UNIVERSIDADE ESTADUAL DE
MATO GROSSO DO SUL

Computação, Licenciatura

Luís Felipe de Oliveira Melle

MIND'S - Uma Ferramenta de Suporte ao Gerenciamento de Manutenção Industrial Apoiada por Scrum

UEMS
2017

Luís Felipe de Oliveira Melle

MIND'S - Uma Ferramenta de Suporte ao Gerenciamento de Manutenção Industrial Apoiada por Scrum

Orientador: Prof. Me. Jorge Marques Prates

Monografia apresentada a Universidade Estadual de Mato Grosso do Sul – UEMS, para o Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Licenciando em Computação.

UEMS
2017

Luís Felipe de Oliveira Melle

MIND'S - Uma Ferramenta de Suporte ao Gerenciamento de Manutenção Industrial Apoiada por Scrum

Monografia apresentada a Universidade Estadual de Mato Grosso do Sul – UEMS, para o Trabalho de Conclusão de Curso, como parte dos requisitos para obtenção do título de Licenciando em Computação.

Membros da banca:

Prof. Me. Jorge Marques Prates (Orientador)
UEMS – Nova Andradina

Prof. Me. Eduardo Machado Real
UEMS – Nova Andradina

Prof. Me. Márcio Demetrius Martinez
UEMS – Nova Andradina

Dedico este trabalho à minha família.

Melle, Luís Felipe de Oliveira
MIND'S – Uma Ferramenta de Suporte ao Gerenciamento de Manutenção Industrial Apoiada por Scrum / Luís Felipe de Oliveira Melle – Nova Andradina, 2017.
62 f. ; 30cm.

Orientador: Jorge Marques Prates
Trabalho de Conclusão de Curso (TCC) Curso de Computação, Licenciatura. UEMS – Universidade Estadual de Mato Grosso do Sul. Unidade de Nova Andradina. Nova Andradina–MS. 2017.

1. Métodos Ágeis. 2. Scrum. 3. Gerenciamento de Manutenção. 4. Manutenção Industrial.

CDU – _____

Agradecimentos

Ao Professor Jorge pelos “puxões de orelha”, paciência, ensinamentos e orientação.
Ao Professor Eduardo pelo apoio, auxílio e ensinamentos.
Ao Professor Márcio pelos ensinamentos e principalmente pela paciência.
Ao Professor Fábio pelo apoio e ensinamentos.
Aos poucos membros da turma de Computeiros que se mantiveram unidos até o fim do curso.
À CAPES e ao projeto PIBID pelo apoio financeiro.

“O mundo não é um grande arco-íris.”

Resumo

Métodos ágeis é o termo utilizado para se referir aos novos métodos criados para suprir as dificuldades encontradas nos métodos tradicionais de desenvolvimento de software, como a demora para conclusão de um projeto, ou até mesmo a desistência do mesmo. Tais métodos são mais eficientes que os tradicionais e podem se adaptarem com facilidade à mudanças durante o projeto. Os métodos ágeis são flexíveis e podem serem utilizados em contextos diferentes do desenvolvimento de software. Uma possível aplicação é no gerenciamento de manutenção de indústrias, auxiliando no monitoramento dos equipamentos e mantendo o seu histórico de falhas e de manutenções. Um grande problema existente em várias indústrias é a falta do gerenciamento de manutenção, que pode acarretar na falha inesperada dos equipamentos. Quando isso ocorre em uma indústria é algo muito prejudicial, pois pode gerar grandes perdas de patrimônio ou até mesmo do equipamento. Este trabalho investiga a aplicação de métodos ágeis na criação de uma ferramenta (MIND'S) que apoia o gerenciamento de manutenções industriais, que por sua vez, também é baseado nesses métodos. Para a validação da MIND's, a mesma foi apresentada em uma empresa em que o gerenciamento de manutenção não é aplicado. Durante a validação foi detalhado o funcionamento da ferramenta e como ela pode apoiar o gerenciamento da manutenção. Os dados da avaliação foram obtidos por meio de uma pesquisa de opinião, onde os participantes responderam à questões sobre a ferramenta. Os resultados foram positivos, sendo apenas necessárias algumas adaptações específicas de acordo com a particularidade de cada indústria.

Palavras–chave: Métodos Ágeis, Scrum, Gerenciamento de Manutenção.

Abstract

Agile Methods is the term used to refer to new methods created to supply the difficulties encountered in traditional methods of software development, as the delay to conclusion of a project, or even the quitting of foregoing. Such methods are more efficients than traditional and they can adapt easily to changes during the project. The agile methods are flexibles and they can be used in different contexts of software development. A possible application is in maintenance management of industry, assisting in the monitoring of equipment and keeping your failure history and maintenance.

A large problem existing in a various industries is the lack of maintenance management, that can entail on unexpected equipment failure. When this occurs in an industry is something very damaging, because it generate large damage losses of equity or even of equipment.

This work investigates the application of agile methods in the creation of the tool (MIND'S) that supports the industrial maintenance management, which in turn, is also based in this methods.

For a validity of MIND'S, it was presented in a company where the maintenance management is not applied. During a validation was detailed how the tool works and how it can support maintenance management. The evaluation data were obtained through an opinion survey, where participants answered questions about the tool.

The results were positive, and only a few specific adaptations were needed according to the particularity of each industry.

Keywords: Agile Methods, Scrum, Maintenance Management.

Lista de Figuras

Lista de Tabelas

Sumário

CAPÍTULO 1

Introdução

1.1 Contexto

A manutenção industrial pode ser definida como uma série de ações que são executadas para manter um equipamento ou peça operável, maximizando sua vida útil. A manutenção se tornou um fator essencial para uma empresa se manter competitiva e com qualidade no mercado, pois ela é de extrema importância para o capital da empresa. E quanto mais eficaz for a gerência da manutenção, melhor será para o capital, pois o mesmo ficará mais protegido (Silveira, 2012).

A gerência de manutenção em algumas empresas é algo precário, pois acabam focando apenas no lucro e deixam a gestão de lado. Por não terem um histórico das manutenções executadas ou de futuras, pode ocorrer uma falha grave em algum equipamento, acarretando um grande prejuízo para empresa. Com uma boa gestão da manutenção é possível ter uma base dos dados contendo todos os serviços, podendo assim prevenir as pausas, gastos inesperados e desperdício de tempo. Além do gerenciamento, também é possível diminuir despesas e tempo ocioso usando estratégias de organização, onde as mesmas ajudaram na hora de relatar todo o planejamento (Xavier e Dorigo, 2005).

Para amenizar tais problemas de gerenciamento de manutenção, é proposto neste trabalho uma ferramenta para auxiliar no controle das manutenções baseadas nas **Metodologias Ágeis**. As metodologias ágeis surgiram devido aos resultados pouco eficientes decorrentes da aplicação de métodos tradicionais, além da sua inflexibilidade e inaptidão de se adaptar à mudanças. Pensando nesses problemas que estavam enfrentando, pesquisadores se reuniram para unir suas experiências de desenvolvimento e encontrarem um meio que minimizasse tais problemas, com isso propuseram as metodologias ágeis (Pressman, 2005).

1.2 Motivação

A ausência do gerenciamento de manutenção é prejudicial às empresas e indústrias, pois com sua inexistência as mesmas ficam vulneráveis à falhas inesperadas dos equipamentos. Quando acontece

uma dessas falhas a empresa acaba gastando mais recursos para tudo voltar ao normal, mas até que tudo se estabilize a produção vai estar parada, causando assim mais perdas.

Visando solucionar alguns problemas devido à métodos pouco eficientes de gerenciamento da manutenção industrial, além das percas que isso gera, é proposta uma ferramenta genérica para auxiliar não só no gerenciamento da manutenção, mas também nos controle de equipamentos presente na indústria, nos funcionários e nas ordens de serviço.

Ao desenvolver a ferramenta, foram utilizados os princípios das metodologias ágeis, pois segundo Sene (2010) os métodos tradicionais apresentam resultados desagradáveis quando se trata de tempo de desenvolvimento e conclusão do projeto. Essa nova metodologia propõe uma nova forma de se trabalhar, proporcionando assim melhores resultados caso ocorra problemas no meio do projeto e em questão de tempo de desenvolvimento.

1.3 Objetivos

Este projeto tem como objetivo propor uma ferramenta para auxiliar na gerência de manutenção industrial, em que a ferramenta irá apoiar a adaptação de técnicas Scrum em atividades de gerenciamento e manutenção industrial. A ferramenta contribui para o problema de ausência no gerenciamento de manutenção e possibilitará a pesquisa de equipamentos e ordens de serviço para analisar as manutenções executadas e as futuras manutenções.

1.4 Organização

Neste capítulo foi apresentado uma visão geral da monografia, ressaltando o contexto no qual o trabalho foi inserido, as principais motivações e os objetivos esperados com sua execução.

No Capítulo 2 apresenta-se uma visão geral sobre os métodos ágeis, relatando seu surgimento e seus princípios. A partir do conteúdo apresentado, serão tratados alguns métodos ágeis de forma detalhada.

No Capítulo 3 apresenta-se uma explicação detalhada do método ágil Scrum. Principais conceitos, seus valores, eventos e tambémd o uso além do desenvolvimento de software, em que são apresentados alguns trabalhos.

No Capítulo 4 apresenta-se uma visão geral das funcionalidades da ferramenta proposta para amenizar problemas presentes no gerenciamento de manutenção industrial.

No Capítulo 5 apresenta-se as considerações finais desta monografia.

CAPÍTULO 2

Métodos Ágeis

2.1 Considerações Iniciais

O surgimento dos métodos ágeis aconteceu para tentar melhorar os resultados de desempenho do desenvolvimento e gerenciamento de software, pois os métodos tradicionais apresentavam resultados desagradáveis (Sene, 2010). O método tradicional é criticado pela sua inflexibilidade de se adaptar a esta nova realidade, já o gerenciamento ágil vem ganhando espaço e popularidade, em razão da sua capacidade de adequação à situações do ambiente (Ribeiro et al., 2006).

2.2 Conceitos Básicos

Segundo Pressman (2005) os métodos ágeis foram desenvolvidos para vencer as dificuldades encontradas nas metodologias tradicionais de desenvolvimento de software.

Visto que os autores mais atuais concordam que o método tradicional de gerenciamento de projetos apresenta vários problemas e desvantagens, começaram a utilizar e pesquisar mais sobre os métodos ágeis, para poder aplicar em seus projetos e ao mesmo tempo aperfeiçoar esses métodos que ainda é algo recente. Porém, não é de hoje que se tem essa insatisfação com o método tradicional devido à atrasos dos projetos e até mesmo cancelamento por não conseguirem terminar o projeto.

Diante destes problemas, os mesmos tinham que ser resolvidos de alguma forma, foi então que um grupo de profissionais decidiu se reunir para discutir formas de melhorar o desempenho de seus projetos. Com base nisto, surgiu o termo Desenvolvimento Ágil (Teles, 2008).

Segundo Santos Soares (2004), o termo metodologia ágil tornou-se popular em 2001 quando alguns desenvolvedores de software e representantes de alguns métodos como *Extreme Programming* (XP), Scrum, DSDM e *Crystal* se reuniram para estabelecerem princípios que ambos compartilhavam em comum. O resultado desta reunião foi a criação da Aliança Ágil e a estipulação do Manifesto Ágil.

O Manifesto Ágil possui quatro valores indispensáveis, que são (Beck et al., 2001):

1. **Indivíduos e interação entre eles** mais que processos e ferramentas;

2. **Software em funcionamento** mais que documentação abrangente;
3. **Colaboração com o cliente** mais que negociação de contratos;
4. **Responder a mudanças** mais que seguir um plano.

A partir do Manifesto os autores também criaram doze princípios a serem seguidos. Não existe um mais importante que o outro, pois trabalhando eles em conjunto que o progresso e o sucesso de um projeto que aplica metodologias ágeis será alcançado. Os princípios do Manifesto ágil são (Beck et al., 2001):

- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de software de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- Software funcional é a medida primária de progresso.
- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
- Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis.
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.

2.3 *Extreme Programming* (XP)

O uso da metodologia XP é indicado para equipes pequenas ou médias, que desenvolvem softwares a partir de requisitos vagos e apresentam modificações constantes. Diferente das metodologias tradicionais, a XP propõe que na sua utilização a equipe faça o uso de feedback constante, abordagem incremental e comunicação direta entre a equipe e o cliente (Koscianski e Santos Soares, 2007).

O método XP faz muito sucesso, pois ajuda a produzir um sistema de melhor qualidade e mais econômico do que o habitual, este método consegue suprir todas essas necessidades, pois segue os princípios e práticas do Manifesto Ágil. Os programadores adotam um método chamado programação em par (*pair programming*), ou seja, trabalham sempre em duplas, pois com isso um auxilia o outro para desenvolver e complementar o código.

Com a programação em par o programa desenvolvido acaba sendo mais ágil e mais difícil de conter erros e o código é sempre testado conforme é desenvolvido e também, o código é aberto, ou seja, outros programadores também podem ter acesso a ele, com isso o código pode ficar ainda melhor. O cliente sempre vai fazer parte do desenvolvimento do projeto com várias reuniões presencias, pois com o cliente presente ele consegue explicar melhor os requisitos do sistema e ver como está ficando. É aconselhável que tenha reuniões diariamente para mostrar o andamento do projeto (Teles, 2008).

2.3.1 Valores XP

As metas individuais de curto prazo muitas vezes conflitam com metas sociais de longo prazo. As sociedades aprenderam a lidar com esse problema desenvolvendo conjuntos compartilhados de valores, apoiados por mitos, rituais, punições e recompensas. Sem esses valores, os seres humanos tendem a reverter para seu próprio interesse a curto prazo (Beck e Andres, 2004).

Os valores do XP são um conjunto de características em que os participantes do projeto devem seguir para o bom andamento do projeto. Esta metodologia é composta por quatro valores muito importantes como a comunicação, simplicidade, *feedback* e coragem. Beck e Andres (2004) fazem a descrição dos quatro valores do método XP:

- **Comunicação:** Os problemas em projetos podem ser analisados e, muitas vezes, chega-se à conclusão de que os mesmos ocorreram devido à ausência de comunicação de algo importante que aconteceu. Por exemplo, um programador realiza alguma mudança crítica no projeto ou não tem a devida comunicação com o cliente; ao do final do projeto o cliente se decepciona com o resultado, pois não era o que ele esperava.

  A XP visa manter a comunicação fluindo através de várias práticas que não podem ser feitas sem a mesma. Mesmo com essas práticas, as pessoas ficam com medo, se distraem e acabam errando. Para esses casos, uma pessoa fica responsável por ser um "treinador" que vai auxiliar as outras quando perceber a falta de comunicação.

- **Simplicidade:** O "treinador" pergunta à equipe: "Qual é a coisa mais simples que poderia funcionar ?". É mais válido criar algo simples e com pouco esforço e depois melhorá-lo, do que criar algo complexo e com funcionalidades extra de imediato. Complexidade e funcionalidades desnecessárias acabam causando uma complicação na interpretação do código e, assim, o

mesmo fica mais vulnerável a erros. Esses erros causam atrasos no projeto e o desvia de seu principal objetivo.

- ***Feedback*:** É inestimável o *feedback* constante sobre o estado atual do sistema, pois fica mais fácil a equipe encontrar problemas no projeto e solucioná-los. Quando os problemas são encontrados e solucionados prematuramente, não irão atrapalhar tanto quanto em uma fase final do sistema.

- **Coragem:** Na XP a coragem é essencial principalmente quando o cliente solicita algo novo com o projeto em andamento, pois a equipe tem que se sentir segura e confiante para realizar as mudanças necessárias. A equipe procura sempre ouvir o cliente e se adaptar as mudanças que é requisitada.

2.3.2 Regras e Práticas da XP

As práticas do XP são tidas como boas práticas na Engenharia de Software e reforçam os valores citados acima. Todas práticas são aplicadas de forma extrema e sempre visando a qualidade e os prazos de entrega do projeto. Pressman (2005) explica que o XP é composto por uma estrutura de regras e práticas com as seguintes atividades:

A Figura 2.1 abaixo demonstra o processo das práticas.

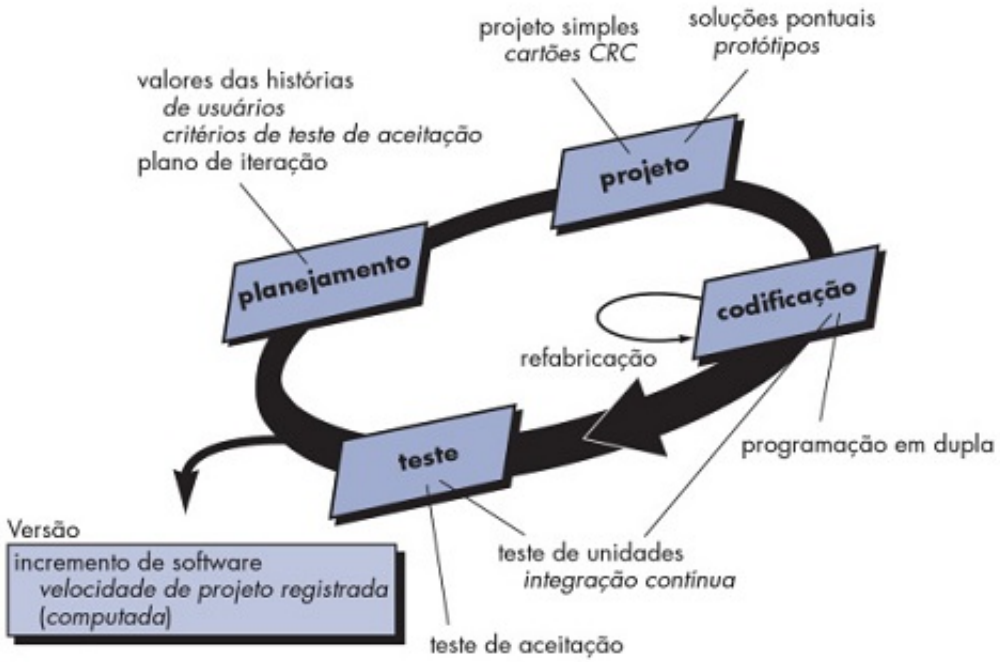

Figura 2.1: Visão geral do processo de desenvolvimento Extreme Programming, extraído de Pressman (2005)

- **Planejamento:** Para iniciar a atividade de planejamento deve ser criado um conjunto de histórias, que nada mais é do que a descrição das funcionalidades e características requeridas para que o software possa ser construído. Esta história é feita pelo cliente e é colocada em um cartão

de indexação. É atribuído uma prioridade (valor) as histórias, descritas pelo próprio cliente. Os membros da equipe XP avaliam o que foi requisitado e atribuem um custo de acordo com o número de semanas que serão gastas para o desenvolvimento.

A equipe XP determina quais histórias serão desenvolvidas primeiro em um dos três modos:

Quando a equipe conclui a primeira versão, entregam ao cliente para o mesmo analisar o andamento do projeto. Após ser entregue, a equipe pode calcular a velocidade do andamento do projeto. Assim, a equipe pode estimar datas de entrega de versões subsequentes.

À medida em que se desenvolve o projeto, o cliente pode adicionar, alterar o valor ou até mesmo eliminar histórias que já não é mais tão importante. A equipe então reconsidera as versões que estão por vir e modificam seus planos para conseguirem cumprir os prazos.

- **Projeto:** O projeto segue a ideia KIS (*Keep it simple* - mantenha a simplicidade). O projeto deve seguir conforme foi descrito. A XP encoraja o uso de cartões CRC (*Class Responsibility Card*) que nada mais é do que é uma coleção de cartões que representam classes. O cartão é dividido em 3 partes, ilustrado na Figura 2.2, no alto do cartão é escrito o nome da classe. No corpo do cartão são listadas as responsabilidade da classe que são os ´´problemas" a serem resolvidos e os colaboradores à direita, ou seja, quem vai colaborar para que as responsabilidades sejam cumpridas.

  Caso um grave problema de projeto for encontrado, recomenda-se que se crie imediatamente um protótipo operacional desta parte, a fim de reduzir os riscos quando iniciar a implementação do projeto real.

Classe: PlantaBaixa	
Descrição	
Responsabilidade:	**Colaboração:**
Define o nome/tipo da planta baixa	
Gerencia posicionamento na planta baixa	
Muda a escala da planta baixa para exibição	
Muda a escala da planta baixa para exibição	
Incorpora paredes, porta e janelas	**Parede**
Mostra a posição das câmeras de vídeo	**Câmera**

Figura 2.2: Modelo de cartão CRC, extraído de Pressman (2005).

- **Codificação:** Após toda história desenvolvida e o trabalho preliminar de projeto finalizado, a equipe não deve iniciar o código e sim desenvolver uma série de testes unitários para cada histórias da versão atual. Quando acabar a criação os testes o desenvolvedor estará melhor preparado para iniciar a codificação e passar nos testes criados. Terminando a codificação o desenvolvedor pode realizar o teste e fornecer um feedback para os outros desenvolvedores.

Um conceito muito importante da XP é a programação em pares, onde duas pessoas trabalham juntas no desenvolvimento do projeto (duas cabeças pensam melhor do que uma). Apesar de trabalharem juntas assumem papéis diferentes. Conforme os pares de programadores terminam seu trabalho, o que foi desenvolvido é integrado ao trabalho dos outros programadores.

- **Teste:** A criação de testes antes da codificação é um elemento-chave da XP. Os testes devem ser criados de forma automatizada, permitindo que seja executado de maneira rápida e repetidamente. Também existem os testes de aceitação, onde o cliente irá testar o software, estes testes são derivados das histórias do usuário que foram implementadas como parte de uma versão do software

2.4 *Feature Driven Development* (FDD)

Criado por Peter Coad e sua equipe em 1997, o FDD é um modelo prático de processo para Engenharia de Software orientada a objetos. Em seguida o trabalho de Coad foi estendido e melhorado por Stephen Palmer e John Felsing que descreveram um processo adaptativo e ágil que pode ser usado em projetos de tamanho moderado ou grande.

Segundo Pavan Kumar (2009), como outras metodologias ágeis, o FDD recomenda que o desenvolvimento busque as melhores práticas e o monitoramento frequente. O FDD é um aprimoramento de abordagens iterativas e incrementais. Algumas práticas do método XP são utilizadas no FDD, como testes unitários, programação em par, integração continua, entre outras.

2.4.1 Papéis e Responsabilidades

O FDD é dividido em basicamente três papéis: **papéis chave** que são responsáveis por desenvolver a aplicação, **papéis de apoio** responsáveis pela criação de programas que auxiliam os demais desenvolvedores a produzir o produto final e **papéis adicionais** que são papéis não obrigatórios e geralmente utilizado em grandes equipes. Vários participantes podem receber diversos papéis, assim podendo ampliar uma função ou repartir responsabilidades.

Os **papéis chave**, são: **Gerente de Projeto**, **Arquiteto Chefe**, **Programados Chefe**, **Gerente de Desenvolvimento**, **Dono de Classe** e **Especialista de Negócios**. Os **papéis de apoio** no desenvolvimento são: **Gerente de Versão**, **Engenheiro de Integração**, **Tool Smith** e o **Guru de Linguagem**. E por fim, os **papéis adicionais** são **Implantadores**, **Escritores Técnicos** e **Testadores**. Todos os papéis desenvolvem um papel vital no FDD (Goyal, 2007).

2.4.2 Práticas do FDD

Visto que as metodologias analisadas possuem suas práticas, no FDD possui 8 práticas a serem seguidas (Goyal, 2007):

1. **Modelagem de Objetos de Domínio:** Consiste na construção de diagramas de classes de acordo com os objetos do sistema, podendo assim verificar possíveis soluções e aprimorar o entendimento das funcionalidades.

2. **Desenvolvendo por Funcionalidade:** Para produzir o que os clientes têm em vista é usado o método de desenvolver e entregar por funcionalidade, assim o progresso do projeto é guiado por seus requisitos funcionais com valor de negócio e em tamanhos pequenos.

3. **Posse Individual de Classe/Código:** Os desenvolvedores recebem uma ou mais classes que são de propriedade individual, ou seja, esta técnica ajuda a evitar problemas com vários desenvolvedores alterando o mesmo código e aumentando a responsabilidade do programadores em posse da classe.

4. **Equipes de Funcionalidade:** A implementação de uma função do projeto pode envolver mais de uma classe, isto é, mais de um Proprietário de Classe envolvido na mesma função. Com isso, é preciso alguém para gerenciar este trabalho, para o mesmo coordenar o esforço dos diversos desenvolvedores.

5. **Inspeções:** As inspeções proporcionam trocas de conhecimento, diminui o uso de códigos de baixa qualidade, devido ao fato de que os desenvolvedores sabem que o código será analisado mais tarde.

6. **Versões com Regularidade:** Permite aos desenvolvedores sempre ter algo novo para mostrar ao cliente, proporcionando um *feedback* melhor quanto ao andamento do projeto, além de permitir resolver e identificar problemas de integração.

7. **Gerenciamento de Configurações:** Tem o intuito de controlar as versões para todos os artefatos do projeto e suas modificações.

8. **Relatórios de Progresso:** Ao longo do projeto é acompanhado a evolução do mesmo e feito relatórios, com base no que já foi concluído.

   Cada prática tem sua vantagem, porém se usadas todas juntas a força da metodologia aumenta.

2.4.3 Processos do FDD

O FDD consiste em 5 processos na fase de projeto e desenvolvimento que é ilustrados na Figura 2.3 e descritos abaixo (Retamal, 2008):

- **Desenvolver um Modelo Abrangente:** Neste processo todo o projeto é abrangido, envolvendo membros do domínio do negócio e os desenvolvedores. São realizados estudos detalhados a respeito do domínio do negócio para cada área a ser modelada. Após os estudos, são formados pequenos grupos pelos membros do domínio que estava sendo estudado e desenvolvedores, que farão seus próprios modelos que satisfaçam o domínio em questão. Os modelos são apresentados, revisados por parceiros e discutidos. Algum modelo, ou alguns modelos são escolhidos, tornando-se um modelo para aquela área do negócio.

- **Construir a Lista de Funcionalidades:** É um processo cujo objetivo é identificar todas as funcionalidades que estão de acordo com os requisitos. Uma equipe com programadores chefes iram decompor os requisitos de negócio e assim será formado uma lista de funcionalidades.

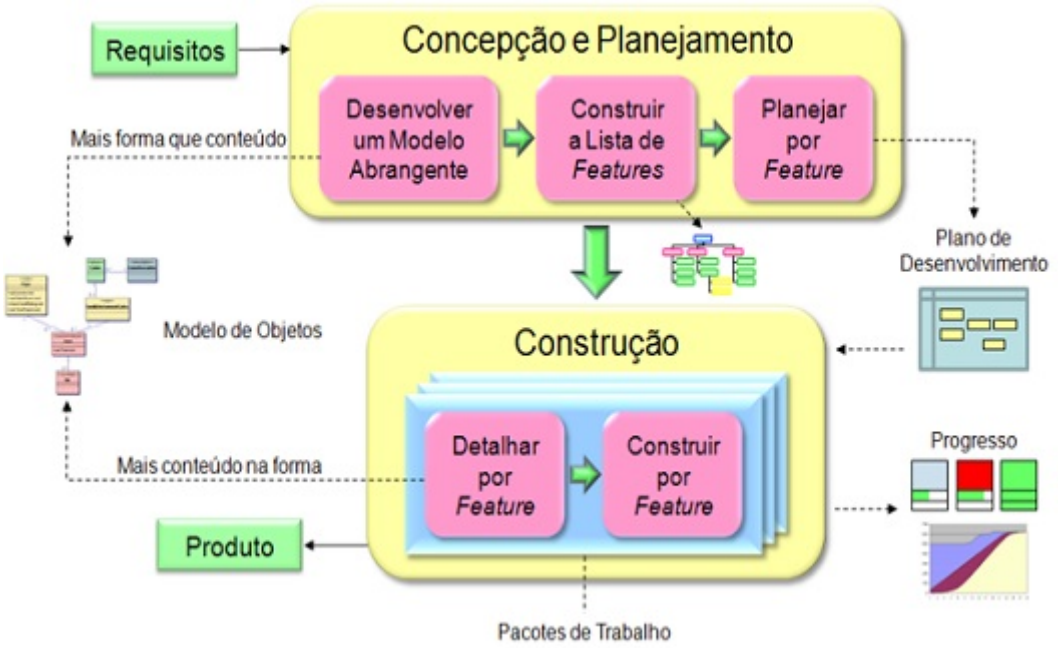

Figura 2.3: Os 5 processos do FDD, extraído de Retamal (2008)

- **Planejar por Funcionalidade:** O gerente do Projeto, o gerente de desenvolvimento e os programadores chefes se reúnem e planejam a ordem que será implementado os recursos, levando em consideração as dependências dos recursos, na complexidade dos recursos e na carga de trabalho da equipe. Neste processo as tarefas principais não são uma sequência estrita e serão consideradas em conjunto e atribuída aos programadores.

- **Detalhar por Funcionalidade:** Após a equipe obter o conjunto de tarefas com prioridades os programadores chefes formam pacotes de trabalho com duração de dois dias à duas semanas. As equipes trabalham sob o comando de um programador chefe.

- **Construir por Funcionalidade:** Os proprietários de cada conjunto de tarefas implementam os itens necessários. O código é testado e inspecionado e, somente quando estiver tudo certo, poderá integrar a versão atual do sistema.

2.5 *Adaptive Software Development* (ASD)

Criado por Jim Highsmith por volta do ano 2000, como uma técnica para construção de software e sistema complexos. A filosofia do ASD se baseia na auto-organização das equipes e colaboração. Este método considera a incerteza e a instabilidade permitindo que a equipe possa aprender durante o desenvolvimento do projeto (Pressman, 2005).

2.5.1 Fases do ASD

Como outras metodologias, o ASD também possui suas fases de desenvolvimento, e Jim Highsmith as divide em um ciclo de vida composto por três fases como na Figura 2.4:

- **Especulação:** É durante a fase de especulação que o projeto e o planejamento do ciclo adaptativo acontecem. Para planejar o ciclo adaptativo é usado informações como declaração de

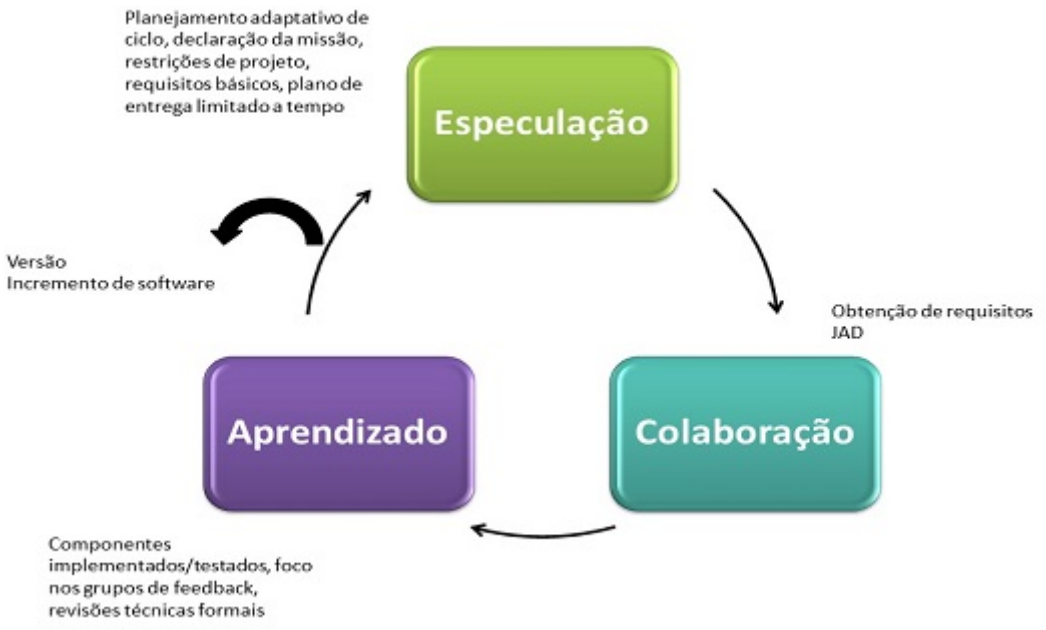

Figura 2.4: Ciclo de vida ASD, extraído de Kolb (2013)

missão feita pelo cliente, as datas a serem cumpridas e requisitos básicos. Essas informações são reunidas para definir o conjunto de ciclos de versões necessárias para o projeto.

- **Colaboração:** Colaboração é algo presente em todos os métodos ágeis, porém não é algo fácil. A colaboração é acima de tudo, confiança. Os membros da equipe devem trabalhar coletivamente para o sucesso do projeto.

- **Aprendizado:** Quando a equipe começa a desenvolver componentes que estão incluídos no ciclo adaptativo, o foco está tanto no progresso para completar um ciclo quanto no aprendizado. Também é fundamental a revisão do projeto em curtos prazos para compreender e validar o que foi planejado.

2.6 *Dynamic Systems Development Method* (DSDM)

O DSDM (*Dynamic Systems Development Method*) foi desenvolvido em meados dos anos 90 a partir da colaboração de vários profissionais de projetos em muitas empresas que buscavam um desenvolvimento de software ágil e de qualidade. As empresas que contribuíram para criação inicialmente conceberam o "Consórcio DSDM"para administrar o compartilhamento, exploração e evolução da propriedade intelectual do DSDM (AgileBusinessConsortiun, 2014).

Os principais objetivos deste método são soluções de qualidade e entregar rápidas, onde devem estar dentro do prazo e do orçamento previsto e é essência o trabalho em equipe. Os criadores do DSDM acreditam que o cliente deve estar envolvido com o projeto, ter o conhecimento e contato com o software, pois com isso os desenvolvedores conseguem um grau de satisfação maior do cliente e um produto de qualidade (Teixeira et al., 2005).

2.6.1 Princípios do DSDM

Oito princípios sustentam a estrutura do DSDM e o torna uma estrutura de projeto ágil, ir contra algum dos princípios enfraquece a filosofia do DSDM podendo causar riscos ao sucesso do projeto. Os oito princípios são (AgileBusinessConsortiun, 2014):

- **Concentrar-se na Necessidade do Negócio:** Cada decisão durante o projeto é essencial, pois refletirá na entrega final do projeto. Para executar este principio a equipe precisa saber as prioridades do negócio, ter compromisso com o projeto, determinar um caso de negócio valido.
- **Entregar no Tempo:** Entregar o produto a tempo é algo fundamental para um projeto, muitas vezes o elemento de sucesso mais importante. Para não falhar neste princípio é necessário ter foco nas prioridades do negócio, cumprir prazos e trabalhar com sprints (semelhante ao Scrum).
- **Colaborar:** Equipes que trabalham em equipe e com compromisso acabam superando grupos que são mal organizados e sem compromisso. O trabalho em equipe bem elaborado permite a maior compreensão do projeto, maior velocidade no desenvolvimento e minimiza erros, pois os colegas de equipe auxiliam os outros quando algo que possa comprometer o final do projeto aconteça.
- **Nunca Comprometa a Qualidade:** Em todas metodologias é fundamental manter o nível de qualidade do produto alto. Para conseguir manter a qualidade do produto é fundamental executar testes desde o início do projeto.
- **Desenvolvimento incremental para chegar a soluções apuradas:** É criado bases sólidas do projeto antes de começar o desenvolvimento, ou seja, primeiro é entendido o problema do negócio e depois planejado como será a solução do mesmo. Ao encontrar uma solução para os problemas, o projeto começa a ser desenvolvido e é entregue de forma incremental, passando um *feedback* constante para o cliente.
- **Desenvolver Iterativamente:** É feito uma demonstração e revisão frequente do projeto com o intuito de mostrar o que foi feito e se está acontecendo alguma problema no projeto. É importante que o Desenvolvimento Iterativo seja usado para estimular a criatividade, a experimentação e a aprendizagem. Também é importante que a equipe do projeto entenda que os detalhes surgem no decorrer do projeto, então a equipe tem que abraçar mudanças.
- **Comunicação Continua e Clara:** A má comunicação no decorrer do projeto é um dos fatores que pode acarretar a falha do projeto. Para evitar esta falha, é incentivado a comunicação presencial entre a equipe e o cliente.
- **Demonstrar Controle:** Em um projeto é essencial demonstrar controle em todo momento. A equipe e as partes interessadas precisam manter o monitoramento e o controle do progresso, assim garantem que o projeto fique alinhado com os planos. É de suma importância que o gerente do projeto e o líder da equipe façam planos e analisem o progresso do projeto.

2.6.2 Processo

O DSDM possui 4 fases principais: **Viabilidade**, **Fundações**, **Desenvolvimento Evolutivo**, **Implantação**. Estas são precedidas pela fase **Pré-Projeto** e seguida pela fase **Pós-Projeto**, totalizando 6 fases (AgileBusinessConsortiun, 2014). Na Figura 2.5 são demonstradas as 6 fases do DSDM descritas a seguir.

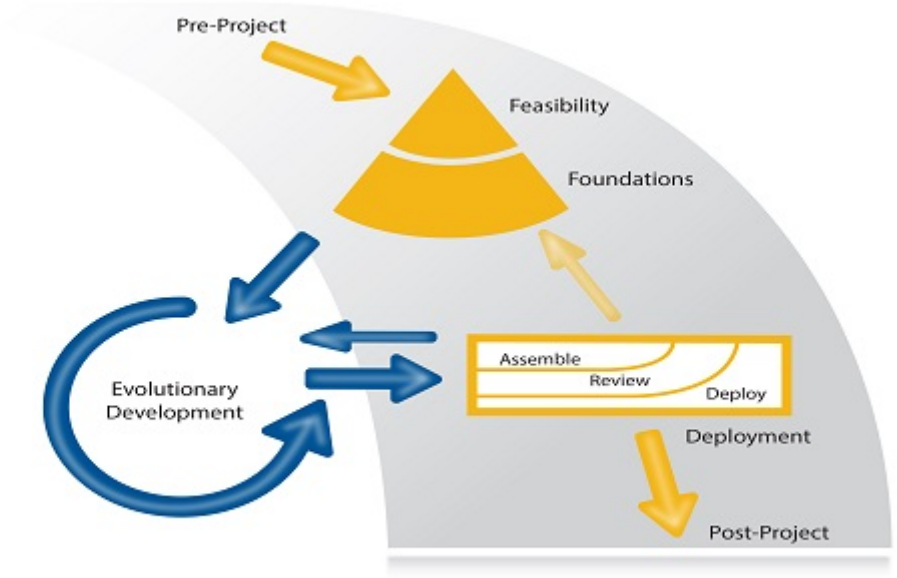

Figura 2.5: Processo DSDM, extraído de AgileBusinessConsortiun (2014)

- **Pré-Projeto:** Nesta fase é assegurado que somente os projetos bem definidos sejam iniciados e que sejam desenvolvidos.
- **Viabilidade:** Essa fase tem como objetivo analisar e estabelecer se o projeto proposto é viável de uma perspectiva técnica e se é economicamente viável da visão do negócio. O esforço gasto com esta fase deve ser apenas para explorar se deve ser dado continuidade ao projeto, ou se tem de ser interrompido, pois é inviável.
- **Fundações:** A investigação do projeto é levada para outro nível. Após a fase da Viabilidade, será realizado uma compreensão não muito detalhada da lógica empresarial do projeto, da possível solução que será criada e como tudo isso estará sendo gerenciado.
- **Desenvolvimento Evolutivo:** A partir das fundamentações que foram estabelecidas na fase anterior, serão desenvolvidas as soluções para as mesmas. A equipe de desenvolvimento tem que aplicar algumas práticas como o desenvolvimento iterativo, priorização das funcionalidades a serem desenvolvidas e o controle das atividades em desenvolvimento, as que não foram iniciadas e as prontas.
- **Implantação:** Após a criação e desenvolvimento das soluções, é feita a implantação. A implantação pode ser feita a partir da solução final, ou um subconjunto da solução final.

- **Pós-Projeto:** O pós-projeto acontece após a última implementação. Realiza-se uma avaliação de desempenho do projeto para verificar o valor do software e ver se os objetivos foram alcançados.

2.6.3 Papéis

O DSDM pode ser dividido de várias formas diferentes, porém sempre vão ter papéis em comum na divisão feita. Um exemplo disso é a divisão que AgileBusinessConsortiun (2014) e Sabbagh (2013) fazem e que possui alguns papéis iguais como:

- **Campeão do projeto:** atua como um gerente executivo, onde o mesmo deve ser organizado, pontual, saber gerenciar prazos e tomar decisões , principalmente quando acontece algum conflito e sua decisão e primordial;
- **Visionário:** é responsável por manter os interesses nos objetivos de alto nível e evitar que os mesmos sejam perdidos quando ocorrer mudanças no projeto e também é responsável em manter o projeto "na linha"obedecendo os prazos;
- **Gerente de projetos:** é encarregado de gerenciar a qualidade, o orçamento e os prazos de entrega do projeto;
- **Líder de equipe:** é um desenvolvedor que tem como objetivo manter a harmonia de sua equipe e motivá-los;
- **Desenvolvedor:** trabalha com o objetivo de encontrar/desenvolver uma solução para os requisitos do negócio proposto;
- **Testador:** realizar os teste no produto final para avaliar se existem algo inadequado e tem que ser modificado;
- **Facilitador:** avaliza como as equipes trabalham e concede o ambiente de trabalho.

Este método, se necessário, pode conter alguns papéis adicionais que serão escolhidos de acordo com o que a equipe precisa e também uma pessoa pode tomar mais de um papel sem prejudicar a equipe (Sabbagh, 2013).

CAPÍTULO 3

Scrum

3.1 Considerações Iniciais

Segundo Schwaber e Sutherland (2014), o Scrum é um *framework* capaz resolver problemas complexos e adaptativos dando um maior valor ao produto final. O Scrum não é uma técnica ou um processo para construir produtos e sim um *framework* no qual pode ser empregado vários processos ou técnicas.

Este método se destaca dos outros, pois se dá uma maior ênfase ao gerenciamento do projeto. O Scrum reúne atividade de monitoramento e *feedback*, também é feito reuniões rápidas e diárias com toda a equipe, com o propósito de identificar e corrigir quaisquer problemas no processo de desenvolvimento (Schwaber, 2004).

Marçal et al. (2007) dizem que o Scrum se baseia em alguns princípios como: equipes pequenas, requisitos que são pouco estáveis ou desconhecidos, iterações curtas.

3.2 Teoria do Scrum

O Scrum é baseado em teorias empíricas de controle de processo, onde a mesma afirma que o conhecimento provém da experiência e de tomada de decisões baseadas no que é conhecido. O controle de processo empírico é apoiado por três pilares (Schwaber e Sutherland, 2014):

- **Transparência:** Aspectos significativos do processo devem estar visíveis aos responsáveis pelos resultados. Esta transparência requer aspectos definidos por um padrão comum para que os observadores compartilharem um mesmo entendimento do que está sendo visto.

- **Inspeção:** Deve ser inspecionado os artefatos do Scrum e o progresso do projeto, para analisar indesejáveis variações, porém as inspeções não devem ser tão frequentes que atrapalhem a execução das tarefas.

- **Adaptação:** Após uma inspeção, caso seja notado alguma variação indesejável em um processo, gerando um produto que não traz os resultados esperados, o mesmo devera ser ajustado

o quanto antes para melhorar os resultados.

A inspeção e adaptação podem ser feitas em algumas das reuniões conhecidas como: ***Sprint Planning Meeting***, ***Daily Scrum***, ***Sprint Review Meeting*** e ***Sprint Retrospective*** que serão discutidas na Seção 3.6 Eventos do Scrum.

3.3 Valores do Scrum

Além dos valores e dos princípios Ágeis, o Scrum também tem seus valores que são (ScrumAlliance, 2012):

- **Foco:** O ideal é trabalhar em apenas um projeto de cada vez, evitando a multitarefa. O Time trabalha em metas de negócios claras e alcançáveis com as quais se comprometem;
- **Coragem:** ter coragem para aceitar as mudanças como parte natural do processo de desenvolvimento do produto;
- **Franqueza:** a franqueza é necessária para que se possa realizar a inspeção e adaptação. Assim, o Time busca melhorar sua forma de trabalhar;
- **Compromisso:** o Time define como seu trabalho será realizado, monitora progressos e realiza correções quando achar necessário;
- **Respeito:** os membros do Time trabalham juntos, compartilhando responsabilidades, e assim ajudando uns aos outros em seu trabalho para cumprirem as metas.

3.4 Papéis e Responsabilidades

No Scrum, é feito uma divisão de papéis e responsabilidades, apesar desta divisão ser pequena e simples já é o suficiente para que a equipe consiga entregar um produto de alto valor. Esta divisão de papéis e responsabilidade no Scrum é feita em três partes, explicadas por (Schwaber, 2004, Sabbagh, 2013):

- **Product Owner:**

    1. **Gerencia o Produto:** O Product Owner tem contato constante com as partes interessadas ao longo do projeto para fazer o levantamento dos requisitos e suas prioridades. Ele determina quais destes requisitos farão parte do **Product Backlog**. Ele também pode retirar itens do **Product Backlog** que já não são mais necessários para o projeto. Os clientes e o **Time de Desenvolvimento** explicam para o Product Owner o que precisa conter no software e assim o Product Owner escolhe e monta os requisitos do sistema.
    2. **Gerencia as partes interessadas no projeto:** O Product Owner faz o papel de mediador, transmitindo as ideias dos clientes e das demais partes interessadas no produto em desenvolvimento. Assim, O Product Owner verifica quem são os clientes e as pessoas

relevantes para o projeto, para poder assim identificar e entender quais as reais necessidades do negócio, administrar as expectativas dos clientes, para o mesmo saber o que irá ser demonstrado e o que será mostrado em seguida no projeto.

3. **Aceita ou rejeita a entrega do Time de Desenvolvimento:** O Product Owner tem o poder de aceitar ou recusar a entrega do **Time**. Para isso é verificado se o produto cumpre com todos requisitos do Sprint, caso a meta tenha sido cumprida, o Product Owner pode aceitar o produto, se não ele tem o poder de recusar, até que tudo esteja certo.

- **Team/Time:**

1. **Planeja seu Trabalho:** O Time junto com o Product Owner planeja o que irá ser feito na Sprint que está iniciando. Isto é, o Time junto com Product Owner decidem quais itens farão parte da Sprint e determinam qual a sua meta. O Time também irá planejar como os itens escolhidos serão feitos, dividindo os itens em tarefas e estimado o tempo de trabalho em cada uma.
2. **Realiza o desenvolvimento do Produto:** Após a divisão das tarefas, o Time começa a desenvolver as tarefas nas Sprints para que essas tarefas se tornem funcionalidades do produto. O objetivo do Time é atingir a Meta do Sprint e com resultados de qualidade.
3. **Entrega valor com frequência:** Em cada Sprint o Time desenvolve as funcionalidade definidas pelo Product Owner. Conforme o conteúdo da Sprint fica pronto, o produto adquire mais valor e os clientes já podem usar o produto com as funcionalidades já prontas.

- **Scrum Master:**

1. **Facilita o trabalho do Time de Scrum:** O Scrum Master não é apenas um facilitador do Time, mas também do Product Owner. Ele estimula a boa relação de trabalho, para que os membros da equipe fiquem cada vez mais próximos e compartilhem mais coisas um com os outros a respeito do projeto.
2. **Remove impedimentos:** O Scrum Master é encarregado de remover os obstáculos que impedem a meta da Sprint, onde ele mesmo poderá resolver ou irá instruir pessoas e os recursos necessário para resolver tal obstáculo. Quando algum impedimento for identificado pelo Time, o mesmo deve notificar o Scrum Master, para ele tomar tais medidas.
3. **Garante o uso do Scrum:** O Scrum Master também é encarregado de ensinar e garantir que os valores, as regras e a prática do Scrum sejam compreendidos e seguidos por todos. A todo momento o Scrum Master verifica se não houve desvios no uso do Scrum no projeto, caso seja identificado algo, o mesmo vai tomar medidas que julga necessário para corrigir este desvio.

3.5 Artefatos do Scrum

Os artefatos são representações dos valores e implementações do projeto. O intuito do uso dos mesmos é ser o mais transparente possível durante o projeto. É ideal que todos os participantes entendam os artefatos da mesma forma (Schwaber e Sutherland, 2014).

- **Sprint Backlog:** São tarefas que serão desenvolvidas na Sprint, e conforme a Sprint evolui, pode surgir novas tarefas a serem implementadas. As tarefas são escolhidas pelo **Time** com a ajuda do **Product Owner** de acordo com a prioridade de cada uma, tendo importância tarefas com maior prioridade;

- **Product Backlog:** O **Product Owner** junto com a equipe do Scrum são responsáveis por determinar os requisitos do produto ou sistema que será desenvolvido. É feito um empilhamento das funcionalidades do programa/produto, de acordo com a Figura 3.1. A sequência deste empilhamento é definida a partir das prioridades de cada funcionalidade. Quanto maior a prioridade, mais importante, ou seja, as funcionalidades que são definidas mais importantes, serão desenvolvidas primeiro. O Product Backlog é volátil, pois pode ser adicionado novas funcionalidades ou até mesmo retirar algumas que já não são mais importantes;

Figura 3.1: Pilha de tarefas de um Product Backlog, extraído de Vieira (2014).

- ***Burndown*:** É possível manter o controle de desenvolvimento das tarefas e até mesmo das Sprints, esse controle é feito por meio de gráficos, conhecidos como *Burndown*. No *Burndown* é ilustrado as tarefas que foram executadas, e as serem executadas e o tempo gasto em cada uma delas (Wazlawick, 2013). Na Figura 3.2 é possível ver um exemplo deste tipo de gráfico. A linha vermelha representa o progresso ideal do projeto, e a linha azul o andamento do projeto, neste caso, a linha azul está a cima da vermelha, então o projeto não está de acordo com o esperado, pois está atrasado e caso esteja abaixo desta linha o projeto estará adiantado. Essa medida foi feito em Sprints X Horas, ou seja a data da entrega de cada Sprint e as horas gastas para completa-las.

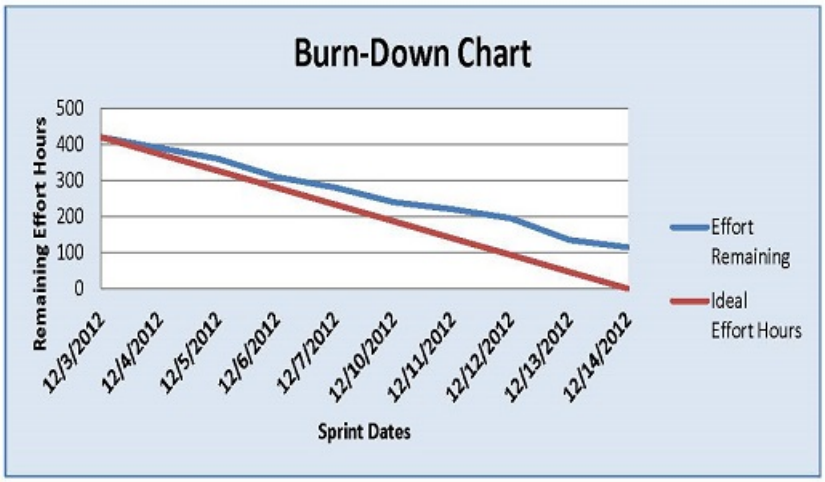

Figura 3.2: Exemplo de um gráfico BurnDown, extraído de Mittal (2013)

- **Quadro de Tarefas:** É o local que pode ser uma parede ou quadro em que vai conter as tarefas do projeto semelhante com o da Figura 3.3. O quadro de tarefas pode ser dividido em quatro colunas: **Estórias** que é a identificação de qual Sprint estão trabalhando; **Fazer**, que é a coluna onde irá ficar as tarefas que ainda não foram começadas; **Fazendo**, coluna onde ficará as tarefas que estão em execução; **Feito**, são as tarefas que já foram implementadas.

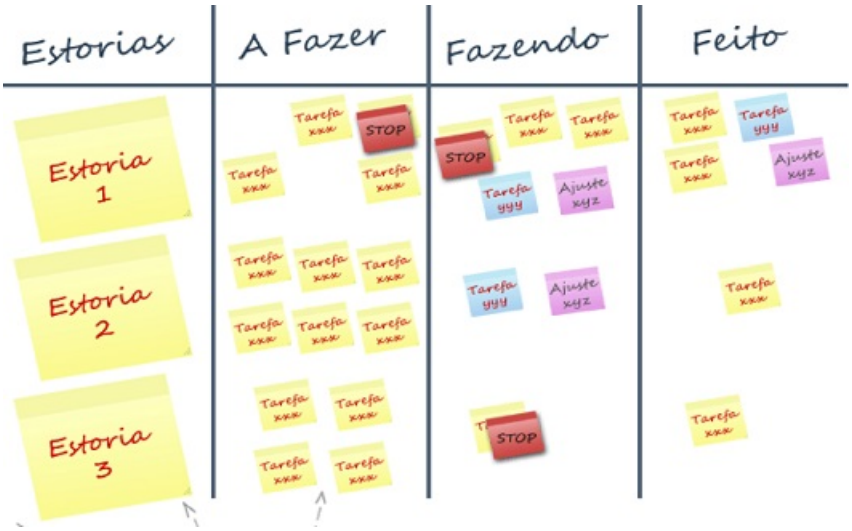

Figura 3.3: Exemplo de um quadro de tarefas, extraído de Cruz (2016)

3.6 Eventos do Scrum

Conforme Schwaber (2004) e Sabbagh (2013) dizem que o Scrum pode ser dividido em eventos. Os eventos são os ciclos de processos que acontecem durante o projeto, como:

- **Sprint:** É a divisão do desenvolvimento do projeto em intervalos de no máximo 30 dias, cada divisão será uma etapa a ser cumprida, como mostra a Figura 3.4. Em cada etapa cumprida deverá ser apresentado algo de valor tangível ao cliente, para que o mesmo já comece a trabalhar com o produto e possa ter um feedback de como está ficando.

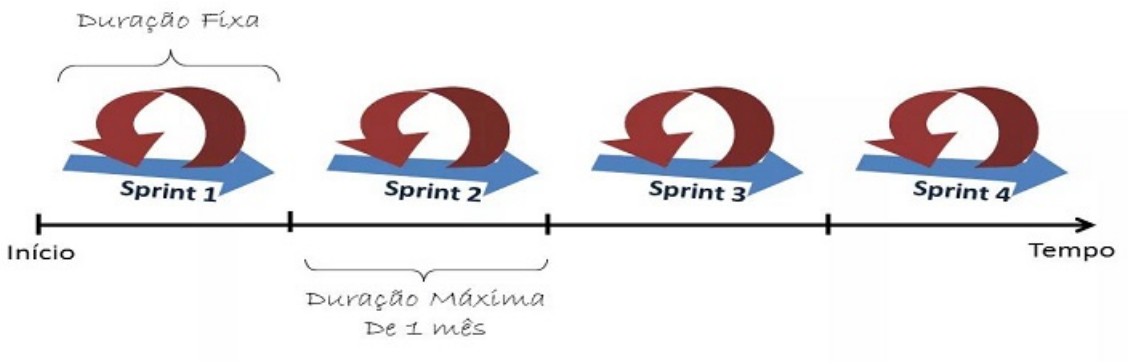

Figura 3.4: Visão geral das Sprints, extraído de Vieira (2014).

- **Sprint Planning Meeting:** São reuniões feitas antes de iniciar uma Sprint, para definir quais subconjuntos de elementos do **Product Backlog** são mais importantes para se construir na Sprint. Nestas reuniões participam o **Product Owner**, **Time** e o **Scrum Master** para chegarem a conclusão de qual o objetivo da Sprint;

- **Daily Scrum:** Reunião de no máximo 15 minutos feita diariamente com a equipe, onde cada membro da equipe responde três perguntas para o **Scrum Master**:

  1. O que você fez desde a última reunião do Daily Scrum?
  2. O que você planeja fazer entre agora e a próxima reunião do Daily Scrum?
  3. Que impedimentos estão no caminho de você cumprir seus compromissos com este Sprint e este projeto?

  O objetivo da reunião é sincronizar o trabalho de todos os membros e analisar o andamento do projeto.

- **Sprint Review Meeting:** São reuniões que acontecem no final de cada Sprint com a presença do **Scrum Master**, do **Time** e do **Product Owner** com o objetivo de mostrar a evolução do projeto e o que já está pronto. O Product Owner analisa se o Time cumpriu a meta da Sprint. Os itens que não ficaram prontos normalmente são os de menos valor.

- **Sprint Retrospective:** É realizado no fim de cada Sprint com o objetivo de analisar a forma como foi trabalhado, se funcionou bem e o que pode ser melhorado. Este evento ocorre depois da Sprint Reviw Meeting e antes da próxima Sprint, para que as melhorias possam ser inseridas na Sprint seguinte.

- **Release:** É a entrega de parcelas do produto que foi desenvolvido pelo Time e já estão prontas. As entregas são feitas após alguns Sprints, pois o produto tem que conter um valor considerável para ser manipulado. Os objetivos do Release são: ***feedback* constante**, para mostrar ao Product Owner e aos interessados se o produto está de acordo com o esperado e se existe alguma mudança a ser feita no Product Backlog; **prover retorno ao investimento dos clientes**, para

mostrar aos clientes do projeto o valor que o produto vem ganhando e o retorno de seus investimentos; e **dar um senso de progresso do projeto**, para o cliente ter a noção do que já foi feito durante o projeto e o que ainda está por vir.

- **Release Planning:** São encontros entre o Product Owner e o Time para definir o que será entregue na Release e o que o Time pretende entregar.

3.7 Scrum na Prática

A Figura 3.5 mostra uma visão geral de um projeto com a metodologia Scrum de acordo com algumas de suas práticas mostradas abaixo.

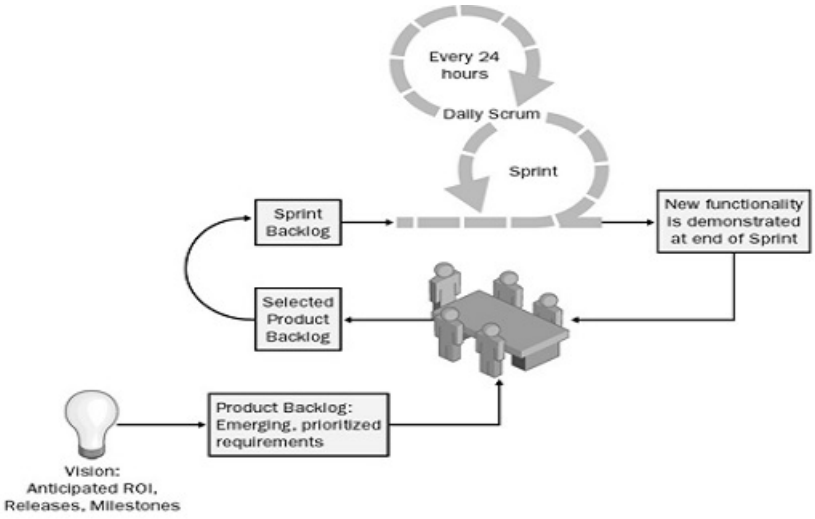

Figura 3.5: Visão geral do Scrum, extraído de Schwaber (2004)

Para iniciar um projeto é criado uma lista contendo todos os requisitos conhecido, esta etapa é conhecida como **Product Backlog.** Cada Sprint é iniciada com uma reunião de planejamento (**Sprint Planning Meeting**), onde o Product Owner e o Time participam para decidirem em conjunto o que terá de ser implementado (Selected Product Backlog). Esta reunião é dividida em duas partes, onde, na primeira parte o Product Owner apresenta os requisitos mais importantes para ele, e os mesmos são priorizados para serem implementados primeiro. Na segunda parte, o Time define o que vão faze, definindo o **Sprint Backlog**, que é apenas uma parte dos requisitos listados no **Product Backlog.**

Na execução das Sprint, é realizado reuniões informais diárias de 15 minutos (**Daily Scrum**) para poder acompanha o andamento do projeto e poder agendar outras reuniões que sejam necessárias. No fim de cada Sprint é feito uma reunião para o Time apresentar seu progresso conhecida como **Sprint Review Meeting**.

3.8 Scrum Além do Desenvolvimento de Software

O Scrum foi criado com o intuito de auxiliar e melhorar a criação de software. Porém, visto que é eficaz, ele foi inserido em outras áreas. Hoje pode ser encontrado em indústrias para fabricação de roupas, brinquedos, eletrônicos, entre outros (Yang et al., 2010).

Nos trabalhos "*Agile industrial design management based on Scrum*" de Yang et al. (2010) e "Scrum Integration in Stage-gate Models for Collaborative Product Development - A Case Study of Three Industrial Manufacturers" de Sommer et al. (2013) é mostrado a inserção do Scrum em alguns meios como os citados acima.

Como exemplo, Yang et al. (2010) propõe o uso do Scrum com a intenção de melhorar as atividade de design industrial para torná-las melhores e mais rápidas. No método que era utilizado antes do Scrum se perdia muito tempo com o esboço do projeto e a sua análise era desprezada por conta de prazos a serem cumpridos e da atenção excessiva com o esboço.

Por conta destes motivos, as empresas começaram a buscar alguma forma de conseguir agilidade para desenvolver um projeto e com isso, foi encontrado o Scrum que, apesar das diferenças, apresenta características em comum no desenvolvimento de software com o design industrial (Yang et al., 2010).

O design industrial é algo difícil de ser feito, devido mudanças na oferta e demanda de cada produto. Para suprir esta necessidade, a equipe trabalha de modo dinâmico e com métodos flexíveis. Estes métodos flexíveis são ferramentas muito eficientes para se obter um rápido design. Em um projeto é escolhido alguns métodos dentro dos vários que existem. Esta escolha é feita de acordo com o tipo de produto a ser produzido. Visto que o Scrum é um método que satisfaz essas exigências, foi desenvolvido um modelo de design industrial baseado no mesmo, como é demonstrado na Figura 3.6.

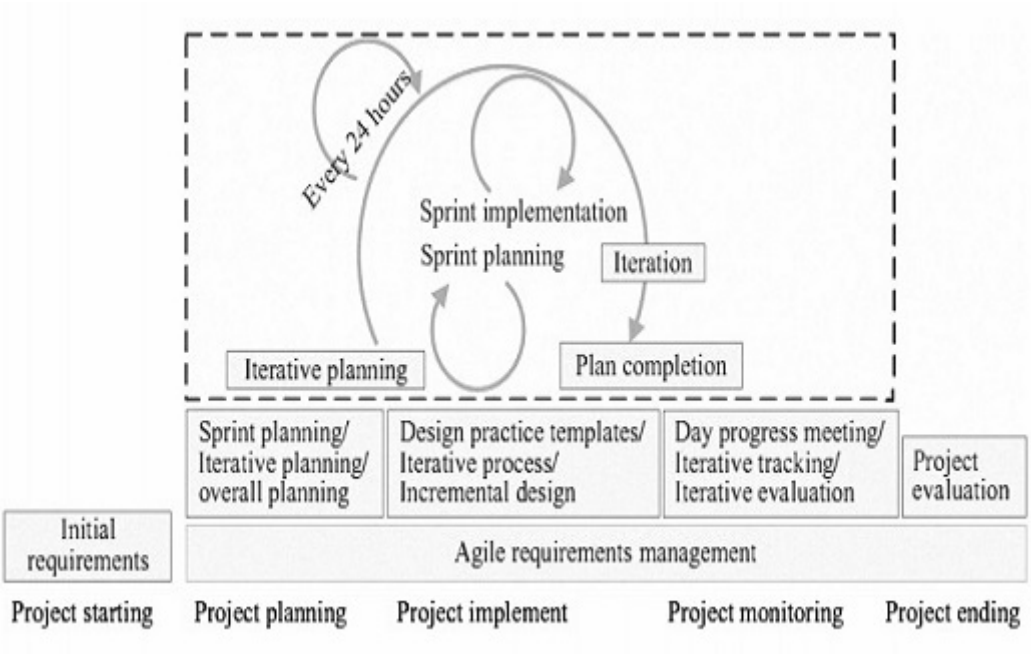

Figura 3.6: Processo Scrum aplicado ao design, extraído de Yang et al. (2010)

Os principais métodos para o desenvolvimento do design industrial de forma ágil foram:

- **Gestão de Demanda:** Uma das coisas mais importantes é a Gestão de Demanda, pois deve estar presente em todo processo de design industrial. Esta gestão é feita a partir de cartões de índice, cartões de história, descrição dos recursos e a prioridade dos recursos, que é determinada pelo cliente.

- **Processo Iterativo:** Este método é composto por planejamento, acompanhamento, avaliação

dentre outros. Alguns Processos Iterativos faz um acompanhamento e avaliação conforme o avanço das Sprints.

- **Modelos de Prática:** São modelos de testes a serem executados para análise e correção de erros. Existem vários modelos de teste como tentativa e erro, questionário de perguntas, experiencia do usuário.

Segundo Yang et al. (2010), o Scrum em conjunto com a gestão é a base para o design industrial, pois mostrou resultados em um curto período de tempo e atendeu os requisitos impostos pelos clientes.

No trabalho de Sommer et al. (2013), o Scrum foi implantando em três grandes fabricantes industriais dinamarqueses de diferentes áreas de atuação, como demonstra a Figura 3.7.

Visão geral do caso

Empresa	Tipo	Processo de desenvolvimento de produto
A	Grande fabricante de brinquedos de plástico	Desenvolvimento de soluções digitais
B	Grande fabricante de janelas	Desenvolvimento de janela
C	Grande fabricante eletrônicos	Scrum em modelo de processo linear para desenvolvimento de amplificador

Figura 3.7: Demonstração das Industrias que foi implementado Scrum, extraído de Sommer et al. (2013)

As fábricas possuem uma concorrência global, onde a diferença é essencial para se manter no mercado. Com o passar dos anos essa concorrência aumenta e a busca por um diferencial em que se possa abaixar os custos e ampliar a produção também aumenta. Com essa ideia de busca por um diferencial, a metodologia Scrum foi inserida nesse meio, por ser um processo ágil voltado principalmente ao cliente.

Outros modelos já foram testados, como o em espiral e o em cascata, entretanto não apresentaram bons resultados devido à dificuldade de execução na prática. Assim, o Scrum foi implementado com outra ferramenta, conhecida como *Stage-Gate* (Sommer et al., 2013).

O *Stage-Gate* é uma ferramenta para gerenciar os gastos de recursos desnecessários em projetos. O projeto é dividido em estágios (*Stages*) e contém pontos de decisão (*Gates*) no qual será repensado se deve ser dado continuidade ao projeto (elogroup, 2015).

Os resultados foram obtidos por meio de entrevistas em três empresas. Porém, nas empresas A e C a entrevista foi realizada com dois gerentes de cada empresa. Já no empresa B, aconteceram duas entrevistas com as oito equipes de desenvolvimento, uma antes e outra depois da implantação.

Na empresa A foi aplicado o Scrum junto ao método *Stage-Gate*, como mostra a Figura 3.8. O processo de desenvolvimento foi dividido em três etapas (Sommer et al., 2013):

1. Integração do *Stage-Gate* ao projeto.
2. Conjunto de ações que devem ser realizadas para cumprir os requisitos.
3. O processo Scrum, cuja finalidade é concluir o trabalho e torná-lo operacional.

Os membros que compõe a equipe da empresa A são: o Scrum Master, o Gerente de Projetos, e ao menos um Product Owner, caso o projeto seja de grande porte é ideal ter mais de um Product Owner. O foco para o Product Owner e o Scrum Master é no desenvolvimento do produto, enquanto o do Gerente de Projetos é de coordenar e interligar o resto da empresa. O ponto que mais se destacou após a inserção do Scrum no projeto foi a aproximação do cliente com a equipe de desenvolvimento. Essa aproximação possibilitou um produto final de maior qualidade e mais preciso ao pedido do cliente.

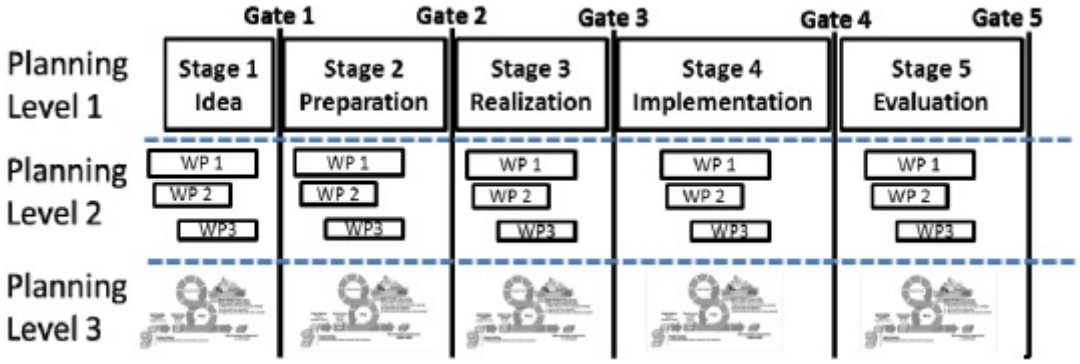

Figura 3.8: Modelo de Desenvolvimento de Produto Caso A, extraído de Sommer et al. (2013)

A empresa B também usa um modelo de *Stage-Gate* e o método Scrum no projeto. O planejamento do projeto é dividido em três níveis, onde os níveis são divididos por etapas (*Gates*) que são seguidos e cumpridos rigorosamente, o exemplo dessa divisão pode ser visto na Figura 3.9.

O segundo nível é um nível departamental, onde possui quatro áreas que estão ligadas diretamente ao Scrum e garantem um desenvolvimento integrado de produtos que são:

1. **Gerenciamento de Projeto:** É representado por coordenadores que possuem conhecimento de tudo o que ocorre no projeto.

2. **Mercado:** O papel do Mercado vai estar em contato com os cliente, dando um *feedback* do andamento do projeto, fazendo reuniões, levantamento de dados para analisar se o software está respondendo as necessidades e até mesmo montar o Product Backlog.

3. **Produto:** Quem está presente nessa área é o Time de Desenvolvimento que é responsável pela produzir e entregar o produto final.

4. **Suprimento:** É representado por pessoas que tem como objetivo não deixar recursos e materiais faltarem para a conclusão do projeto.

Na empresa B não acontece o Daily Scrum devido ao fato de a empresa estar localizada em locais diferentes da Dinamarca, impossibilitando essa reunião. Para suprir esta necessidade, os funcionários realizam uma reunião ao menos um dia na semana que toma quase todo o dia. Ao inserir alguém na área do mercado notou-se um aumento na satisfação do cliente, por o mesmo sempre estar a parte do que está acontecendo, como no caso do cliente, foi notado também um efeito positivo na equipe de desenvolvimento, pois estão mais comunicativos organizados (Sommer et al., 2013).

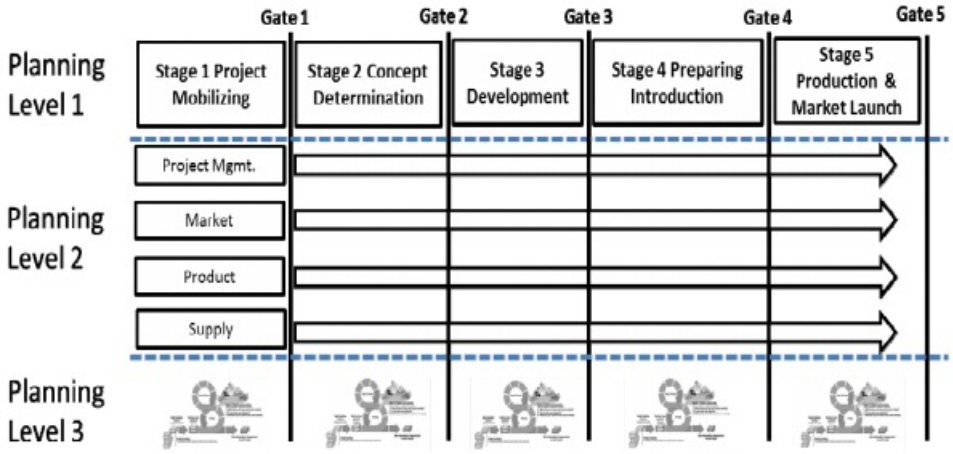

Figura 3.9: Modelo de Desenvolvimento de Produto Caso B, extraído de Sommer et al. (2013)

Como nas outras empresas, na empresa C também foi usado o *Stage-Gate* e o método Scrum. O Scrum neste caso tem como objetivo reorganização da equipe, para melhorar a organização e dedicação nas área de negócio. Outro fator que muda é que o Gerente de Projetos foi alterado pelo Product Owner e pelo Scrum Master. O Gerente de Negócios sempre procurar deixar o cliente sempre informados do que ocorre do projeto (Sommer et al., 2013).

Apesar de alguns desvios no uso do Scrum durante o projeto, tudo ocorreu como o planejado e até melhor do que se esperava. O número de funcionários diminuiu, mas a qualidade do produto aumentou e houve menos reclamações de clientes.

Nota-se que em ambos casos o resultado esperado foi alcançado, ou até melhor superado e os funcionários estão mais motivados a participarem da equipe. Possivelmente, devido a flexibilidade permitida no Scrum em conjunto com outras táticas, foi notado essas melhoras, tanto da parte da equipe quanto do cliente.

CAPÍTULO 4

Ferramenta de Suporte ao Gerenciamento de Manutenção Industrial

4.1 Considerações Iniciais

Neste capítulo é proposto uma ferramenta, denominada MIND'S, que é utilizada para realizar o suporte ao gerenciamento de manutenções industriais tendo como base o *framework* ilustrado na Figura 4.15, e que está presente no fim deste capítulo. Esse *framework* representa procedimentos de manutenções que terão de ser seguidos, onde alguns desses procedimentos estão contidos em uma interface que ajudará a manter uma documentação sobre o equipamento.

4.2 Descrição Geral do Sistema

A ferramenta utilizada para implementação da MIND'S foi o Delphi, uma IDE (*Integrated Development Environment*) que auxilia o uso da técnica RAD (*Rapid Application Development*), onde essas técnicas facilitam a criação de softwares de uma forma rápida e com um bom resultado final.

Para o armazenamento de todos dados obtidos através do uso da ferramenta será criado um banco de dados através da ferramenta IBExpert e contará com um servidor, e para isso, será usado o Firebird, onde o mesmo cria um servidor local.

Na Figura 4.1 é exibida a interface inicial do sistema, contendo uma barra de menu no canto superior, onde a mesma levará o usuário para as outras interfaces. Também está presente na interface um gráfico que ilustra a quantidade de ordens de serviço fechada pelos funcionários, um relatório com o total de ordens abertas e fechadas e a próxima manutenção preventiva a ser executada.

O primeiro item da barra de menu é a **Área Administrativa** onde possui a seguinte função:

- **Atualizar:** A função de atualizar serve para atualizar o sistema como um todo. Caso o sistema tenha tido alguma inserção, remoção ou edição de dados esta função poderá ser utilizada para atualizar os mesmos.

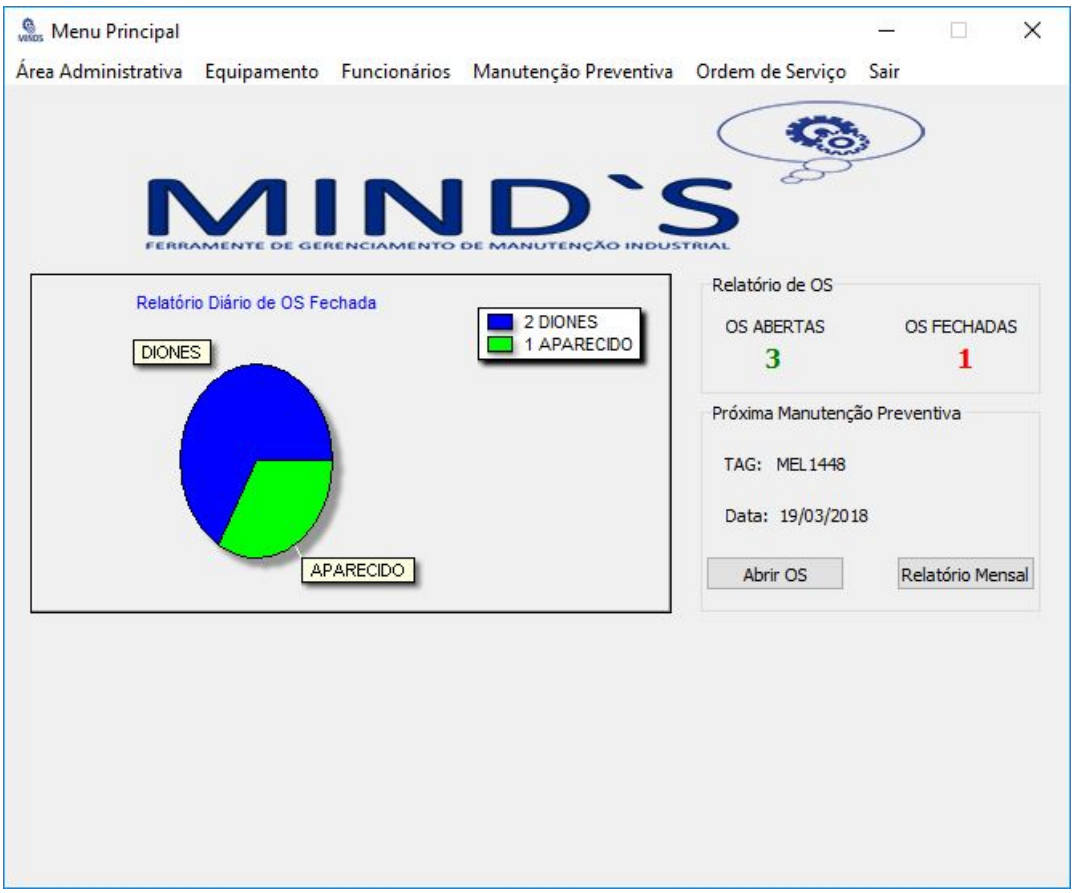

Figura 4.1: Menu Principal

O segundo item da barra de menu é o de **Equipamentos**, onde o usuário poderá **Cadastrar**, **Editar** ou **Pesquisar** algum equipamento. Os itens que representam essas funções são:

- **Cadastrar Equipamento:** Essa função permite que o usuário cadastre novos equipamentos no sistema. Para realizar o cadastro é necessário adicionar as características do equipamento que o sistema requer, como demostra a Figura 4.2.

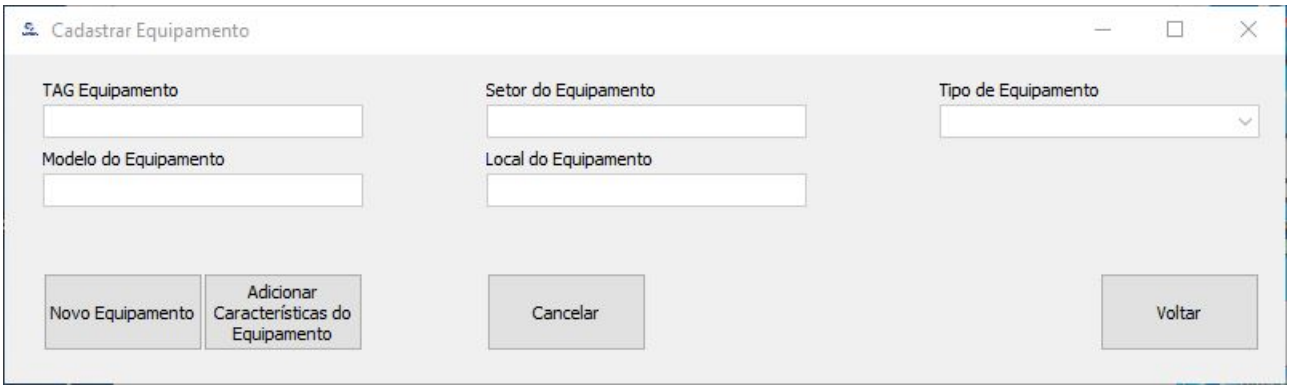

Figura 4.2: Interface de cadastro de equipamentos

Ao preencher os campos com as especificações do equipamento o usuário terá que **Adicionar Características do Equipamento** de acordo com o seu tipo, caso o equipamento for elétrico será aberto a interface da Figura 4.3, caso for mecânico o da Figura 4.4.

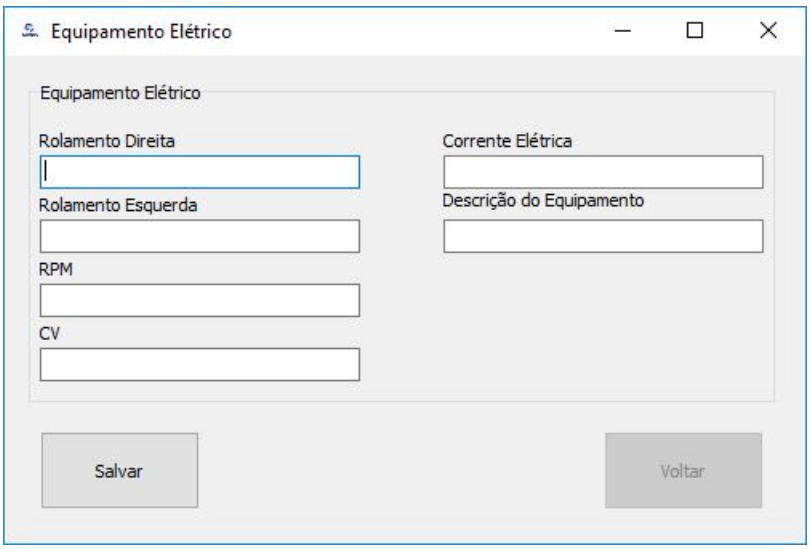

Figura 4.3: Interface de cadastro das características do equipamento elétrico

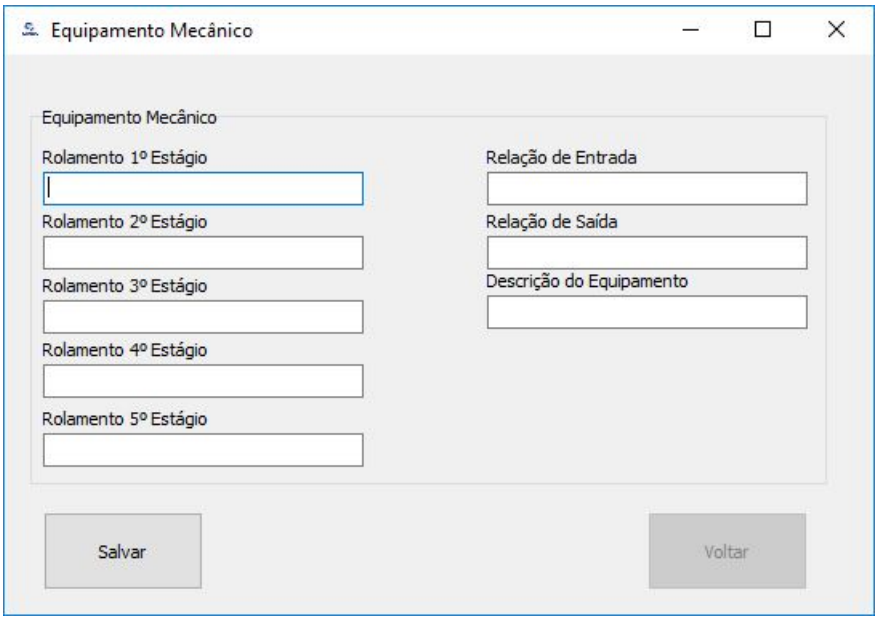

Figura 4.4: Interface de cadastro das características do equipamento mecânico

- **Editar Equipamento:** Essa função é ilustrada na Figura 4.5 e permite que o usuário edite os dados de algum equipamento que já foi inserido no sistema, ou até mesmo o exclua caso necessário. Para facilitar a edição existe um campo de pesquisa, onde a mesma pode ser feita pela TAG do equipamento ou pela descrição, após encontrar o equipamento desejado o usuário terá que clicar em "**Editar**" para editar o que deseja.

- **Pesquisar Equipamento:** Nesta função o usuário poderá escolher se pretende pesquisar o equipamento pela TAG ou pela descrição, após a pesquisa será mostrado todas as características do equipamento, como na Figura 4.6.

O terceiro item da barra de menu é o de **Funcionários**, onde o usuário poderá **Cadastrar** novos funcionários, **Editar** ou **Pesquisar** algum funcionário já existente. Os itens que representam essas funções são:

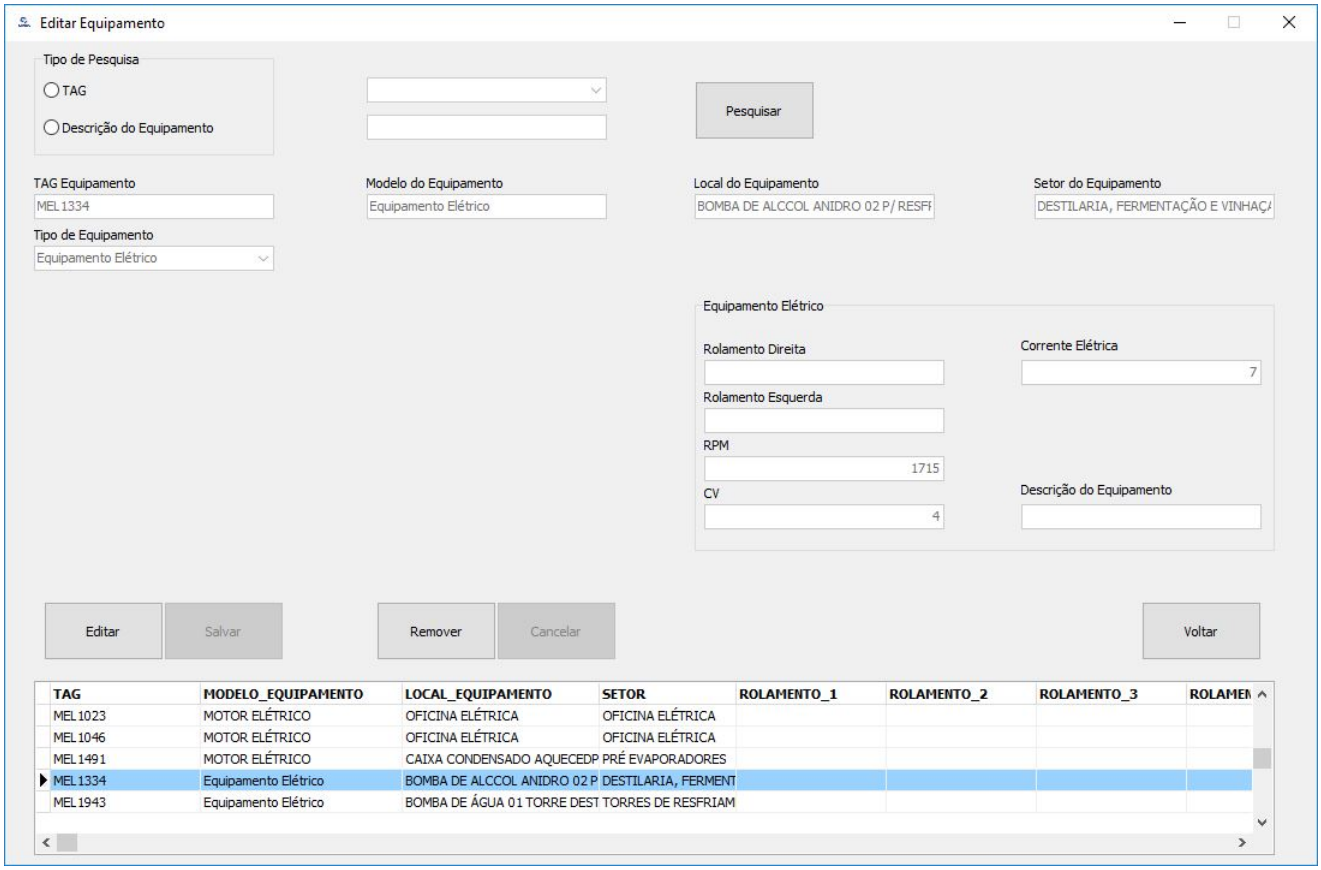

Figura 4.5: Interface de Edição de equipamentos

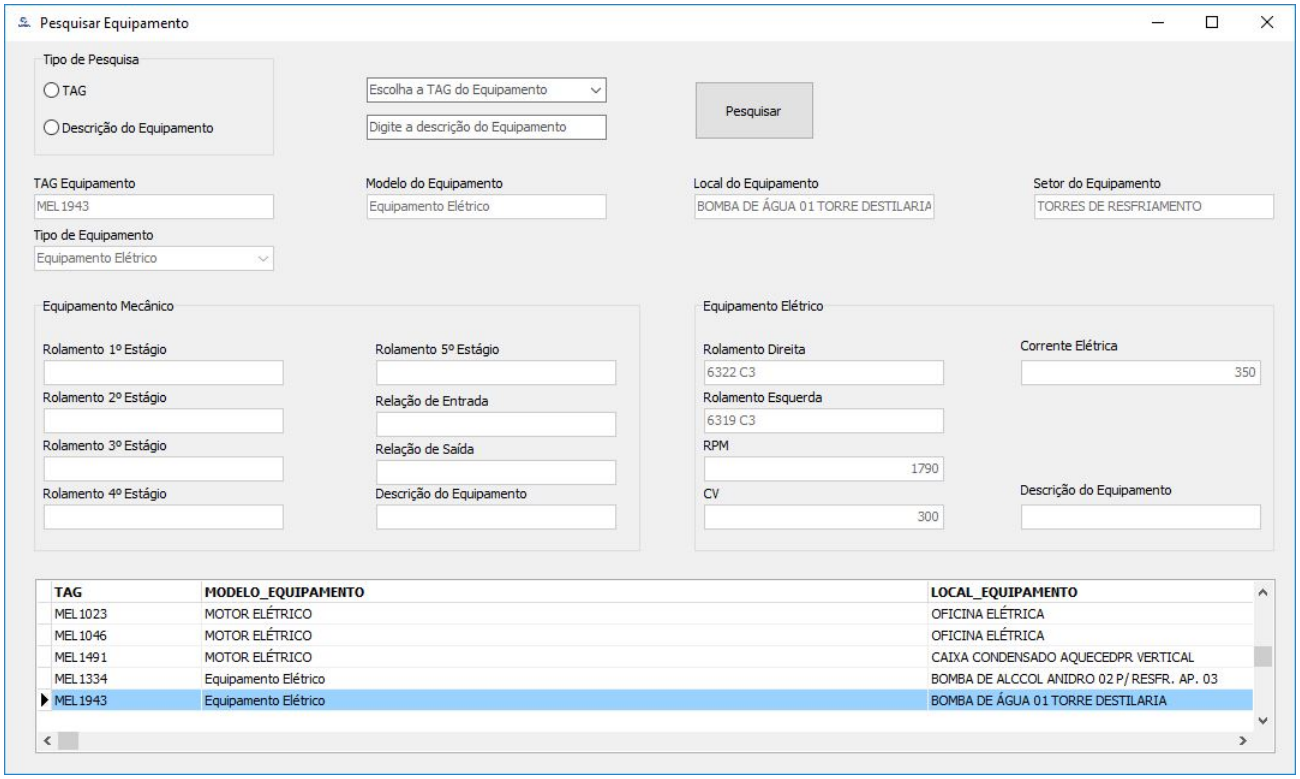

Figura 4.6: Interface de pesquisa de equipamentos

- **Cadastrar Funcionário:** Esta função do sistema permite o usuário adicionar novos funcionários preenchendo o formulário presente na Figura 4.7.

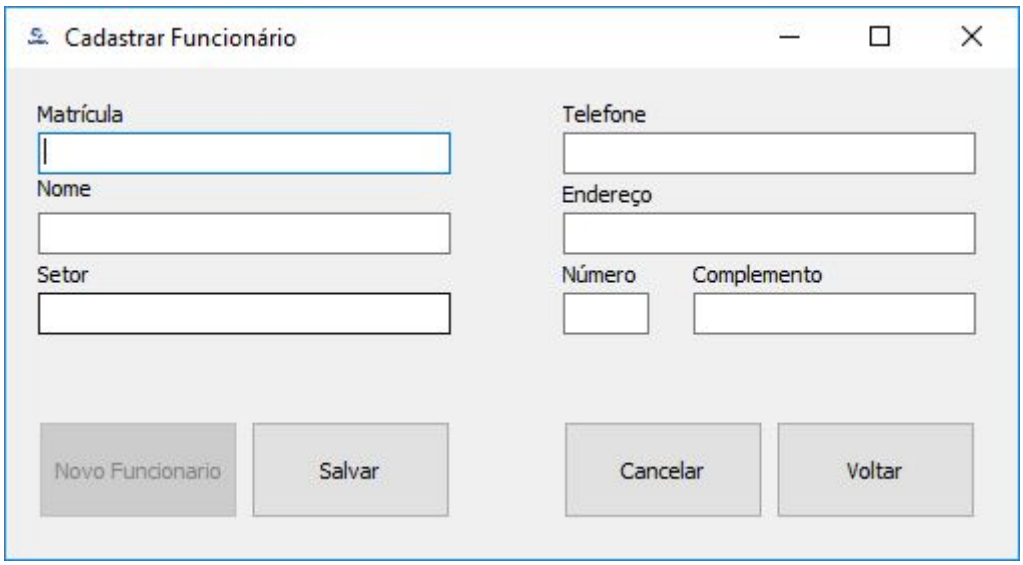

Figura 4.7: Interface de cadastro de funcionário

- **Editar Funcionário:** Na interface de edição de funcionário caso necessário é possível pesquisar o funcionário que deseja editar ou excluir, a pesquisa pode ser efetuada a partir do número da matrícula ou pelo nome do funcionário, após encontrar quem deseja editar o usuário pode acionar o modo de edição.

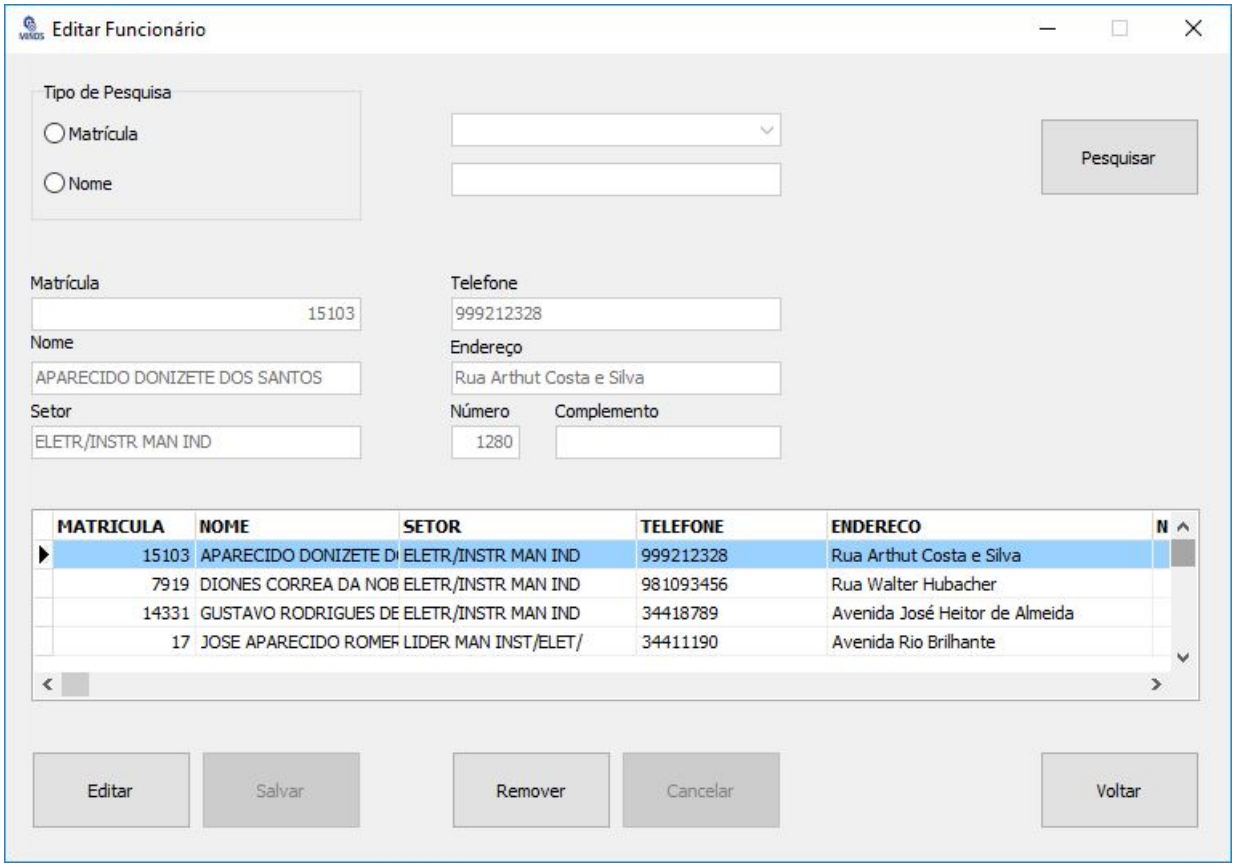

Figura 4.8: Interface de edição de funcionário

- **Pesquisar Funcionário:** Na interface de pesquisar funcionário é possível pesquisar um funci-

onário por sua matrícula ou pelo seu nome.

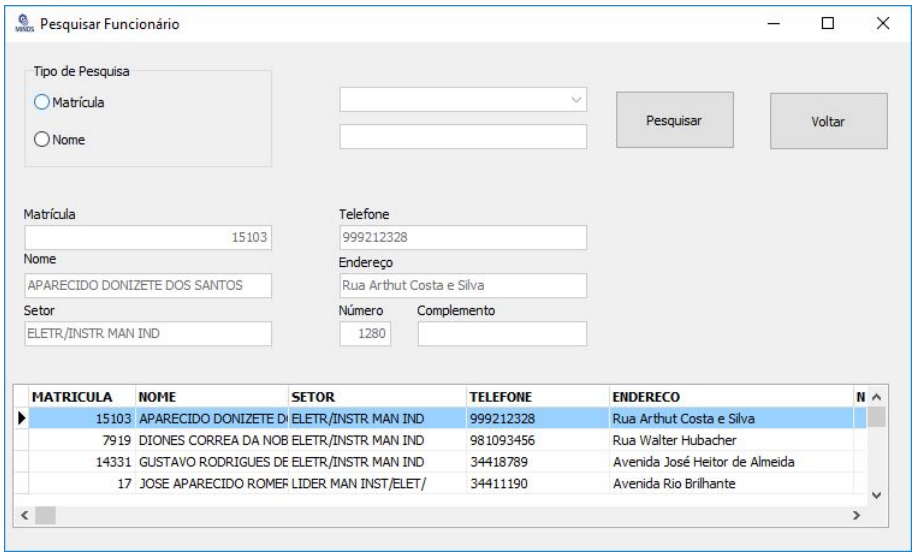

Figura 4.9: Interface de pesquisa de funcionário

4.3 Sprint Planning

Caso a manutenção não seja corretiva, o *framework* indica que deverá ser feito o planejamento da mesma. O Sprint Planning representa esse planejamento da manutenção, representado na ferramenta por uma funcionalidade especifica em que onde é possível realizar o agendamento de manutenções preventivas.

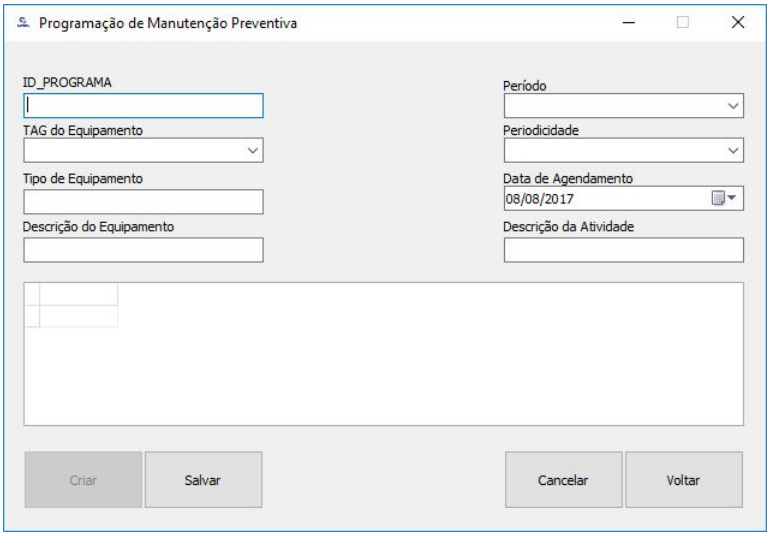

Figura 4.10: Interface de programar manutenções preventivas

O quarto item da barra de menu é a Manutenção Preventiva, que é onde o usuário poderá **agendar** e **pesquisar** manutenções preventivas. Os itens que representam essas funções são:

- **Programação de Manutenção Preventiva:** Seguindo a ideia do mesmo, caso a equipe tenha analisado o equipamento, mas não realizou nenhum tipo de manutenção, será feito um agendamento de manutenções preventivas a partir da interface apresenta na Figura 4.10, evitando que

o equipamento pare de forma inesperada no futuro. Para realizar o agendamento é preciso escolher a TAG de um equipamento, em seguida é preciso selecionar o período do agendamento da manutenção, onde esse período pode ir de 1 mês até 3 anos. Após escolher o período o usuário deverá escolher a periodicidade das manutenções, que pode ser desde diária até anual. Por fim, o usuário precisará designar a data de início das manutenções e descrever a atividade que será realizada.

- **Pesquisar Manutenção Preventiva:** Nesta funcionalidade, o usuário poderá escolher o equipamento que deseja pesquisar os agendamentos que foram feitos a partir da TAG ou descrição do equipamento, como na Figura 4.11.

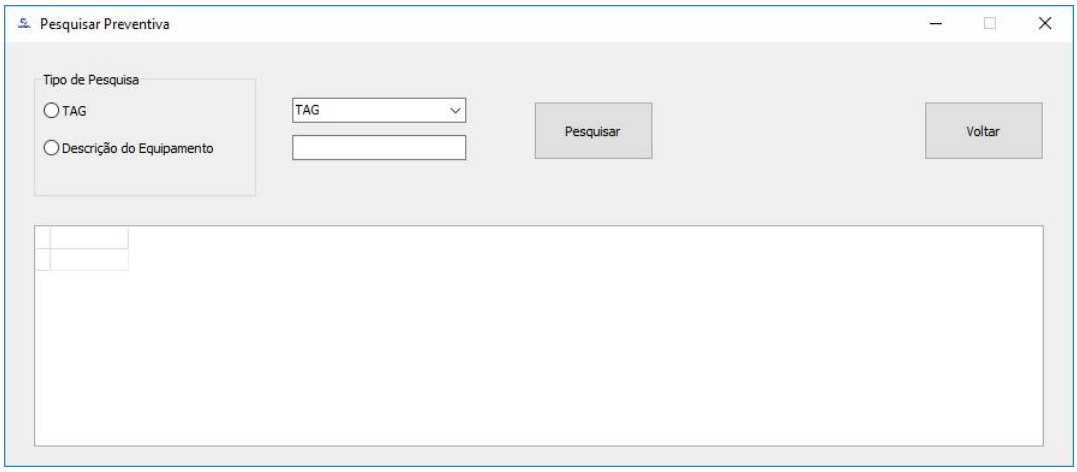

Figura 4.11: Interface de pesquisa de manutenções preventivas

4.4 Product Backlog

O Product Backlog é uma fase muito importante do Scrum, pois é nesta fase que acontece o levantamento de requisitos. Após o levantamento de dados, é analisado o tipo de manutenção a ser feita. Quando decidido, a equipe de manutenção poderá criar ordens de serviço baseada nos dados coletados.

Na ferramenta é possível escolher entre a **Manutenção Corretiva** e **Manutenção Preditiva** referente ao *framework*. O quinto item da barra de menu é o de Ordem de Serviço, onde o usuário poderá **Criar OS** , **Fechar OS** ou **Pesquisar OS**.

Na criação da OS será feita uma análise do equipamento e será identificado qual o tipo de manutenção ocorrerá. Assim, nesta interface o usuário poderá criar uma ordem de serviço escolhendo a TAG de um equipamento e preenchendo os dados do tipo de serviço, tipo de manutenção, a data de solicitação e o setor responsável, como ilustrado na Figura 4.12. Ao termino da criação, é permitido ao usuário imprimir o relatório da OS criada.

4.5 Daily Scrum

No *framework* o Daily Scrum é uma reunião que acontece para fazer o levantamento das ferramentas utilizadas na manutenção e também para evitar dúvidas durante a mesma. Após esses passos,

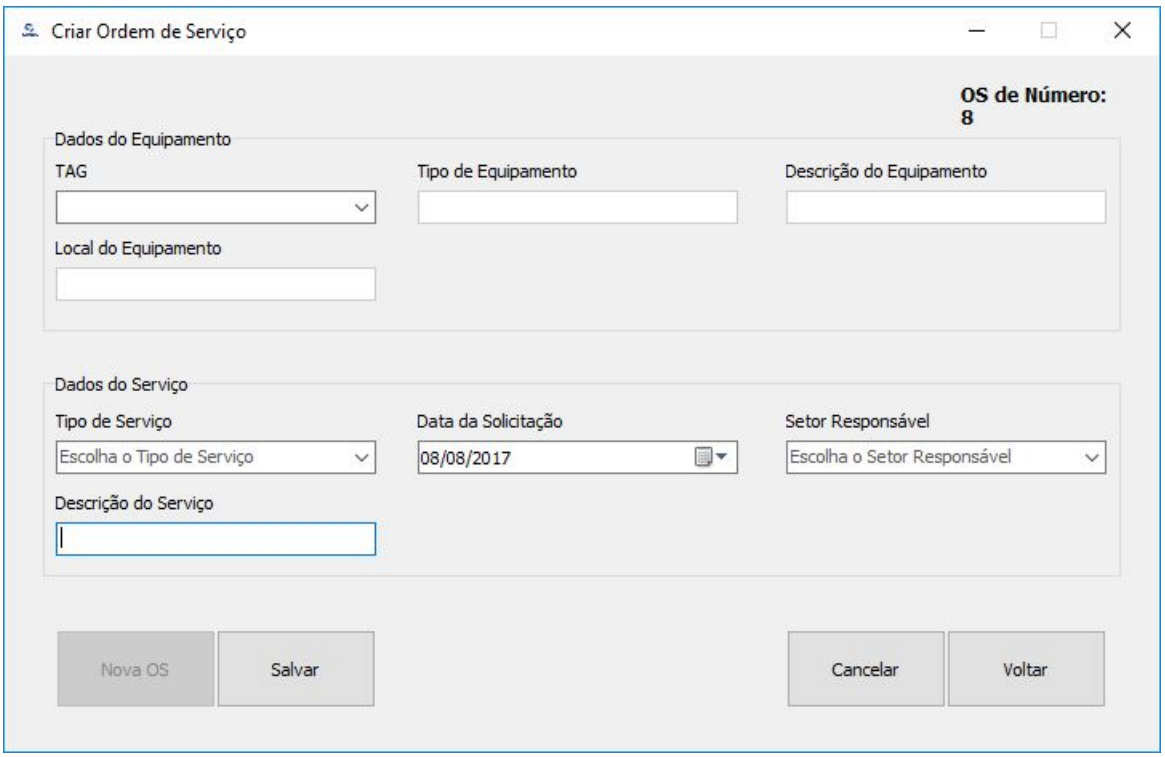

Figura 4.12: Interface de criação de ordem de serviço

a manutenção será executada e a equipe poderá acessar a interface de fechar ordem de serviço para guardar os dados recolhidos e outras como, a hora de início e a hora final da OS, a data de início e fim, o serviço que foi executado e os executantes da OS.

Quando aberta uma OS acontecerão reuniões para analisar os equipamentos e peças que serão utilizadas, quando o usuário precisar fechá-la deverá acessar a funcionalidade para fechar a OS, ilustrada na Figura 4.13.

4.6 Sprint Review

O Sprint Review no *framework* acontece após as manutenções, onde a equipe de manutenção deverá informar a liderança quais foram os gastos com a manutenção e as peças que foram usadas. No sistema a funcionalidade que representa essa descrição será a de geração de relatórios (**Relatório de OS**).

Nesta funcionalidade, o usuário poderá realizar pesquisas por OS aberta, fechada, pelo número da OS ou listar todas as OS. Também é possível fechar OS e imprimir alguma OS aberta caso necessário.

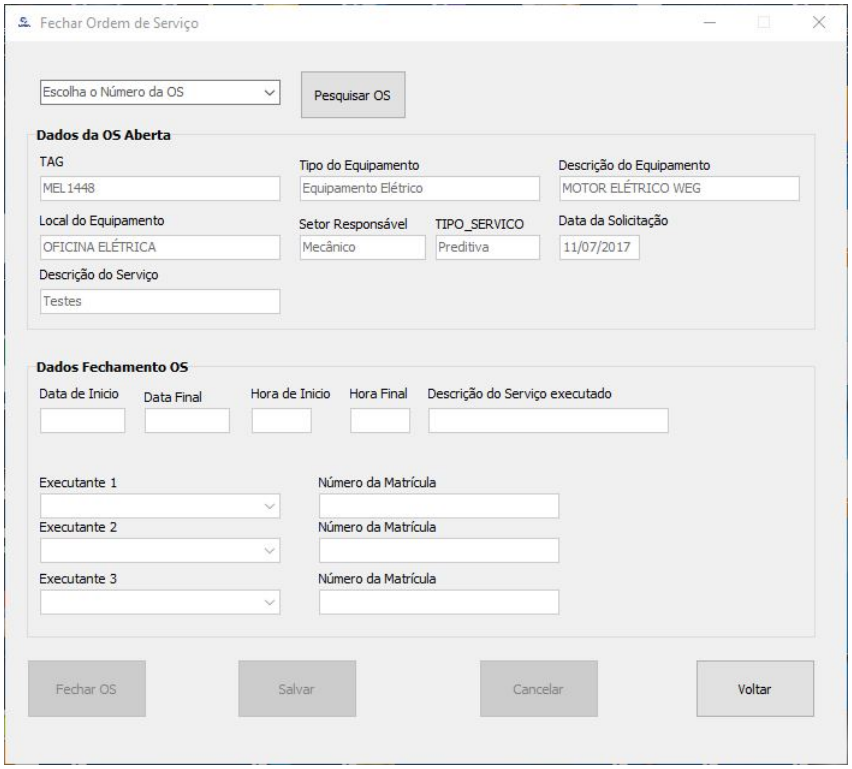

Figura 4.13: Interface de fechamento de ordem de serviço

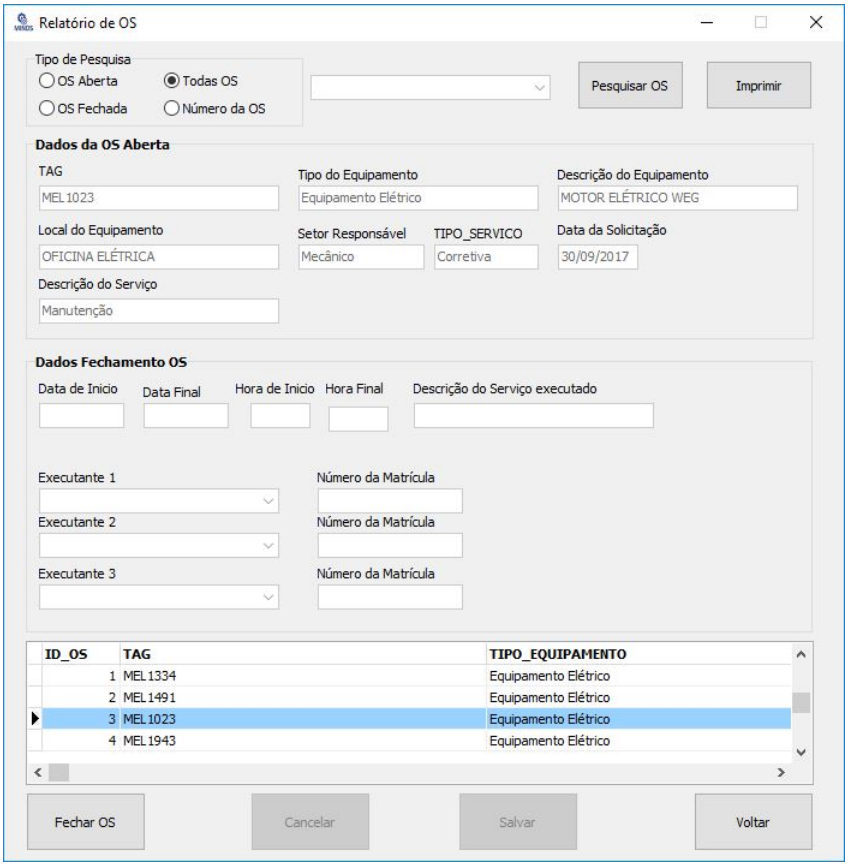

Figura 4.14: Interface de relatório de ordem de serviço

Figura 4.15: Framework Geral da Manutenção, extraído de Silva (2017)

CAPÍTULO 5

Resultados e Conclusão

5.1 Considerações Iniciais

Para a solução do problema de gerenciamento de manutenção, foi proposta a ferramenta apresentada no Capítulo 4. Tal ferramenta foi avaliada em uma empresa cuja manutenção não é gerenciada por um software, ou seja, é controlada de maneira não tecnológica. O acompanhamento dos serviços gerados é feito apenas por formulários que são impressos e preenchidos manualmente.

O restante deste capítulo é organizado da seguinte maneira: na Seção 5.2 é apresentada a caracterização dos participantes; na Seção 5.3 é relatada a condução da avaliação; os resultados são discutidos na Seção 5.4; por fim, na Seção 5.5 são expostas as considerações finais deste trabalho.

5.2 Caracterização dos Participantes

A avaliação foi realizada para um grupo de dez funcionários, dos quais estavam presentes líderes, supervisores e *trainees*, com uma grande variação de idade como ilustrado na Figura 5.1. Para o levantamento das características dos participantes foi solicitado aos mesmos que preenchessem um formulário com perguntas relacionadas à sua função e experiência na profissão e empresa (Apêndice A).

Os resultados indicam uma variação de tempo de profissão, de empresa e de função. O tempo de profissão varia entre alguns meses até aproximadamente 30 anos de profissão, conforme ilustrado na Figura 5.2.

Há funcionários com apenas poucos meses de contratação na empresa, mas também há funcionários com quase 30 anos de atuação. A maioria apresenta até 5 anos de profissão, como demonstra a Figura 5.3.

O tempo de atuação na função atual é ilustrado na Figura 5.4. Apenas quatro funcionários exercem funções diferentes desde a sua contratação. Os demais atuam na função atual por volta de cinco anos. Também é possível analisar a média geral de tempo na Tabela 5.1.

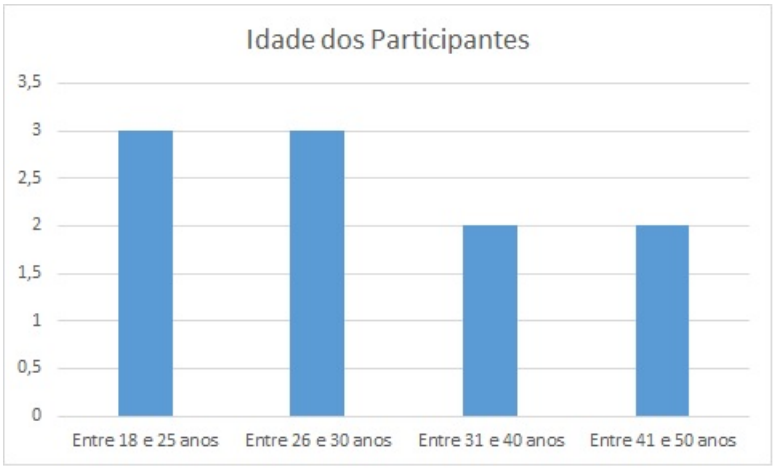

Figura 5.1: Idade dos Participantes.

Figura 5.2: Tempo de atuação na profissão.

Figura 5.3: Tempo de atuação na empresa.

Figura 5.4: Tempo de atuação na função atual.

Tabela 5.1: Médias de Tempo de Profissão, Empresa e Função

Tempo Profissão	Tempo Empresa	Tempo Função
9,4	**10,05**	**5,2**

5.3 Condução da Avaliação

Durante a condução da avaliação, foram demonstradas as funcionalidades da ferramenta e a sua ligação ao *framework*. Foram apresentadas operações como as de abertura de ordens de serviço e cadastro e pesquisa de equipamentos.

Ao termino da exibição foi aplicado uma pesquisa de opinião referente à ferramenta proposta, contido no Apêndice B. A partir das respostas do formulário de caracterização foram gerados gráficos como os apresentados na próxima seção, para melhor análise dos dados.

5.4 Resultados e Discussão

A partir dos dados coletados foram gerados gráficos para sua melhor análise da pesquisa de opinião. Assim, é possível notar os pontos positivos da ferramenta e, se houver, os pontos negativos.

Nesta pesquisa de opinião a avaliação foi feita por meio da atribuição de notas que variaram entre 1 e 5, onde 1 representa que a ferramenta não satisfaz os requisitos da pergunta, e a nota máxima (cinco) representa que a ferramenta satisfaz todos os requisitos.

A primeira questão, que se refere a eficiência da ferramenta no gerenciamento de manutenção, é ilustrada no gráfico da Figura 5.5. Nesta questão, 50% dos entrevistados atribuíram notas máximas, 40% atribuíram notas 4 e apenas 10% atribuíram notas 3. Com tais notas, fica visível que a ferramenta pode ser eficiente e pode proporcionar um bom gerenciamento de manutenção.

A segunda questão diz respeito às informações necessárias para o gerenciamento de manutenção, onde as respostas ficaram divididas entre notas 5 e notas 4, como ilustrado na Figura 5.6. Com isso, conclui-se que as informações requeridas para o gerenciamento da manutenção são de fato essenciais.

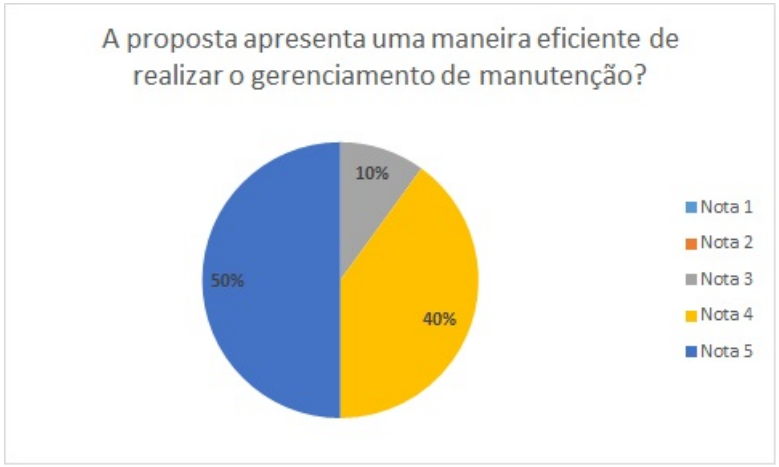

Figura 5.5: Primeira questão da Pesquisa de Opinião

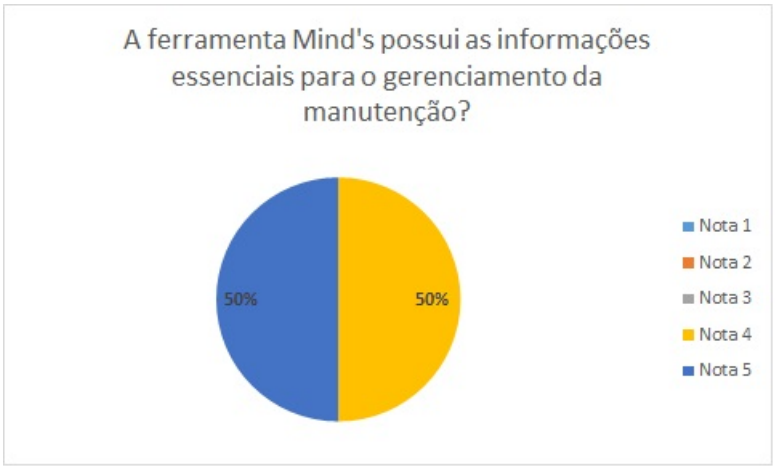

Figura 5.6: Segunda questão da Pesquisa de Opinião

A terceira questão, presente na Figura 5.7, está relacionada ao uso da ferramenta para manter o controle das manutenções preventivas que serão executadas. Essa funcionalidade foi muito bem aceita, pois 80% dos entrevistados deram notas 5, 10% deram notas 4 e 10% deram notas 3.

Figura 5.7: Terceira questão da Pesquisa de Opinião

A quarta questão corresponde à funcionalidade de ordem de serviço e às informações requeridas

para realizar o seu gerenciamento. Como ilustrado no gráfico da Figura 5.8, 90% dos entrevistados estão satisfeitos com esta funcionalidade e suas informações requeridas. Apenas 10% apontaram a falta de itens para o seu gerenciamento, discutida com maiores detalhes na quarta questão.

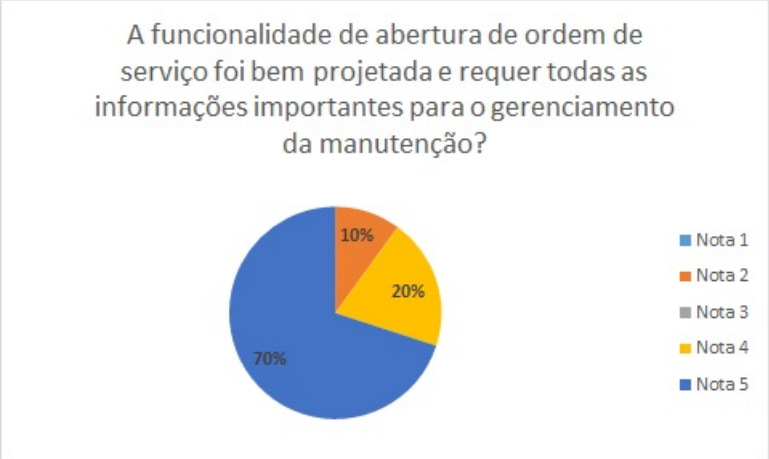

Figura 5.8: Quarta questão da Pesquisa de Opinião

A quinta questão, presente na Figura 5.9, está relacionada à usabilidade, isto é, a facilidade em que se pode utilizar a ferramenta. Os entrevistados se mostraram satisfeitos, pois 80% atribuíram notas 5 e 10% notas 4. Somente 10% das notas foram insatisfatórias, o que pode ser justificado pela pouca familiaridade do participante com dispositivos tecnológicos.

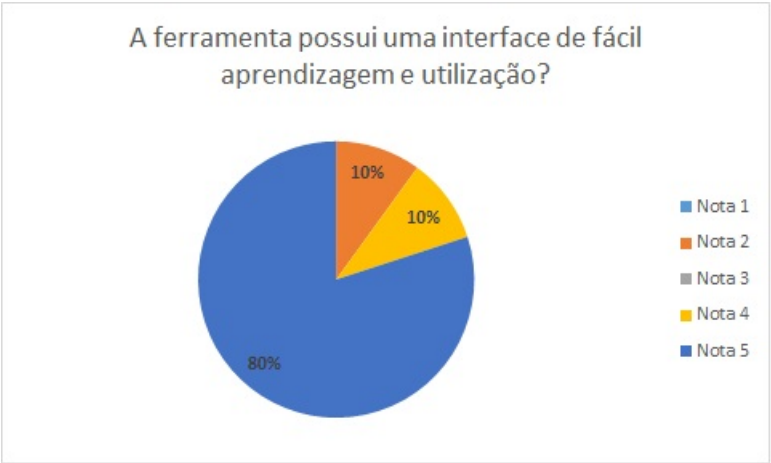

Figura 5.9: Quinta questão da Pesquisa de Opinião

A sexta questão diz respeito ao auxílio que a ferramenta pode fornecer ao gerenciamento de manutenção na empresa. A análise do gráfico contido na Figura 5.10 indica que a ferramenta pode apoiar o gerenciamento, pois os participantes atribuíram as duas maiores notas.

A sétima questão está relacionada à aplicação da ferramenta na empresa, em que as possíveis respostas eram: “Sim”, “Sim, com algumas adaptações” e “Não”. A maioria dos participantes (80%) opinaram que a ferramenta poderia ser utilizada, porém com algumas adaptações. Enquanto que 20% responderam que “Sim”, ou seja, a ferramenta já estava adequada para o uso. O gráfico desta questão pode ser analisado na Figura 5.11.

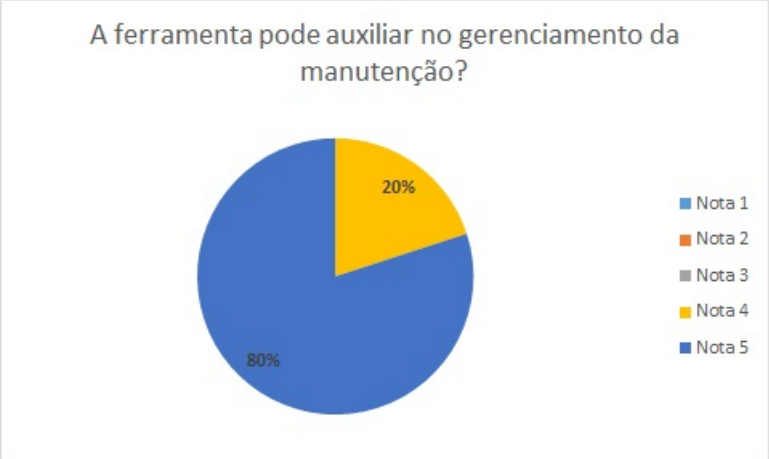

Figura 5.10: Sexta questão da Pesquisa de Opinião

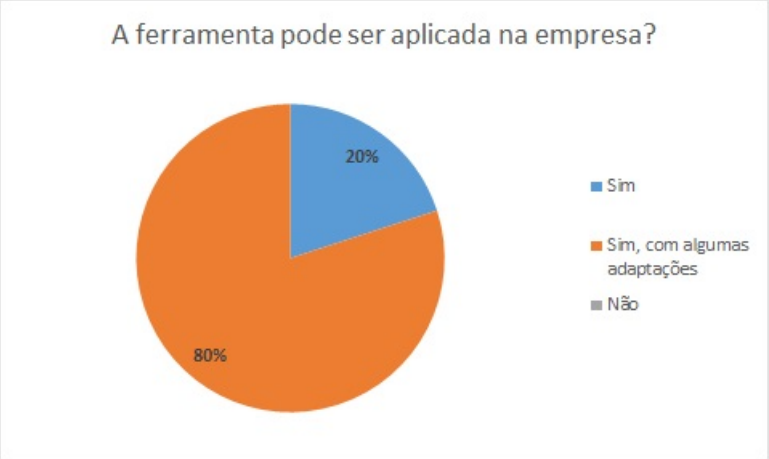

Figura 5.11: Sétima questão da Pesquisa de Opinião

A oitava questão é discursiva e corresponde às opiniões dos participantes sobre possíveis adaptações e inclusão de itens. Alguns participantes sugeriram que a ferramenta poderia conter uma descrição mais detalhada dos equipamentos.

Também foi sugerida a inclusão de outras funcionalidades que poderiam auxiliar no gerenciamento da manutenção, como a apresentação do estoque de peças na interface de Ordens de Serviço, a criação de opções para o gerenciamento de outros tipos de manutenção, a visualização de documentos de segurança dos equipamentos no momento da solicitação de uma ordem de serviço e, por fim, o controle das peças já trocadas e dos custos gerados.

5.5 Considerações Finais

A ferramenta foi desenvolvida com o intuito de gerenciar a manutenção industrial, ajudando assim a minimizar os problemas discutidos no Capítulo 1. A proposta de apoio ao gerenciamento das manutenções foi executada se baseando no *framework* apresentado no Capítulo 4.

A concepção e utilização da ferramenta seguem estes os mesmos princípios. Para a validação da ferramenta foi aplicada uma pesquisa de opinião para analisar a eficácia da mesma.

A pesquisa de opinião realizada mostrou que a ferramenta foi bem aceita pelos participantes. A

análise dos dados indicam que a ferramenta proposta é uma forma inovadora de realizar o gerenciamento de manutenção, sendo uma alternativa ao método atual utilizado. Os participantes propuseram algumas melhorias na ferramenta visando a sua evolução e ampliação de suas funcionalidades.

A ferramenta foi projetada de uma maneira em que possa ser utilizado em qualquer empresa. Entretanto, é possível que uma empresa precise que alguma alteração seja feita, ou que seja inserida alguma nova funcionalidade para se adequar melhor às suas necessidades. O que pode ser constatado no caso da empresa onde foi feita a pesquisa.

APÊNDICE

A

Caracterização dos Participantes

1 - Nome:

2 - Idade:
( )Entre 18 e 25 anos.
( )Entre 26 e 30 anos.
( )Entre 31 e 40 anos.
( )Entre 41 e 50 anos.
( )Entre 51 e 60 anos.
( )Acima de 60 anos.

3 - Tempo aproximado que atua na profissão:

4 - Tempo aproximado que atua na empresa:

5 - Função que exerce na empresa:

6 - Tempo aproximado que atua na função atual:

APÊNDICE

B

Pesquisa de Opinião - Ferramenta

1 - A proposta apresenta uma maneira eficiente de realizar o gerenciamento de manutenção?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

2 - A ferramenta Mind's possui as informações essenciais para o gerenciamento da manutenção?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

3 - É possível manter um controle das manutenções preventivas agendadas com o uso da ferramenta?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

4 - A funcionalidade de abertura de ordem de serviço foi bem projetada e requer todas as informações importantes para o gerenciamento da manutenção?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

5 - A ferramenta possui uma interface de fácil aprendizagem e utilização?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

6 - A ferramenta pode auxiliar no gerenciamento da manutenção?

Nula 1( ) 2( ) 3( ) 4( ) 5( ) **Totalmente**

7 - A ferramenta pode ser aplicada na empresa?

( )Sim.

( )Sim, com algumas adaptações.
( )Não.

8 - Quais melhorias podem ser inseridas na ferramenta proposta?

Referências Bibliográficas

AGILEBUSINESSCONSORTIUN The dsdm agile project framework, 2014. [*On-line*], disponível em: <https://www.agilebusiness.org/resources/dsdm-handbooks/the-dsdm-agile-project-framework-2014-onwards>. Acesso em: Setembro/2016.

BECK, K.; ANDRES, C. *Extreme programming explained: Embrace change (2nd edition)*. Addison-Wesley Professional, 2004.

BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D. Manifesto for agile software development. 2001.

CRUZ, F. Artefatos scrum, 2016. [*On-line*], disponível em: <http://www.fabiocruz.com.br/outros/artefatos-scr um/>. Acesso em: Setembro/2016.

ELOGROUP Stage-gate: Uma ferramenta flexível para a otimização do portfólio de projetos, 2015. [*On-line*], disponível em: <http://elogroup.com.br/conhecimento/insights/stage-gate-uma-ferramenta-flexivel-para-a-otimizacao-do-portfolio-de-projetos-2/>. Acesso em: Fevereiro/2017.

GOYAL, S. Agile techniques for project management and software engineering. In: *Major Seminar on Feature Driven Development*, 2007.

KOLB, J. J. Desenvolvimento adaptativo de software (das), 2013. [*On-line*], disponível em: <http://jkolb.com.br/desenvolvimento-adaptativo-de-software-das/>. Acesso em: Setembro/2016.

KOSCIANSKI, A.; SANTOS SOARES, M. D. *Qualidade de software - 2ª edição: Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software*. Novatec, 2007.

MARÇAL, A. S. C.; FREITAS, B. C. C.; SOARES, F. S. F.; MACIEL, T. M. M.; BELCHIOR, A. D. Estendendo o scrum segundo as áreas de processo de gerenciamento de projetos do cmmi. *CLEI Eletronic Journal*, 2007.

MITTAL, N. The burn-down chart: An effective planning and tracking tool. scrum alliance. scrum alliance, 2013. [*On-line*], disponível em: <https://www.scrumalliance.org>. Acesso em: Novembro/2016.

PAVAN KUMAR, G. Build your project using feature driven development. *International Project Management Association (IPMA)*, 2009.

PRESSMAN, R. *Software engineering: A practitioner's approach*. 6 ed. New York, NY, USA: McGraw-Hill, Inc., 2005.

RETAMAL, A. M. Feature-driven development. heptagon, 2008. [*On-line*], disponível em: <http://heptagon.com.br>. Acesso em: Outubro/2016.

RIBEIRO, A. L. D.; DE SOUSA, F. D. C. C.; ARAKAKI, R. Gerenciamento de projetos tradicional x gerenciamento de projetos á gil: Uma análise comparativa. In: *Congresso Internacional de Gestão da Tecnologia e Sistemas de Informação*, 2006.

SABBAGH, R. *Scrum: Gestão ágil para projetos de sucesso*. Casa do Código, 2013.

SANTOS SOARES, M. D. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação ISSN 1677-3071 doi: 10.21529/RESI*, v. 3, n. 1, 2004.

SCHWABER, K. *Agile project management with scrum*. Redmond, WA, USA: Microsoft Press, 2004.

SCHWABER, K.; SUTHERLAND, J. Guia do scrum: Um guia definitivo para o scrum: As regras do jogo. 2014.

SENE, P. F. D. Gerenciamento ágil de projetos, 2010. [*On-line*], disponível em: <http://www.leansixsigma.com.br/acervo/1815618.PDF>. Acesso em: Outubro/2016.

SILVA, D. R. D. Manutenção industrial apoiada por métodos ágeis. 2017.

SILVEIRA, C. B. Manutenção industrial: Como funciona?. citisystems, 2012. [*On-line*], disponível em: <https://www.citisystems.com.br/manutencao-industrial-como-funciona/>. Acesso em: Novembro/2016.

SOMMER, A. F.; SLAVENSKY, A.; NGUYEN, V. T.; STEGER-JENSEN, K.; DUKOVSKA-POPOVSKA, I. Scrum integration in stage-gate models for collaborative product development x2014; a case study of three industrial manufacturers. In: *2013 IEEE International Conference on Industrial Engineering and Engineering Management*, 2013, p. 1278–1282.

TEIXEIRA, D. D.; PIRES, F. J. A.; DE SOUSA, J. P. G.; PINTO, T. A. G. P. S. Dsdm–dynamic systems development methodology. *Faculdade de Engenharia da Universidade do Porto. Disponível em: http://paginas. fe. up. pt/˜ aaguiar/es/artigos% 20finais/es_final_14. pdf-Acesso em: Setembro/2016*, v. 27, p. 05–09, 2005.

TELES, V. M. D. Manifesto ágil, 2008. [*On-line*], disponível em: <http://www.desenvolvimentoagil.com.br/>. Acesso em: Novembro/2016.

VIEIRA, D. Scrum: A metodologia ágil explicada de forma definitiva. mindmaster, 2014. [*On-line*], disponível em: <http://www.mindmaster.com.br/scrum/>. Acesso em: Novembro/2016.

WAZLAWICK, R. *Engenharia de software: Conceitos e práticas*. Elsevier Brasil, 360 p., 2013.

XAVIER, J. N.; DORIGO, L. C. A importância da gestão na manutenção ou como evitar as "armadilhas"na gestão da manutenção. *Congresso Brasileiro de ManutenÃ§Ã£o*, 2005.

YANG, G.; YU, S.; CHEN, G.; CHU, J. Agile industrial design management based on scrum. In: *2010 IEEE 11th International Conference on Computer-Aided Industrial Design Conceptual Design 1*, 2010, p. 889–891.