único local. Também foi possível configurar lembretes por e-mail para tarefas que necessitam de algum tratamento especial. Bem como, foi possível visualizar de forma prática a divisão das tarefas durante a Sprint e a carga de trabalho dos envolvidos. Isso possibilitou uma melhor identificação dos trabalhos dependentes e dos tempos, garantindo assim, o bom planejamento das Sprints.

Para o segundo experimento, montou-se uma equipe de quatro integrantes, visando criar o papel do PO e, principalmente, retirar a responsabilidade da análise do desenvolvedor. Para este experimento também foram utilizadas as aprendizagens identificadas no primeiro projeto. Esta equipe contou com um SCRUM Master, que garantiu que a metodologia fosse aplicada e adaptada a realidade do setor; um PO - que neste primeiro momento, por falta de recursos humanos no setor, desenvolveu mais de um papel. E, o time teve dois desenvolvedores, para o desenvolvimento da aplicação.

[7] Há uma tabela com a lista de todas as atividades para cada uma das etapas que foram aplicadas nos experimentos. A mesma possui mais de quatro páginas e não foi adicionada ao artigo.

Etapa Em Andamento

Nesta fase iniciou-se o desenvolvimento da Sprint e realizou-se o monitoramento do trabalho realizado pelo programador. Para isso, utilizou-se a prática das reuniões diárias para *feedback* e entendimento do andamento do projeto, bem como a apresentação do gráfico de *burndown* para demonstrar a relação entre o que foi estimado e o que foi realizado. O detalhamento dos experimentos foi separado em duas subseções.

Experimento 1

No primeiro experimento as reuniões diárias ocorreram no início do dia, com horário e local predefinidos. Esta interação com o desenvolvedor teve uma duração média maior que o esperado nos primeiros dois Sprints, porém a partir do terceiro Sprint, os envolvidos se adaptaram ao método e pode-se realizar reuniões diárias de em média 15 minutos, que foram determinantes para o bom andamento do projeto.

Foi realizado o acompanhamento das tarefas diariamente através do gráfico de *burndown*, os quais foram utilizados para medir a velocidade da execução das tarefas em comparação ao estimado, e que puderam ser analisados durante o andamento da Sprint.

Os primeiros dois Sprints tiveram uma carga média de 31 horas semanais. Já o terceiro e último Sprint teve uma carga de trabalho estimada em 24 horas, mas que demandou apenas 13 horas de desenvolvimento, ficando assim mais compacta se comparada às anteriores.

A primeira Sprint teve a divisão do trabalho em 10 tarefas. Avaliando o gráfico do demonstrado através da Figura 6, percebe-se que houve um pequeno atraso no desenvolvimento das tarefas dos primeiros 3 dias. Contudo, este tempo pode ser recuperado a partir do quarto dia. Embora tenha ocorrido esse pequeno atraso no início, ainda se obteve um tempo remanescente de 2 horas ao final da Sprint. De uma forma geral o tempo estimado ficou muito próximo ao realizado.

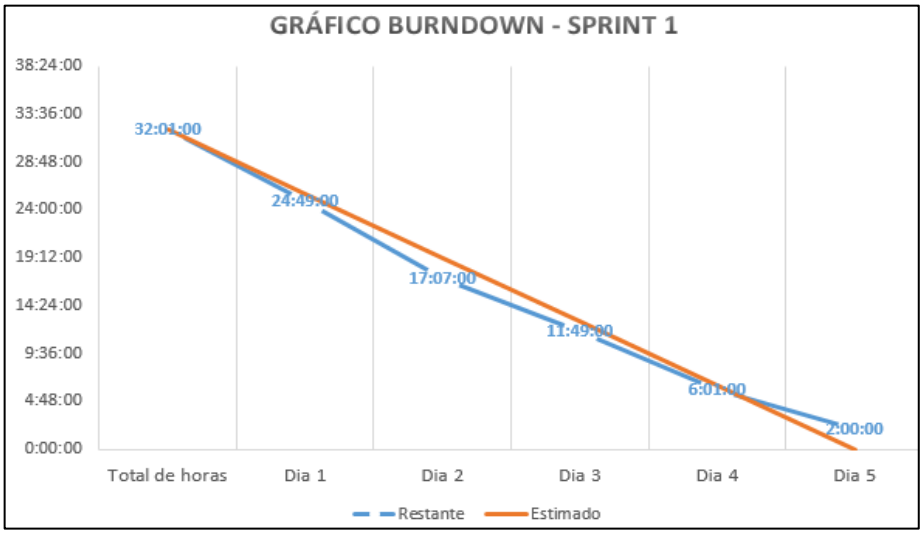


Figura 6 - Projeto 1: tempo estimado X realizado do 1º Sprint

Na Figura 7 verifica-se o andamento do segundo Sprint o qual foi repartido em 4 tarefas. Neste Sprint também se incluiu o tempo das reuniões diárias que foi um aprendizado da primeira Sprint, e incluiu-se um *bug* que ocorreu em consequência da primeira Sprint. O gráfico da Figura 7 demonstra que o desenvolvimento se manteve adiantado em relação ao estimado, e finalizou-se com um tempo remanescente de 5 horas e 48 minutos.

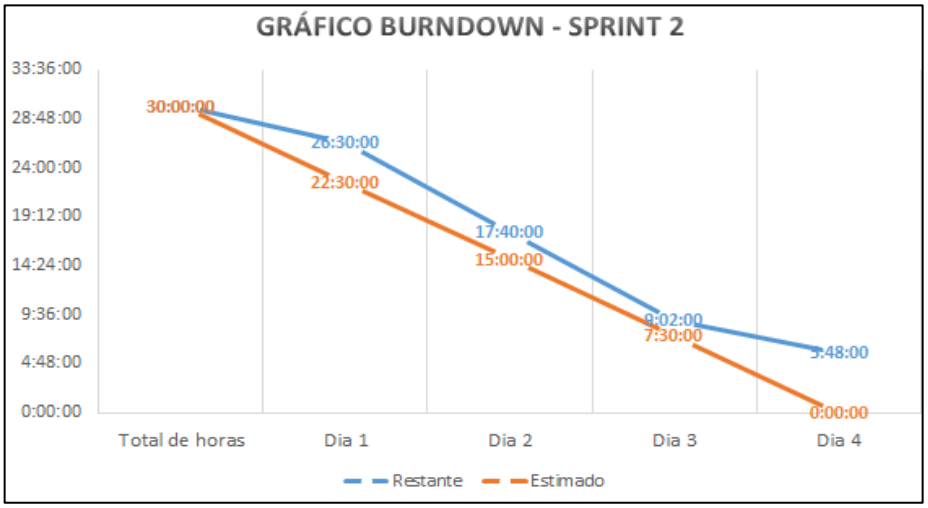


Figura 7 - Projeto 1: tempo estimado X realizado do 2º Sprint

Na Figura 8 verifica-se o andamento do terceiro Sprint o qual possui uma tarefa de desenvolvimento e mais três tarefas de monitoramento e/ou reuniões. Através dos Sprints anteriores, houve o aprendizado da necessidade de incluir o tempo das reuniões de revisão e planejamento da *Sprint* na estimativa e tempos.

No gráfico da Figura 8 percebe-se um atraso considerável no primeiro dia da Sprint, que ocorreu em detrimento de *bugs* do sistema legado que ocorreram e precisaram ser resolvidos urgentemente. Independente disso, no segundo dia o objetivo foi retomado e como esta Sprint possuía uma carga menor de trabalho o projeto não sofreu nenhuma consequência e ainda finalizamos com um tempo remanescente de 3 horas e 29 minutos.

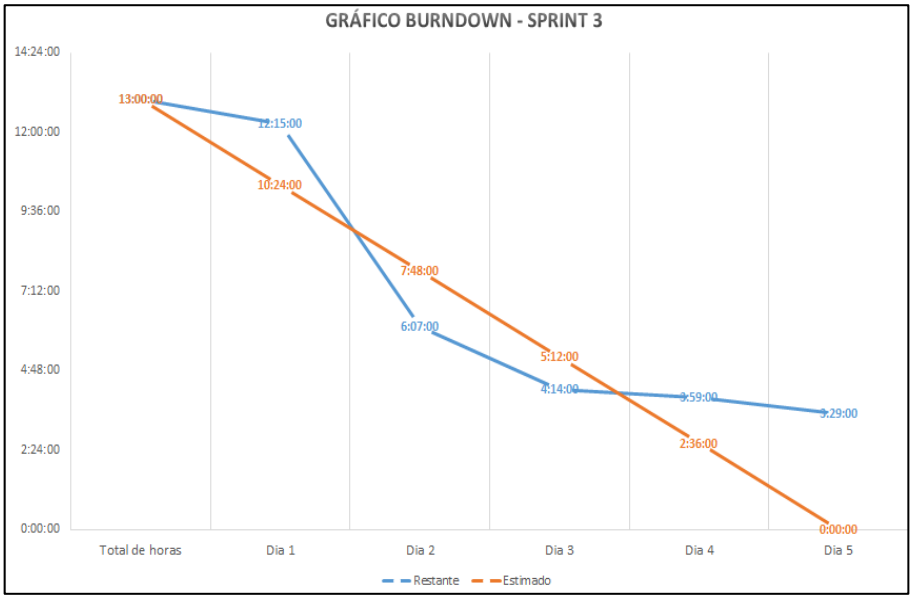


Figura 8 - Projeto 1: tempo estimado X realizado do 3º Sprint

Na Figura 9 verifica-se o andamento do projeto na totalidade, onde há a divisão levando-se em consideração os três Sprints semanais realizados. Verifica-se que os primeiros Sprints ficaram poucas horas atrasados em relação à velocidade total do projeto, e como o terceiro Sprint era menor, pode-se recuperar e ainda adiantar-se em torno de 19 horas do total estimado.

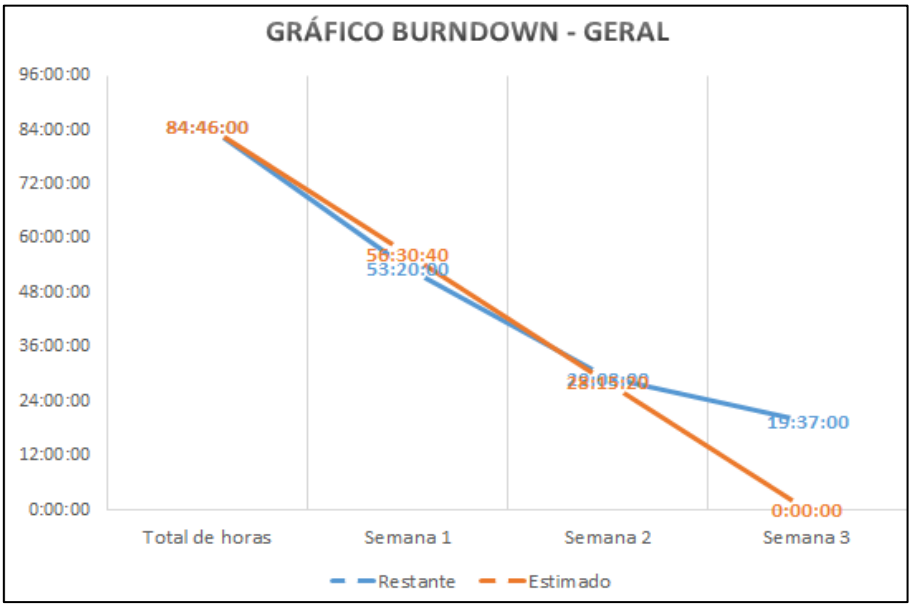


Figura 9 - Relação do tempo estimado X realizado do projeto 1

Experimento 2

No segundo experimento, as reuniões diárias também ocorreram no início do dia, sempre no início do expediente e em local predefinido. Esta interação com o time teve uma duração média de 15 minutos, que foi determinante para o bom andamento do projeto e principalmente para a identificação dos problemas e aprendizagens contínuas.

Neste segundo momento, também foi realizado o acompanhamento das tarefas através do gráfico de *burndown*. A primeira Sprint teve a divisão do trabalho em três histórias de usuários, que foram subdivididas em nove tarefas, totalizando pouco mais de 47 horas de estimativa de trabalho.

O importante a ser verificado no gráfico do primeiro Sprint demonstrado através da Figura 10, é a linha vermelha em relação à linha cinza (diretriz). Quanto mais próximas estas duas linhas estiverem, mais assertiva foi a estimativa das tarefas pelo time. Se a linha vermelha ficar abaixo da cinza significa que estamos adiantados, se a linha ficar acima, significa atraso. Percebe-se assim que na maior parte do tempo a Sprint esteve adiantada em relação ao tempo estimado, finalizando com um tempo remanescente de mais de 6 horas, para uma sprint semanal.

A segunda Sprint teve a divisão do trabalho em quatro histórias de usuários, que foram subdivididas em 15 tarefas, totalizando 141 horas de estimativa de trabalho.

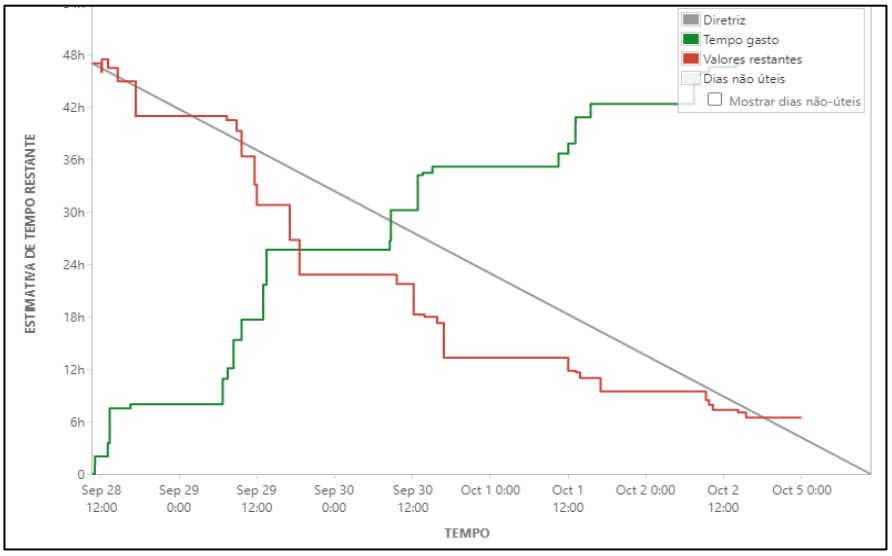


Figura 10 - Projeto 2: tempo estimado X realizado do 1º Sprint

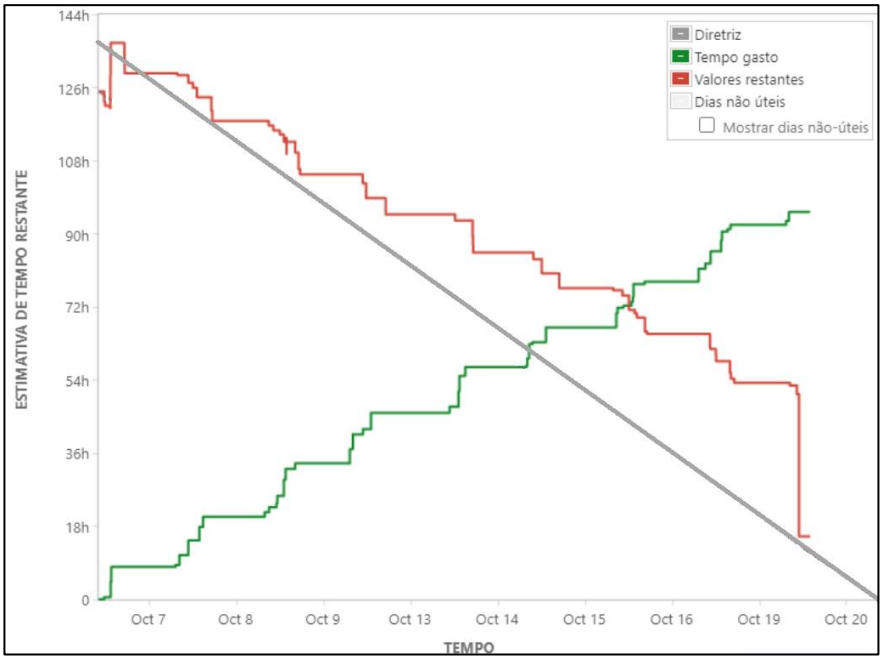


Figura 11 - Projeto 2: tempo estimado X realizado do 2º Sprint

A Figura 11 demonstra o gráfico de *burndown* do 2º Sprint. Através da linha vermelha em relação à linha cinza, verifica-se que no início da Sprint o time esteve bem alinhado ao que foi proposto, ao passar dos dias houve um pouco de atraso em relação ao estimado, contudo, o time esforçou-se para realizar a entrega no prazo e ainda foi possível finalizar o projeto com um tempo remanescente de mais de 15 horas, para uma Sprint de 15 dias.

Etapa Em Homologação

No primeiro experimento, os testes foram realizados pelo *SCRUM Master* em uma base específica para testes. O controle desta fase foi realizado no quadro Kanban através da coluna *review* e através da etiqueta identificada como homologação.

No segundo experimento, na primeira Sprint os testes foram realizados pelo *PO,* que realizou este trabalho em uma base específica para testes, e pode observar se as demandas solicitadas concordavam com o tratado com o usuário.

No segundo experimento, experimentou-se realizar os testes através da equipe de desenvolvimento, de tal forma que o programador que escreveu o código repassa a tarefa de teste para outro programador do mesmo time que não teve participação no desenvolvimento, evitando assim testes viciados de programação. Como resultado desta alternância de tarefas de testes identificou-se situações de ajustes em programas e bugs que o programador não identificaria sozinho, por estar realizando os testes de sua maneira e ainda tendo a visão da programação previamente realizada.

O controle desta fase e destas tarefas foi realizado no quadro de *Kanban* através de tarefas de testes e através da etiqueta identificada como homologação.

Em ambos os experimentos, e ao final de cada Sprint, realizaram-se reuniões de revisão. Nesta reunião foi possível coletar o *feedback* dos *stakeholders*, garantindo assim a entrega do projeto alinhada e de acordo com o esperado pelo cliente. A Sprint somente é enviada para produção após a apresentação das mudanças e a confirmação do cliente. Caso contrário, são revisados os ajustes necessários e replanejado um novo ciclo.

Etapa Concluído

No primeiro experimento decidiu-se em não realizar a documentação, pois o projeto terá uma segunda etapa que será tratada posteriormente. Assim, como ainda haverá algumas mudanças de regras e de funcionamento do programa, para evitar retrabalhos, deixou-se a etapa de documentação para um segundo momento, quando o programa estiver estável. Esta decisão foi tomada em conjunto com o analista de negócio e cliente. A Tabela 3 apresenta um resumo de cada experimento.

Tabela 3: Resumo das Informações dos Experimentos

	PRIMEIRO EXPERIMENTO	SEGUNDO EXPERIMENTO
Integrantes	3 pessoas	4 pessoas
Função de PO	Não	Sim
Lições Aprendidas	Não	Sim
Reuniões Diárias	Sim	Sim
Gráfico Burndown	Sim	Sim
Sprints	3	2
Horas estimadas	86	188
Horas efetivas	73	167
Reuniões de Revisão	Sim	Sim
Documentação técnica	Não	Sim

No segundo experimento, a documentação foi realizada pelo PO. O controle desta fase foi realizado no quadro Kanban através da coluna 'Documentação', que antecede a coluna concluído, garantindo assim que a tarefa somente seja disponibilizada ao usuário após a revisão da documentação.

Em ambos os experimentos se realizaram reuniões de retrospectiva a cada finalização de Sprint. Na reunião de retrospectiva, colheu-se o *feedback* do time de desenvolvimento e traçaram-se ações para os pontos negativos identificados, garantindo assim a evolução e o aprendizado contínuo em relação à metodologia de trabalho realizada.

Após a conclusão de todas as Sprints referenciadas ao projeto e a documentação de todos os elementos acima mencionados, o projeto finalmente é encerrado, contabilizando-se as horas de todo o processo.

Discussão da aplicação da metodologia

Através das reuniões de retrospectiva da Sprint foi possível identificar os resultados da metodologia, bem como os itens que precisavam de uma atenção para obterem melhor resultado. Estas reuniões são muito importantes, pois realizam um feedback do time e são o caminho para implementação da melhoria contínua.

Implantação da Metodologia

Antes da aplicação da metodologia houve um período de treinamento e acompanhamento de cada etapa. As reuniões de retrospectiva foram importantes para uma avaliação das tarefas executadas, bem como para o entendimento da nova metodologia. Com isso, houve uma diminuição das dúvidas em relação à metodologia de trabalho. DOBRIGKEIT e DE PAULA (2019) salientam a importância do treinamento das pessoas para uso do DT integrado com o SCRUM como um dos principais fatores de sucesso. Os autores complementam que as pessoas passam a usar a nova metodologia, bem como, aplicar a mesma em outros projetos da empresa.

O cliente interno da empresa ABC percebeu a importância de alinhar ainda mais as necessidades antes de iniciar, evitando retornos ao assunto durante a execução. Além disso, o cliente pode entender com maior facilidade o que estava sendo entregue e programar-se para repassar a informação aos departamentos envolvidos. DOBRIGKEIT e DE PAULA (2017) corroboram ressaltando a necessidade de se ter um plano para o desenvolvimento do projeto, bem como a necessidade de ser escalável. PERREIRA e DE FSM RUSSO (2018) salientam que a aproximação entre a equipe de desenvolvimento e os usuários finais é um ponto importante para o desenvolvimento de software.

Das cinco pessoas envolvidas nos experimentos, em uma reunião de retrospectiva ao final dos experimentos realizados, três pessoas comentaram a respeito da rápida adaptabilidade da metodologia, sendo que as outras duas pessoas disseram que ainda precisam de mais tempo para a adaptação.

Preparação dos Sprints

Um dos primeiros problemas apontados foi o fato de se usar no Experimento 1 a Sprint de 1 semana. Esse tempo se mostrou muito curto pelos participantes. Com isso, ajustou-se a Sprint para 15 dias. A Equipe reportou isso como algo positivo, pois possibilitou fazer ajustes necessários em meio a Sprint e ficou mais fácil de entregar no prazo.

Outro ponto percebido a respeito do Experimento 2 foi o aumento da carga de trabalho inicialmente planejada, com isso optou-se pela criação de uma terceira Sprint. Somado a isso, sugeriu-se deixar um tempo reservado para os imprevistos relativos aos eventuais *bugs* da Sprint anterior, e no final para revisões do que foi produzido. Além disto, deve-se prever uma tarefa de análise observando as necessidades ou preenchimentos obrigatórios e/ou condicionais; incluir as tarefas de reunião de planejamento, monitoramento e revisão/encerramento da Sprint. Na distribuição das tarefas deve-se verificar se haverá feriados, férias ou saídas programadas de qualquer um do time, além de verificar as dependências entre as tarefas.

Outro ponto diz respeito a priorização das tarefas em que se percebeu a necessidade de ajeitar as tarefas que são de maior importância ao cliente como prioritárias, e na sequência, as com menor importância. Aliado a isso, sugeriu-se incluir no Kamban um item para bloqueio para tarefas que possam ficar bloqueadas por conta de algum retorno do terceiro.

As estimativas usando-se o Planning Poker funcionaram adequadamente, permitindo o cumprimento dos prazos estabelecidos. Surgiu a oportunidade de ser criar um backlog de refatoração e ajustes de código legado para os casos de ociosidade não estimados.

PERREIRA e DE FSM RUSSO (2018) ressaltam que a comunicação da equipe é um ponto muito importante para o sucesso do DT integrado ao SCRUM. Sendo assim, quando se percebe um bom detalhamento das tarefas junto com uma boa estimativa com o cumprimento dos prazos, demonstra uma boa aceitação e compreensão da metodologia que está sendo implantada.

Reuniões nas Sprints

A reunião diária foi indicada como um ponto positivo e essencial para o foco e a motivação de quem está desenvolvendo o projeto. Algumas tarefas puderam ser esclarecidas na reunião de revisão, evitando assim dúvidas futuras. A alternância das tarefas de testes (um desenvolvedor programa e outro testa), possibilitou identificar situações de ajustes em programas e bugs que o programador não identificaria sozinho.

DOBRIGKEIT e DE PAULA (2019) argumentam que é necessário usar o DT para fornecer um estilo de trabalho diferente e evitar as reuniões tradicionais, dessa forma quando as reuniões diárias, de revisão e de retrospectiva agregam a motivação da equipe, o desempenho das pessoas envolvidas melhora significativamente.

STRAY et al. (2020) corroboram a necessidade de engajamento da equipe para o sucesso de uma metodologia de desenvolvimento de software. As reuniões de lições aprendidas, ao final de cada Sprint, possibilitaram a melhoria da metodologia. Percebeu-se que não foi possível executar a tarefa de atualização da documentação, pois atualmente não há uma equipe para isso. Tornou-se um trabalho muito custoso para ser realizado pelo analista ou desenvolvedor, visto que não existe documentação atual, e o programa ainda está em fase de ajustes.

Alguns aspectos técnicos de implementação e testes foram percebidos e sugeridos. Um deles foi a necessidade de padronizar a data e hora da atualização da base de testes. Os desenvolvedores tiveram dificuldade no entendimento da regra de negócio e das tabelas, bem como com dificuldade de entender o sistema legado. Os desenvolvedores reportaram a necessidade de mais tempo para análise das tarefas e compreensão dos sistemas existentes. Outro ponto notado foi que houve diversos conflitos na compilação de módulos, que demandam bastante tempo para resolução dos problemas. Por fim, o teste, por ficar a cargo dos programadores, acaba sendo mais uma revisão de código do que um teste de negócio.

Considerações finais

A aplicação de *DT* foi eficiente nas etapas iniciais de elicitação, prototipagem, e principalmente, de relacionamento com o cliente. No entanto, o método tem suas limitações no que diz respeito ao gerenciamento do tempo, desempenho e entregas do projeto. Para suprir estas limitações, utilizaram-se práticas e artefatos do SCRUM que agregaram efeitos positivos e complementaram as fases de condução desta metodologia.

De forma geral, o *DT* foi utilizado como estratégia do projeto, contribuindo para a definição do escopo e compreensão das necessidades do usuário final, e o SCRUM foi utilizado para a implementação do projeto, através da abordagem dos prazos, entregas em etapas e priorização dos trabalhos. A combinação destas abordagens ofereceu uma metodologia enxuta, focada em resultados e no entendimento das necessidades do cliente.

Para o correto funcionamento do *framework* é imprescindível que a empresa ou departamento siga e acredite nos valores do manifesto ágil, do contrário, é insuficiente implementar os ritos do SCRUM, realizar a divisão de papéis, utilizar artefatos como o *Kanban* ou demais práticas indicadas pelo gerenciamento ágil.

Para este experimento realizou-se um treinamento de 30 minutos. Notou-se que este tempo foi suficiente quando ministrado a uma equipe pequena de até quatro pessoas, para treinamentos em equipes maiores, entende-se que haverá a necessidade de um tempo superior, visto as dúvidas que surgirão durante e após a explanação.

Através das ferramentas Jira Software e ActiveCollab foi possível ver, de forma clara e simplificada, o planejamento do aumento de demandas na empresa. Dentre as duas ferramentas experimentadas, o Jira Software mostrou-se mais completo e preparado para trabalhar com gerenciamento de projetos de *software*, possui a criação de versionamentos, quadro Kanban, quadro SCRUM, e muitos relatórios e gráficos de acompanhamento, inclusive o gráfico de *burndown*.

Verificou-se que a estimativa do tempo via *Planning Poker* foi bem aceita pelos times e em algumas situações auxiliou a repensar no planejamento dos tempos. Esta técnica, utilizada junto do gráfico de *burndown*, pôde medir o

progresso em relação às tarefas e assim garantir maior assertividade no planejamento do projeto.

Outro item importante a ser ressaltado é que a Sprint quinzenal demonstrou maior facilidade de planejamento e menor pressão de trabalho ao time, concluindo-se assim que este tempo de 15 dias é mais adequado para o método.

Durante o desenvolvimento das tarefas, o fato de passar a responsabilidade de testes para um programador que não tenha desenvolvido a solução, ajudou na identificação precoce de erros que antes não seriam encontrados, resultando num ganho de qualidade no software entregue ao cliente.

O aumento na concentração e no foco foi identificado através das etapas de análise e nos relatos das reuniões diárias. Com as tarefas chegando bem analisadas aos programadores e com a aproximação da equipe, houve maior controle da execução do projeto, cumprimento dos prazos e aumento de produtividade.

Pode-se observar que os experimentos foram realizados com uma equipe pequena com o intuito de validar o método com um grupo restrito inicialmente, especialmente no primeiro experimento. No segundo experimento inclui-se o papel de Dono do Produto (PO), passou-se a analisar as lições aprendidas e a documentação técnica voltou a ser produzida por uma necessidade levantada durante o primeiro experimento. O número de horas estimadas mostrou-se muito assertivo nos dois experimentos.

Durante a execução do experimento identificaram-se dificuldades que deverão ser sanadas para uma expansão bem-sucedida do método. As sugestões de resolução para cada tópico identificado são:

Utilização do versionamento de código com *branches* - O lançamento de uma versão do programa não pode bloquear a correção de erros. Com a utilização dos *branches*, cada equipe trabalharia numa ramificação independente, diminuindo a ocorrência de conflitos e a possibilidade de atrasos.

Base de dados de testes - Sugere-se automatizar o procedimento de atualização da base de dados de testes, de forma que a atualização ocorra sempre no mesmo dia do mês, para que a carga dos dados não atrapalhe os testes que estão em execução.

Usar a lista de projetos atuais como *backlog* - Todos as tarefas devem ser detalhadas na inclusão, os que não forem detalhados perdem prioridade no *backlog*.

Aumentar o número de responsáveis pela alteração de tabelas - Hoje apenas uma pessoa realiza esta tarefa, sugere-se indicar uma pessoa por time, aumentando a autonomia da equipe e diminuindo os gargalos durante a execução.

Realizar uma reestruturação do setor - Será necessário montar os times, definir os produtos que cada um ficará responsável, definir os papéis de cada integrante e separar o suporte ao usuário do desenvolvimento.

Ao final do experimento, através da pesquisa aplicada, verificou-se que os envolvidos nos experimentos sentiram-se motivados em utilizar a metodologia proposta. O aumento na motivação dos programadores foi um dos objetivos pelo qual se optou pelo uso de uma metodologia ágil. Identificou-se também que houve um bom entendimento da complexidade dos projetos, isso auxiliou na assertividade da estimativa de prazos e automaticamente fez com que a satisfação dos clientes aumentasse. Estes aspectos também possibilitarão realizar a estimativa de custos do projeto, quando for necessário.

De forma geral, a contribuição deste trabalho foi de extrema importância para a empresa rever conceitos, métodos de trabalho, e até mesmo tópicos ligados a engenharia de software. A equipe inicial deste trabalho mostrou-se engajada e receptiva a melhorias e o resultado do primeiro experimento estimulou os gestores do setor a expandir e aplicar o método proposto.

Através da oportunidade de expansão, realizou-se o segundo experimento para uma equipe ainda maior. Esta nova equipe demonstrou uma boa maturidade em relação à busca de resultados, e o método foi de fato aplicado para o time, o qual vem trabalhando na busca de melhoria contínua.

Com as contribuições deste trabalho, a empresa está buscando adaptar-se às sugestões do método proposto, visando realizar a expansão da metodologia aos demais times, auxiliando não somente no gerenciamento de projetos como no controle e no gerenciamento do setor em sua totalidade.

Ajustes na área de TI após os experimentos realizados

Com os experimentos realizados internamente, percebeu-se um grande desafio em expandir a metodologia a todo o setor. Isso ocorre, pois, o SCRUM é uma metodologia revolucionária. Para mitigar este efeito de revolução, decidiu-se utilizar o Kanban como base. O Kanban propõe de modo geral realizar o acompanhamento do trabalho em andamento e o acompanhamento do lead time (tempo necessário de atravessamento de uma tarefa). Isso relacionado aos ritos e procedimentos do SCRUM e do DT já mencionados nesse trabalho.

Neste momento, a empresa decidiu contratar mais desenvolvedores e pessoas para realizar atendimento de usuários. Houve a expansão do uso dessa metodologia de trabalho para todos os times do setor de desenvolvimentos de TI.

A empresa está em fase de negociação de compra da ferramenta Jira Software e Jira Service Management (Service Desk), providenciando a utilização de versionamentos em *branches* integrado ao software Gitlab.

Também está se utilizando a lista atual de projetos como backlog e realizando uma mudança bem contundente: separação do suporte ao usuário do desenvolvimento com a criação de níveis de atendimento (nível 1, nível 2 e nível 3) atrelados a nova ferramenta de Service Desk.

As modificações mencionadas na equipe de TI foram identificadas a partir dos experimentos realizados e das sugestões mencionadas neste artigo. Como principais resultados da utilização desta metodologia de gerenciamento de projetos, os times relataram:

- Com uma melhor visualização do trabalho é possível identificar os gargalos, e o que precisa de maior atenção para o fluxo fluir;
- O limite de trabalho faz com que o time a finalize o que está em andamento;
- As reuniões diárias geram um maior engajamento da equipe e conhecimento do que está acontecendo;
- A reunião de retrospectiva auxilia e promove a melhoria contínua;
- A revisão de código-fonte auxilia a padronizar o código e a resolver problemas antes do envio ao cliente;
- A fase de homologação também auxilia na captação de falhas;

- Hoje ocorre um melhor acompanhamento da execução das tarefas e seu tempo;
- Tem-se uma maior percepção do trabalho entregue.

Referências

ANDRADE, A. J. F. *et al. Gestão de Projeto com SCRUM: Um Estudo de Caso*. Aracati, CE: Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE), 2011.

BECK, K. *The agile manifesto*. 2001. https://agilemanifesto.org/

CAMARGO, R.; RIBAS, T. *Gestão ágil de projetos*. 1. ed. São Paulo, SP: Saraiva Educação, 2019.

COHN, M. *Desenvolvimento de software com SCRUM: aplicando métodos ágeis com sucesso*. 1. ed. Porto Alegre, RS: Bookman, 2011.

CORRAL, L.; FRONZA, I. Design thinking and agile practices for software engineering: an opportunity for innovation. In: *Proceedings of the 19th Annual SIG Conference on Information Technology Education*. 2018. p. 26-31.

DOBRIGKEIT, F.; DE PAULA, D. The best of three worlds-the creation of InnoDev a software development approach that integrates Design Thinking, SCRUM and lean startup. In: *DS 87-8 Proceedings of the 21st International Conference on Engineering Design (ICED 17)* Vol 8: Human Behaviour in Design, Vancouver, Canadá, 21-25.08. 2017. 2017. p. 319-328.

DOBRIGKEIT, F.; DE PAULA, D. Design Thinking in practice: understanding manifestations of DT in software engineering. In: *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering*. 2019. p. 1059-1069.

DOBRIGKEIT, F.; DE PAULA, D.; UFLACKER, M. InnoDev: a software development methodology integrating Design Thinking, SCRUM, and lean startup. In: *DT Research. Springer*, Cham, 2019. p. 199-227.

FERRARI, A. T. *Metodologia da ciência*. 2. ed. Rio de Janeiro: Kennedy, 1974. Cap. 2.

FOGGETTI, C. *Gestão ágil de projetos*. São Paulo, SP: Pearson Education do Brasil, 2014.

GALLOTTI, G. M. A. *Arquitetura de software*. São Paulo, SP: Pearson Education do Brasil, 2016.

GIL, A. C. *Métodos e técnicas de pesquisa social*. 6. ed. São Paulo: Atlas, 2008.

KERZNER, H. R. *Gerenciamento de projetos: uma abordagem sistêmica para planejamento, programação e controle*. 2. ed. New York, NY: Edgard Blücher, 2015.

LAKATOS, E. M.; MARCONI, M. A**.** *Fundamentos de metodologia científica*. 8. ed. São Paulo, SP: Atlas, 2017.

LI, P. *Jira Software Essentials: Plan, track, and release great applications with Jira Software*. Packet Publishing Ltd, 2018.

MAKIOSZEK, A. A. *Organização, sistemas e métodos (OSM) e design organizacional: novas práticas*. Curitiba, PR: Intersaberes, 2019.

MASSARI, V. L. *Gerenciamento Ágil de Projetos*. 8. ed. São Paulo, SP: Brasport, 2018.

MORAES, I. C. et al. DT Transformando a Cultura Organizacional. In: *Congresso internacional de conhecimento e inovação*, 2019, Porto Alegre. *Anais* (...). Porto Alegre: Ciki, 2019.

ÖZKAN, D.; MISHRA, A. Agile Project Management Tools: A Brief Comprative View. *Cybernetics and Information Technologies*, v. 19, n. 4, p. 17-25, 2019.

PERREIRA, J. C.; DE FSM RUSSO, R. *Design Thinking integrated in agile software development: A systematic literature review*. Procedia computer science, v. 138, p. 775-782, 2018.

PHAM, A.; PHAM, P. *SCRUM em ação*. Novatec Editora, 2011.

PHAM, A.; PHAM, P. *Business-Driven IT-Wide Agile (SCRUM) and Kanban (Lean) Implementation: An Action Guide for Business and IT Leaders*. Productivity Press, 2019.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de software: uma abordagem profissional*. 8. ed. Porto Alegre, RS: AMGH, 2016.

SCHWABER, K.; E SUTHERLAND, J. *Guia do SCRUM*. Estados Unidos: Creative Commons, 2017.

SENAPATHI, M.; DRURU-GROGAN, M. L. *Systems thinking approach to implementing kanban: A case study*. *Journal of Software: Evolution and Process*, v. 33, n. 4, p. e2322, 2021.

STRAY, V. et al. *Agile Processes in Software Engineering and Extreme Programming: 21st International Conference on Agile Software Development, XP 2020*, Copenhagen, Denmark, June 8–12, 2020, Proceedings. Springer Nature, 2020.

VIANNA, M. et al. *Design Thinking: inovação em negócios*. 2. ed. Rio de Janeiro, RJ: MJV Press, 2018.

XAVIER, C. M. S. *Gerenciamento de projetos: como definir e controlar o escopo do projeto*. 3. ed. São Paulo, SP: Saraiva, 2016.