ANEXO I

METODOLOGIA DE GESTÃO EM DESENVOLVIMENTO DE SOFTWARE DA AGÊNCIA NACIONAL DE MINERAÇÃO (VERSÃO 1.0)

1.INTRODUÇÃO

1.1. Este documento tem o propósito de descrever e normatizar o processo de desenvolvimento, manutenção e testes de sistemas de informação da ANM. A Metodologia de Gestão em Desenvolvimento de Software (MGDS) aqui descrita envolve diferentes processos, dentre os quais, o processo para desenvolvimento de novos sistemas, baseado no SCRUM, um modelo de desenvolvimento ágil amplamente utilizado e consagrado no mercado mundial, além de adaptações e outras técnicas complementares, incluindo os processos de sustentação de sistemas e de realização de manutenções evolutivas de pequeno porte.

1.2. Nas seções a seguir, serão detalhados, dentre outros aspectos, o processo principal de desenvolvimento baseado em SCRUM, denominado Processo de Desenvolvimento Ágil (PDA) e os processos auxiliares, denominados Processo de Sustentação de Sistemas (PSS) e Processo de Evolução de Pequeno Porte (PEP).

1.3. A Divisão de Desenvolvimento de Sistemas (DDSI) da Gerência de Tecnologia, Gestão e Suporte à Informação (GTGS) vem desenvolvendo ações voltadas à implementação dos direcionamentos estratégicos definidos pelos órgãos orientadores, normativos e fiscalizadores do Governo Federal para a Administração Pública.

1.4. Neste sentido, a DDSI, de forma planejada, vem trabalhando na padronização, simplificação e divulgação da estratégia de melhoria contínua dos processos organizacionais que visam o uso racional da Tecnologia da Informação.

1.5. A Metodologia de Gestão em Desenvolvimento de Software (MGDS) da ANM está inserida em um contexto organizacional de gestão dos serviços de concepção, projeto, desenvolvimento, sustentação, teste, integração, implantação e documentação de software mensurados em Pontos de Função para esta autarquia.

1.6. Esta metodologia tem o objetivo de estabelecer procedimentos para a gestão dos processos e serviços previstos dentro do escopo de Fábrica de Software (FSW), responsável por executar os serviços de concepção, projeto, desenvolvimento, sustentação, teste, integração, implantação, documentação e métrica de sistemas.

1.7. De acordo com as diretrizes do PDTI da ANM, visando permitir o adequado investimento em sistemas informatizados, de acordo com a necessidade dos órgãos demandantes da ANM, os serviços de desenvolvimento de software serão realizados de forma terceirizada, com a contratação de uma fábrica de software.

1.8. É importante observar que, conforme a Súmula TCU 269 (2012), a remuneração nas contratações de serviços de Tecnologia da Informação deve estar vinculada à entrega de resultados e ao atendimento de níveis de serviço. Portanto, é relevante que o modelo de

remuneração do contrato firmado defina os níveis de serviço e os critérios de qualidade para a aceitação dos resultados entregues ao final das *sprints* do processo ágil.

2. DEFEITOS DE SOFTWARE

2.1. Com efeito na aceitação de entregas, estão definidos nesta seção os níveis de criticidade dos defeitos de software, cujo conceito se estende a quaisquer dos processos em questão.

2.2. Defeito Impeditivo.

2.2.1. De caráter crítico, que envolve situações tais como:

a) A impossibilidade de conclusão de um fluxo principal;
b) Inoperância de integrações externas essenciais ao sistema;
c) Divergência de regra de negócio em relação à especificação, que impossibilite o uso do sistema;
d) Performance ou tempo de resposta em nível que impossibilite o uso do sistema;
e) Corrupção de múltiplos registros de dados em produção.

2.3. Defeito não impeditivo.

2.3.1. De caráter regular, envolve situações tais como:

a. Dificuldades para concluir fluxos não essenciais do sistema e validações inconsistentes;
b. Integrações não críticas inoperantes;
c. Ausência de parâmetros ou tabelas de domínio que dificultem o uso do sistema;
d. Não-conformidades de layout e aspectos ineficientes de usabilidade;
e. Inconsistências de massa de dados para homologação que dificultem testes do sistema;

3. PROCESSO DE DESENVOLVIMENTO ÁGIL (PDA)

3.1. Nesta seção será descrito o processo de desenvolvimento ágil, baseado no SCRUM, que é utilizado para o desenvolvimento de novos sistemas e para as evoluções elencadas como projeto, que tipicamente são evoluções de maior porte e/ou de alta criticidade, e cuja categorização é a cargo do Órgão Contratante.

3.2. O PDA aqui descrito, seguindo as diretrizes que são empregadas no processo SCRUM, tem como principais valores:

- Interação e confiança entre os participantes;
- Janela fixa de tempo para cada ciclo de desenvolvimento;
- Adaptação rápida às mudanças;
- Documentação concisa e objetiva;
- Entrega rápida de produtos e satisfação das áreas de negócios;
- Revisão e melhoria contínuas no processo.

3.3. Considerando o exposto, é possível observar que neste processo, os produtos são continuamente incrementados, agregando valor à área de negócio desde os primeiros ciclos de desenvolvimento. O foco na documentação é reduzido, mas se mantém um conjunto de artefatos plenamente satisfatório para exprimir o sistema em termos documentais. Por fim, todo o trabalho é continuamente avaliado e monitorado, de forma que melhorias são aplicadas constantemente nas experiências entre os participantes.

3.4. Termos Utilizados no PDA

- **Backlog** – coleção de funcionalidades definidas pelo cliente e que geram valor para o negócio;
- **Design Thinking** - é o conjunto de ideias e *insights* para abordar problemas, relacionados a futuras aquisições de informações, análise de conhecimento e propostas de soluções. Etapa inicial da fase de requisitos;
- **Sprint** – iteração no processo de desenvolvimento, na qual é produzida uma parte do sistema, previamente definida pelo cliente;
- **Kanban** – técnica utilizada em processos industriais que consiste no simples mapeamento das atividades, e as unidades de trabalho responsáveis por elas, sendo aqui aplicada ao desenvolvimento de software através do Quadro Kanban de atividades;
- **História de usuário** – é a menor unidade de funcionalidade que possui valor para o cliente, e que normalmente representa um cenário de uso do sistema;
- **MVP** - sigla de *Minimum Viable Product* e significa produto mínimo viável. É uma prática de administração de empresas que consiste em lançar um novo produto ou serviço com o menor investimento possível.

3.5. Papéis Definidos no PDA

3.5.1. Product Owner (PO)

3.5.1.1 É o representante do Órgão Contratante responsável por:

- Conhecer as necessidades relacionadas ao sistema;
- Validar a visão do produto;
- Estabelecer, priorizar e refinar as necessidades continuamente;
- Estar disponível para dúvidas e questionamentos do time de desenvolvimento;
- Participar das reuniões de demonstração de Sprints e decidir pela aceitação de entregas;

3.5.2. SCRUM Master

3.5.2.1 É o representante da contratada responsável por:

- Priorizar e remover os impedimentos da equipe de desenvolvimento;
- Garantir o funcionamento do processo, ou seja, que a equipe utilize corretamente a MGDS;
- Evitar que membros da equipe implementem hierarquias;
- Facilitar e garantir as reuniões;

- Assegurar as entregas previstas em cada Sprint;
- Garantir a entrega dos releases de software conforme planejadas.

3.5.3. Gerente de Sistema

3.5.3.1 É o representante do Órgão Contratante responsável por:

- Reforçar os fundamentos do processo e garantir a correta execução das tarefas, atuando como SCRUM Master Interno;
- Garantir apoio ao Product Owner na priorização e demais atividades relacionadas às necessidades do produto, inclusive com o aprofundamento no entendimento do negócio;
- Definição de datas relacionadas ao plano de releases;
- Acompanhamento e discussão das atividades com a equipe de desenvolvimento diariamente, inclusive com inspeção dos resultados diários;
- Participar das reuniões de demo e retrospectiva de Sprints;
- Decidir pela homologação técnica de entregas;
- Definir questões que envolvam caráter técnico;
- Apoiar na resolução de conflitos e dificuldades da equipe contratada.

3.6. Visão Geral do Processo

3.6.1. Neste processo foi definida uma fase de **Iniciação**, onde se busca representar a visão inicial do produto. Após sua conclusão, se inicia a execução cíclica das Sprints, onde cada Sprint é composta de 3 fases: ***Discovery***, que contempla o refinamento das histórias de cada ciclo; ***Delivery***, que contempla a construção e implantação do produto planejado; e a ***Homologação*****,** onde o Product Owner verifica em detalhes o produto entregue. Esta visão macro do processo consta na Figura 1.

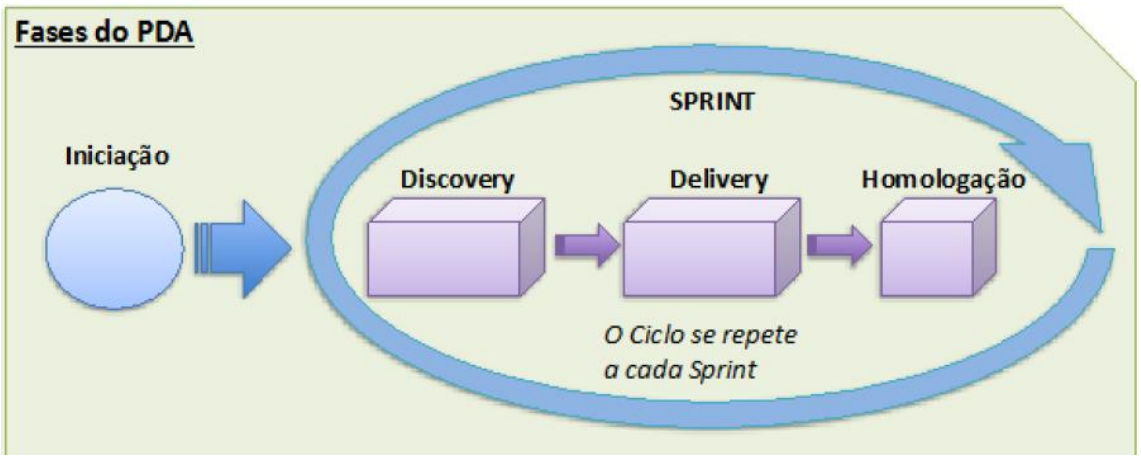

Figura 1 – Fases do PDA

3.6.2. Aproximando o processo e decompondo as fases envolvidas, é possível observá-lo através de uma perspectiva mais detalhada (Figura 2). A iniciação, abrangendo a definição de visão do produto, compreende, em linhas gerais, os objetivos do sistema a

ser desenvolvido, premissas arquiteturais, além do conjunto inicial previsto de histórias de usuário (backlog do produto), um plano de releases e demais necessidades de caráter não-funcional identificadas pela área demandante.

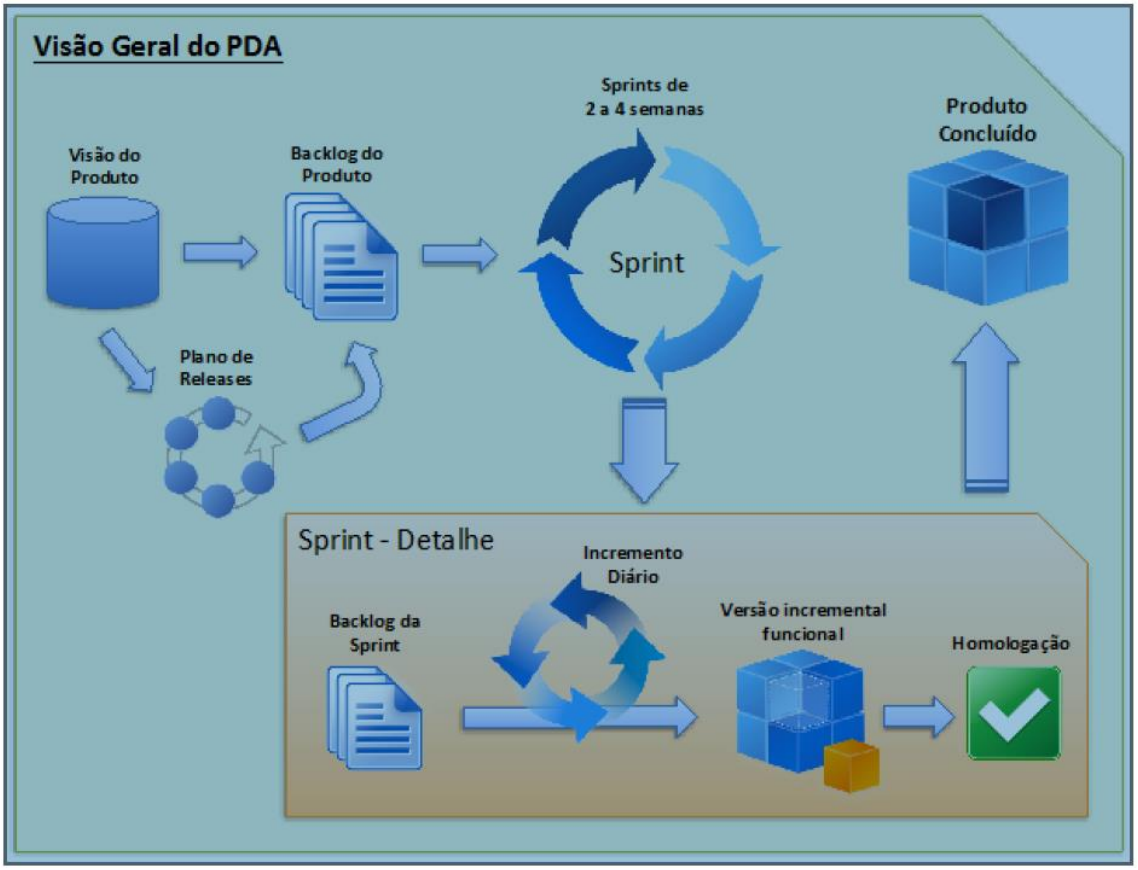

Figura 2 – Visão Geral Detalhada do PDA

3.6.3. Uma vez concluída a iniciação, o ciclo de Sprints começa a ser executado. Cada Sprint possui um conjunto de histórias selecionadas (Backlog da Sprint), que são refinadas (Discovery), e que através de incrementos diários, é concluída com um produto entregue contendo estas histórias planejadas (Delivery). Este fluxo permanece até que o produto seja completamente construído.

3.7. Fase de Iniciação

3.7.1. A fase de iniciação, **com duração média de uma semana**, é a primeira fase do projeto, onde se busca nivelar as necessidades e atingir um consenso entre todos os envolvidos, sobre qual produto deverá ser desenvolvido. Esta fase inclui o processo de **Design Thinking,** e corresponde à concepção de um projeto, como etapa inicial da fase de requisitos. A iniciação é realizada apenas uma vez para cada projeto. O escopo da concepção inclui a descrição macro dos requisitos, a partir da ideia ou solução trabalhada. A Figura 3 ilustra as etapas e artefatos detalhados da fase de iniciação.

3.7.1.1. A concepção, por meio do *Design Thinking*, é composta dos seguintes subprocessos:

1. Imersão/Empatia:
   a. Objetivo: Investigar previamente o trabalho, observar, compreender e conhecer o Problema. Empatia: "*Walk in a user's shoes*". Definir e validar o problema;
   b. Ações/ferramentas: pesquisa, entrevistas, observações, análise.

2. Definição:
   a. Objetivo: identificar, analisar, sintetizar e estruturar o problema no contexto do negócio.
   b. Ações/ferramentas: cartões de *insights*, mapas mentais.

3. Ideação:
   a. Objetivo: *brainstorming*, levantar ideias criativas que podem envolver soluções, criar hipóteses, categorizar e escolher a solução ou as melhores soluções.
   b. Ações/ferramentas: *brainstorming*, resumo dos i*nsights*, plano de *releases*.

4. Prototipação:
   a. Objetivo: construir protótipos físicos ou virtuais das principais soluções; entregar minipilotos de baixo custo para engajamento.
   b. Ações/ferramentas: prototipação e criação de croquis ou MVP (*Minimum Viable Product*) com o cliente, *storyboard*, *landing page*.

5. Teste e Validação:
   a. Objetivo: validar a ideia/solução prototipada, com *feedback* dos envolvidos; em caso de impedimento, volta para ideação.
   b. Ações/ferramentas: evolução, simpatizar, obter feedback do demandante, validar a ideia.

3.7.2. Definição da Visão do Produto

3.7.2.1. O Documento de Visão contém a missão do sistema a ser desenvolvido, os conceitos básicos relacionados à sua área de negócio, as necessidades que justificam seu desenvolvimento e macroobjetivos a serem cumpridos. Além disso, devem estar explicitados requisitos arquiteturais específicos e demais informações de cunho geral relacionadas à visão do sistema.

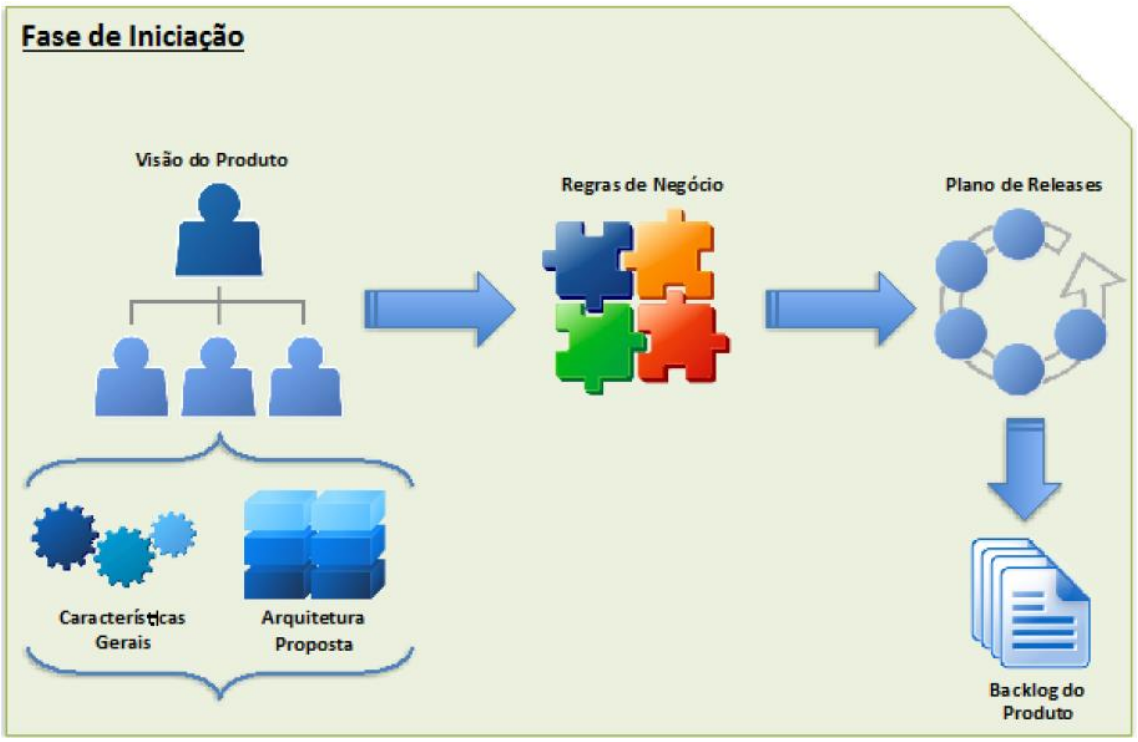

Figura 3 – Fase de Iniciação

3.7.3. Definição de Regras de Negócio

3.7.3.1 Após a delimitação da visão do produto, devem ser elencadas as principais regras de negócio relacionadas ao sistema. Tais regras tem grande importância para entender premissas concretas que orientem o levantamento do Backlog do Produto. Estas regras devem ser revisadas, ampliadas e aprimoradas a cada fase de Discovery.

3.7.4. Estabelecimento do Plano de Releases

3.7.4.1 O Plano de Releases envolve definir em alto nível as versões significativas do sistema que devem ser alcançadas. Eventualmente, dependendo do tipo de negócio ou tamanho do sistema, apenas uma versão é desejável. Mas em geral, vários marcos podem ser estabelecidos. Para facilitar a comunicação e a validação dos releases com o Product Owner devem ser feitas prototipações, croquis ou MVP. Deste modo, é possível visualizar a associação destes objetivos de alto nível e funcionalidades com valor significante para o Product Owner. Um exemplo de relação entre releases e Sprints está demonstrado na Figura 4.

3.7.5. Estabelecimento do Backlog do Produto

3.7.5.1 O backlog do produto é uma lista de todas as histórias que devem ser necessárias na construção do produto, de maneira ordenada por prioridade. É de responsabilidade do Product Owner elicitar estas histórias, e priorizá-las de maneira que a ordem do backlog reflita o grau de importância de cada história.

3.7.5.2 Este artefato está em constante evolução e é sempre passível de alterações. A versão concluída na iniciação reflete uma visão geral das funcionalidades que o produto deve conter, de forma a delimitar uma noção de escopo para o projeto. Inclusão e exclusão de necessidades é algo comum e rotineiro, cuja revisão é realizada a cada Sprint.

3.7.5.3 Além do exposto, constam no backlog do produto eventuais correções necessárias e alterações de funcionalidades existentes.

3.7.6. Artefatos Resultantes

3.7.6.1 Os artefatos resultantes da fase de Iniciação são:

- Protótipos/ Croquis/ Storyboards/ MVP;
- Documento de Visão;
- Regras de Negócio;
- Plano de Releases;
- Backlog do Produto;

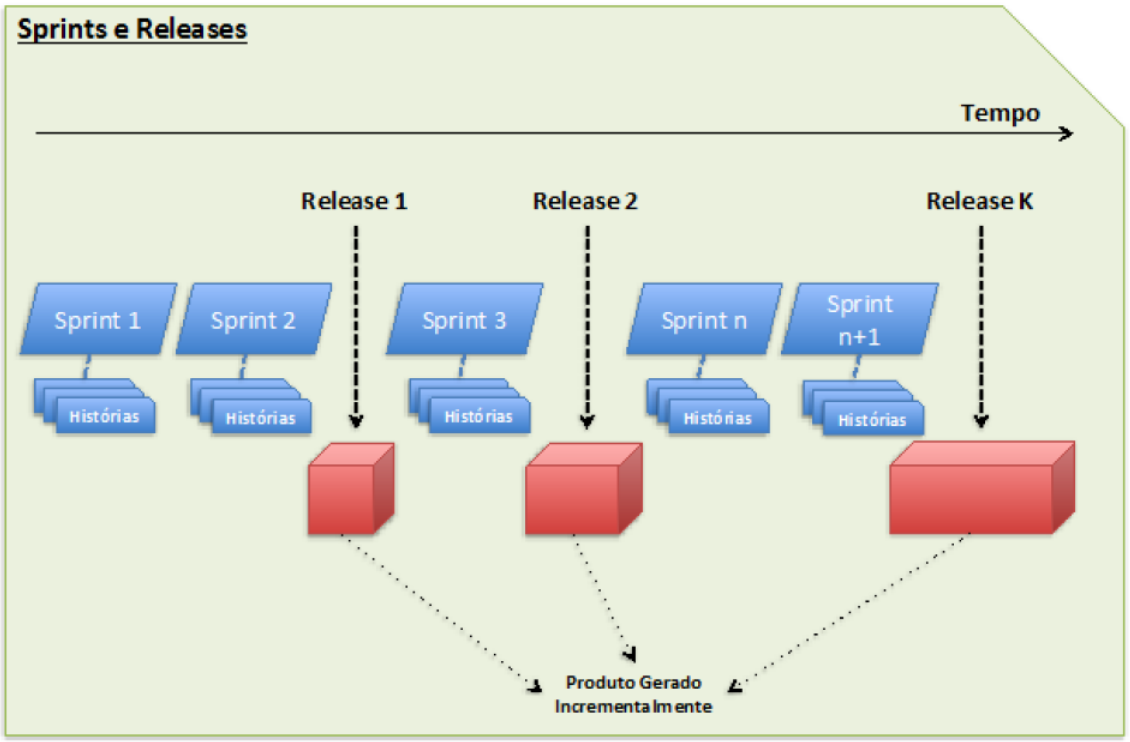

Figura 4 – Sprints e Releases

3.8. Fase de Discovery

3.8.1. A fase de Discovery, que é executada a cada Sprint, e tem duração de até 2 semanas corridas, compreende a etapa relacionada ao planejamento do conteúdo de cada Sprint, e consequentemente, com o refinamento deste conteúdo. Para isso, são realizadas algumas etapas, descritas a seguir.

3.8.2. Revisões da Visão, Backlog do Produto e Regras de Negócio:

3.8.2.1 A cada Discovery, é realizada uma revisão da Visão do Produto, com o intuito de verificar se a missão, premissas e características gerais estão mantidos conforme foi planejado no início do projeto. Em adição, o backlog do produto também é revisado, de tal forma que novas histórias possam ser incluídas, além de permitir a repriorização das histórias existentes, ou mesmo a exclusão de histórias que não se façam mais necessárias. Por último, devem ser revisadas também as regras de negócio do sistema, modificando-se regras existentes ou incluindo-se novas regras. Esta fase tipicamente envolverá até 4 horas.

3.8.3. Planejamento de Discovery

3.8.3.1 Faz-se necessário, após a ratificação do conteúdo de backlog e visão do produto, realizar o planejamento de discovery da Sprint a ser executada. Neste caso, a equipe se reúne com o PO e o SCRUM Master para definirem juntos as histórias a serem elencadas como **obrigatórias ou opcionais** naquela Sprint. A quantidade de histórias deve ser ajustada de acordo com a produtividade da equipe. Naturalmente, a produtividade tende a melhorar à medida em que o time compreende mais profundamente o sistema.

3.8.3.2 Delimitado o escopo aproximado da Sprint, deve haver uma contagem estimada das funcionalidades previstas, para que verificar se há viabilidade na realização das histórias selecionadas.

3.8.3.3 Esta fase tipicamente envolverá até 4 horas.

3.8.4. Refinamento do Backlog da Sprint e Modelo de Dados

3.8.3.1 Uma vez definidas as histórias que devem compreender o backlog da Sprint, estas devem ser refinadas em conjunto com o Product Owner. Nesta fase, haverá reuniões envolvendo toda a equipe para o amplo registro e entendimento de cada história, além da geração do modelo de dados atualizado que reflita aquilo que foi elicitado pela equipe.

3.8.3.2 Várias técnicas podem ser utilizadas neste período de refinamento, incluindo Entrevistas, Brainstorming e Prototipação.

3.8.3.2 Juntamente com as histórias, devem ser trazidos os critérios de aceitação providos pelo PO, que serão transformados em testes unitários na fase de Delivery. Além destes, será considerada a necessidade, em determinadas histórias de usuário, a depender da criticidade e da complexidade das regras de negócio, de que sejam feitos Testes Funcionais Automatizados, de acordo com tais critérios de aceitação, utilizando ferramenta padronizada pelo Órgão Contratante.

3.8.3.3 Tipicamente, esta etapa deve durar até 9 dias úteis de trabalho.

3.8.3.4 Na Figura 5, é possível visualizar a sequência e duração de cada etapa do Discovery.

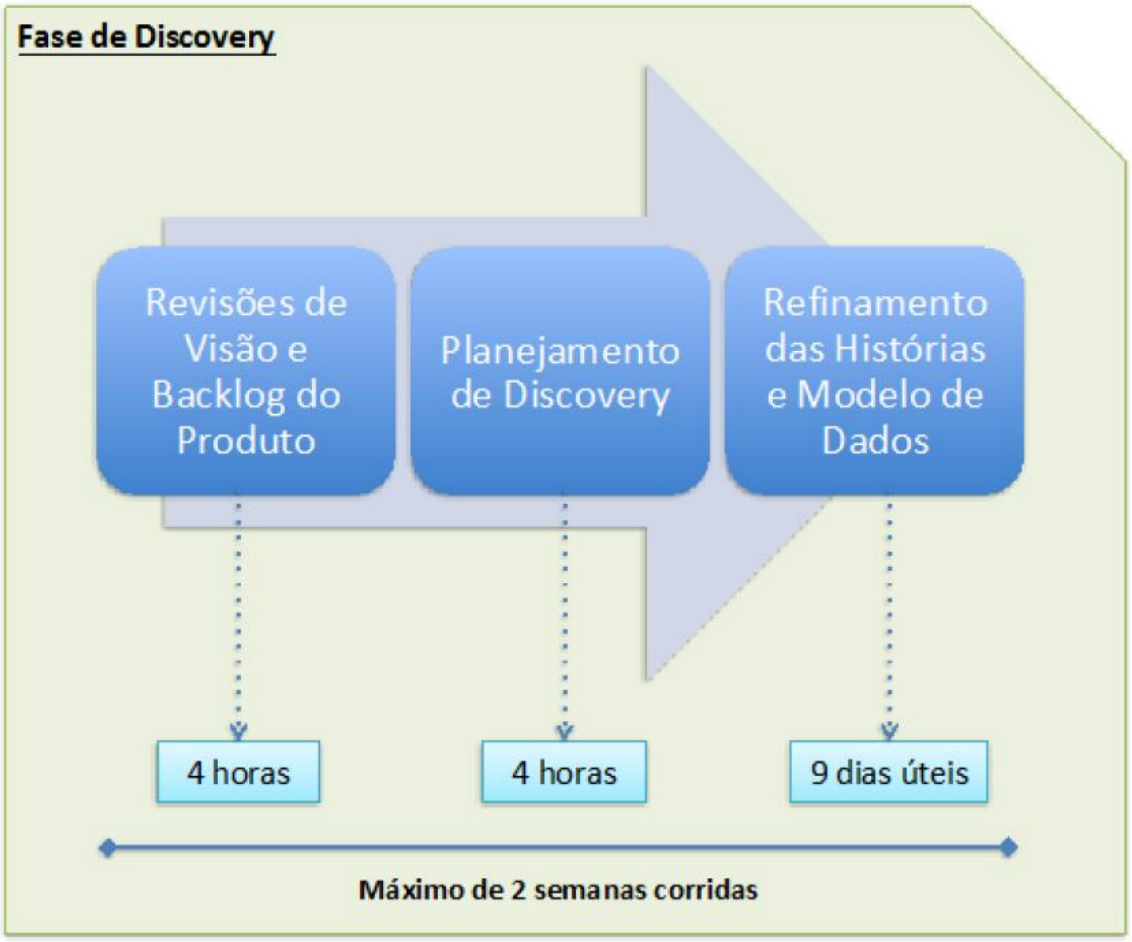

Figura 5 – Fase de Discovery

3.8.5. Artefatos resultantes

3.8.5.1 Os artefatos resultantes da fase de Discovery são:

- Visão do Produto, Backlog do Produto e Regras de Negócio atualizados;
- Histórias de Usuário refinadas, com protótipos de tela e critérios de aceitação, além da indicação de necessidade de testes funcionais automatizados;
- Modelo de Dados atualizado;

3.9. Fase de Delivery

3.9.1. A fase de Delivery, que também é executada a cada Sprint, logo após a fase de Discovery, tem como prazo previsto duas semanas corridas, e compreende a construção das necessidades pactuadas no planejamento da Sprint, ou seja, envolve a codificação e a entrega de um incremento do produto que está sendo tratado no projeto ágil. Para isso, são realizadas algumas etapas, descritas a seguir.

3.9.2. Planejamento de Delivery

3.9.2.1 No planejamento do delivery, as histórias da Sprint são subdivididas em tarefas, que são distribuídas entre os integrantes do time. Há um planejamento de metas de curto prazo para cada tarefa, para que possa se alcançar a construção do produto.

3.9.3. Criação de Testes Unitários

3.9.3.1 Esta metodologia preza pelo desenvolvimento orientado a testes (TDD – *Test Driven Development*), onde os testes unitários são construídos antes mesmo do código executável, para que, utilizando os critérios de aceitação definidos pelo Product Owner nas histórias de usuário, os testes orientem antecipadamente o próprio código a ser desenvolvido, em um processo incremental.

3.9.4. Implementação das Histórias de Usuário

3.9.4.1 Como objetivo principal da Sprint, serão implementadas as tarefas planejadas de forma a satisfazer as necessidades expressas nas histórias de usuário previstas para a Sprint.

3.9.5. Criação de Testes Funcionais/Automatizados (Testes de Requisitos, Testes de Regressão, Testes de Integração) e Testes Não Funcionais/ Automatizados ou Semi-Automatizados (Testes de Usabilidade, Testes de Performance, Testes de Segurança).

3.9.5.1 Esta etapa diz respeito à construção dos testes automatizados, funcionais e não funcionais, que podem ter sido planejados para a Sprint. Os testes funcionais e não funcionais serão desenvolvidos utilizando ferramentas que estejam padronizadas pelo Órgão Contratante (Ex: Selenium).

3.9.6. Execução de Testes

3.9.6.1 Antes de liberar o incremento do produto previsto na Sprint, a contratada deve executar os testes desenvolvidos para as histórias em questão, de forma que relatórios com o resultado da execução dos testes estejam disponíveis para verificação por parte do Órgão Contratante.

3.9.7. Pré-Homologação

3.9.7.1 Nesta etapa, o Órgão Contratante recebe oficialmente o produto para uma pré-homologação, que tem como finalidade verificar:

- Se o produto entregue atende ao checklist para admissão, conforme subseção a seguir;
- Se todas as histórias planejadas na Sprint estão contempladas no produto entregue;
- Se há defeitos de natureza impeditiva.

3.9.7.2 **Definição de PRONTO** (Checklist de Admissão do Produto)

3.9.7.2.1 Qualquer produto enviado para homologação por parte da CONTRATADA deve atender a uma série de critérios para sua admissão à implantação em ambiente de homologação, sem os quais o produto é rejeitado de imediato. Tais critérios estão listados a seguir:

- Código-fonte submetido ao controle de versões do Órgão Contratante;
- Existência de testes unitários e do Relatório de Testes;
- Existência de scripts de banco de dados com **dicionário de dados embutido nos metadados** (ausência apenas quando não houver mudança no modelo de dados)
- Existência de arquivo para geração de Build (ex: Arquivo de projeto Maven).

|

3.9.7.3 Aceitação da Demanda

3.9.7.3.1 Após realizar a inspeção do produto quanto à sua admissibilidade (item anterior), o gerente de sistemas, juntamente com o Product Owner, poderá:

- Executar testes automatizados, funcionais e não funcionais, que tenham sido solicitados, e consequentemente verificar se estão corretamente implementados ou mesmo se existem, além de observar os resultados da execução;
- Executar testes unitários ou verificar os relatórios de execução correspondentes, que possam envolver porções críticas do produto;
- Realizar alguns testes funcionais, pelo menos nos principais fluxos do produto entregue.

3.9.7.4.1 Após a realização destes testes, pode se proceder a uma das ações a seguir:

- **Rejeição**: caso sejam percebidos defeitos de natureza impeditiva em alguma história implementada;
- **Aceitação parcial**: caso a demanda possua alguns defeitos significativos de natureza não impeditiva ou não tenha coberto o escopo planejado de tal forma que ainda seja passível de aceitação;
- **Aceitação integral**: caso a demanda esteja em nível de qualidade tal que não sejam percebidos defeitos significativos, bem como envolva cumprimento do escopo planejado.

3.9.7.4.2 Todos os aspectos julgados relevantes devem ser registrados pelo Gerente de Sistemas e/ou Product Owner no Relatório de Não-Conformidade. Os defeitos percebidos na rejeição e na aceitação parcial devem **obrigatoriamente** fazer parte de um item de backlog da próxima Sprint, específico para correção dos defeitos, salvo determinação contrária do PO ou Gerente de Sistemas. Todos os entregáveis de software e documentação deverão estar versionados nos repositórios da CONTRATADA, inclusive o TAD (Termo de Aceite da Demanda), fator condicionador para o faturamento da demanda pela CONTRATANTE.

3.9.8. Retrospectiva

3.9.8.1 A etapa de retrospectiva diz respeito à melhoria contínua do processo. Nesta etapa, os integrantes se reúnem para discutir a Sprint que está sendo concluída, com foco nos desafios, oportunidades e problemas ocorridos. Não faz parte do escopo desta etapa a discussão sobre histórias de usuário e backlog do produto, ou seja, discute-se apenas o processo, e como melhorá-lo. Tipicamente, esta reunião leva até 4 horas.

3.9.9. Artefatos Resultantes

3.9.9.1 Os artefatos resultantes da fase de Delivery são:

- Histórias de Usuário implementadas e cujo código-fonte esteja submetido ao controle de versões;
- Testes unitários, funcionais e não funcionais, de integração, implementados e executados, com o resultado de testes;
- Demais artefatos relacionados ao deployment do sistema: projeto para criação do build (Ex: Projeto Maven), scripts de banco e manual de implantação.

3.9.9.2 Na Figura 6, é possível visualizar a sequência e duração das etapas desta fase.

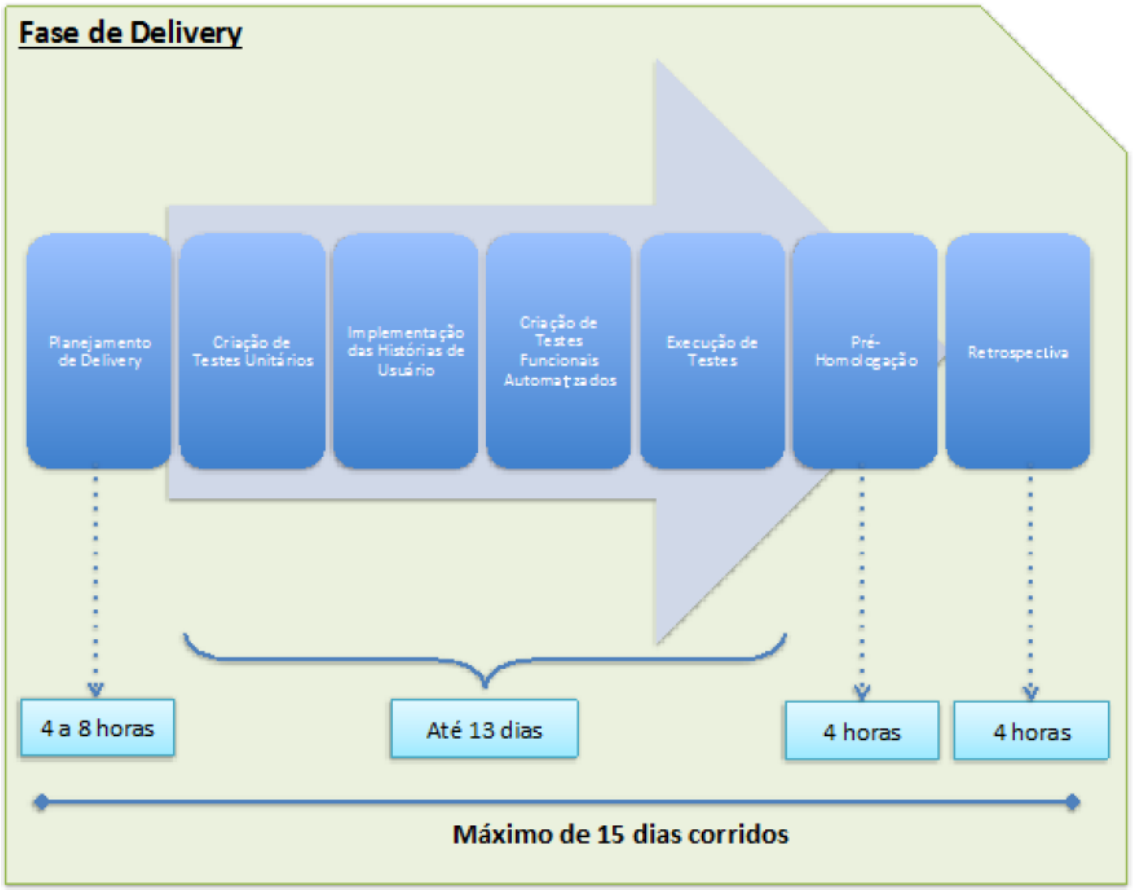

Figura 6 – Fase de Delivery

3.10. Fase de Homologação

3.10.1. Esta fase compreende apenas os testes e experimentação detalhados do produto entregue, por parte do PO e/ou Gerente de Sistemas, em até 5 dias úteis, para que possa haver alguma decisão de ordem negocial, como inclusão de novas regras, melhoria da implementação existente ou mesmo rejeição das regras implementadas. Qualquer problema ou observação deve ser acrescido ao Relatório de Não-conformidade, criado na Pré-Homologação.

3.11. Técnicas e Ferramentas Auxiliares ao Processo

3.11.1. Além das fases citadas anteriormente, o PDA envolve a utilização de técnicas e ferramentas auxiliares ao longo do processo, de forma a garantir a correta gestão e minimizar os riscos envolvidos. Tais práticas são detalhadas nas subseções a seguir.

3.11.2. Quadro de Tarefas com Kanban

3.11.2.1 Todos os projetos desenvolvidos no PDA devem ter um quadro apoiado pela técnica KANBAN contendo as tarefas previstas no projeto, separadas em raias que significam etapas e estados relacionados à execução. Um exemplo deste quadro pode ser visto na Figura 7.

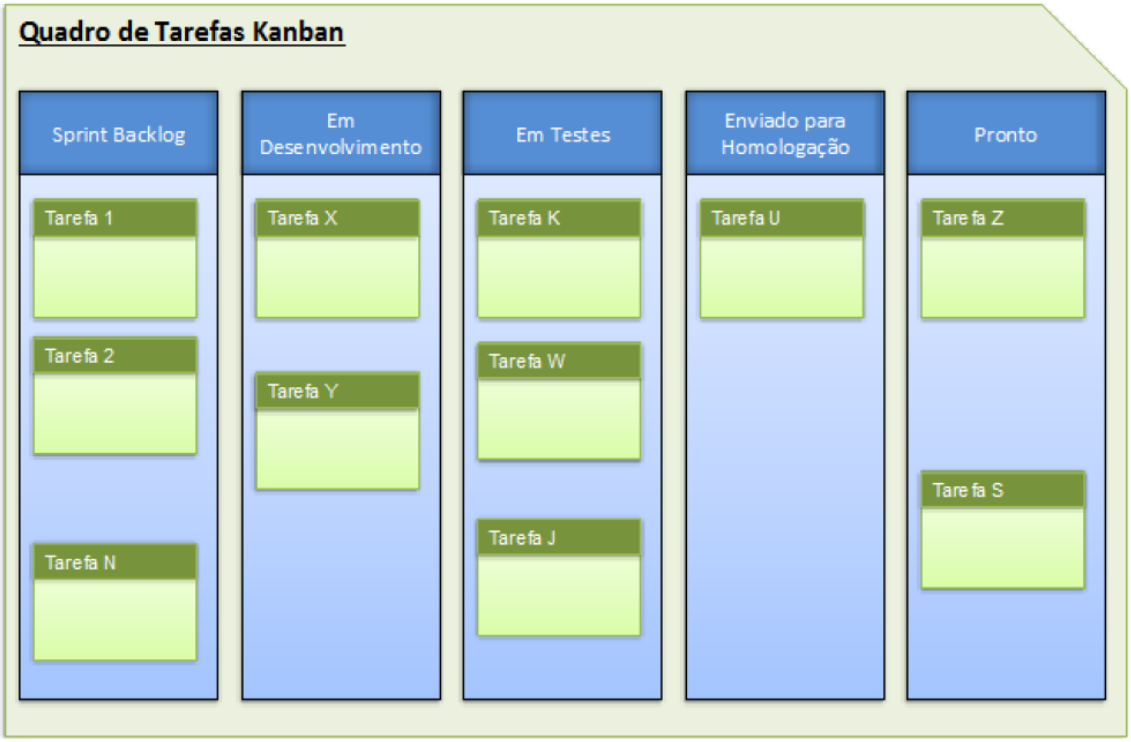

Figura 7 – Quadro de Tarefas Kanban

3.11.3. Reuniões Diárias

3.11.3.1 Além das reuniões previstas no PDA, tais como Planejamento, Refinamento e Retrospectiva, as equipes devem obrigatoriamente fazer reuniões diárias de 15 minutos,

com a participação do Gerente de Sistema e do Scrum Master, de forma que sejam apresentadas rapidamente e **por cada integrante** as metas do dia, problemas a serem enfrentados e pendências previstas.

3.11.3.2 Esta deve ser uma reunião transparente, onde todos os riscos são mapeados, e onde todos os integrantes apresentam sua visão diária do projeto.

3.11.4. Gráfico de Burndown

3.11.4.1 Além do quadro Kanban, um gráfico de produtividade deve ser mantido visível para a equipe de desenvolvimento, exibindo, ao longo do tempo, a relação entre o trabalho planejado e efetivamente realizado. Na figura 8 é possível visualizar um exemplo de gráfico de Burndown. Este gráfico pode ser feito com diferentes índices que representem quantidade de trabalho, como por exemplo, o número de histórias restantes existentes. À medida que o time ganha maturidade, índices mais exatos podem ser utilizados, como horas ou Pontos de Função previstos.

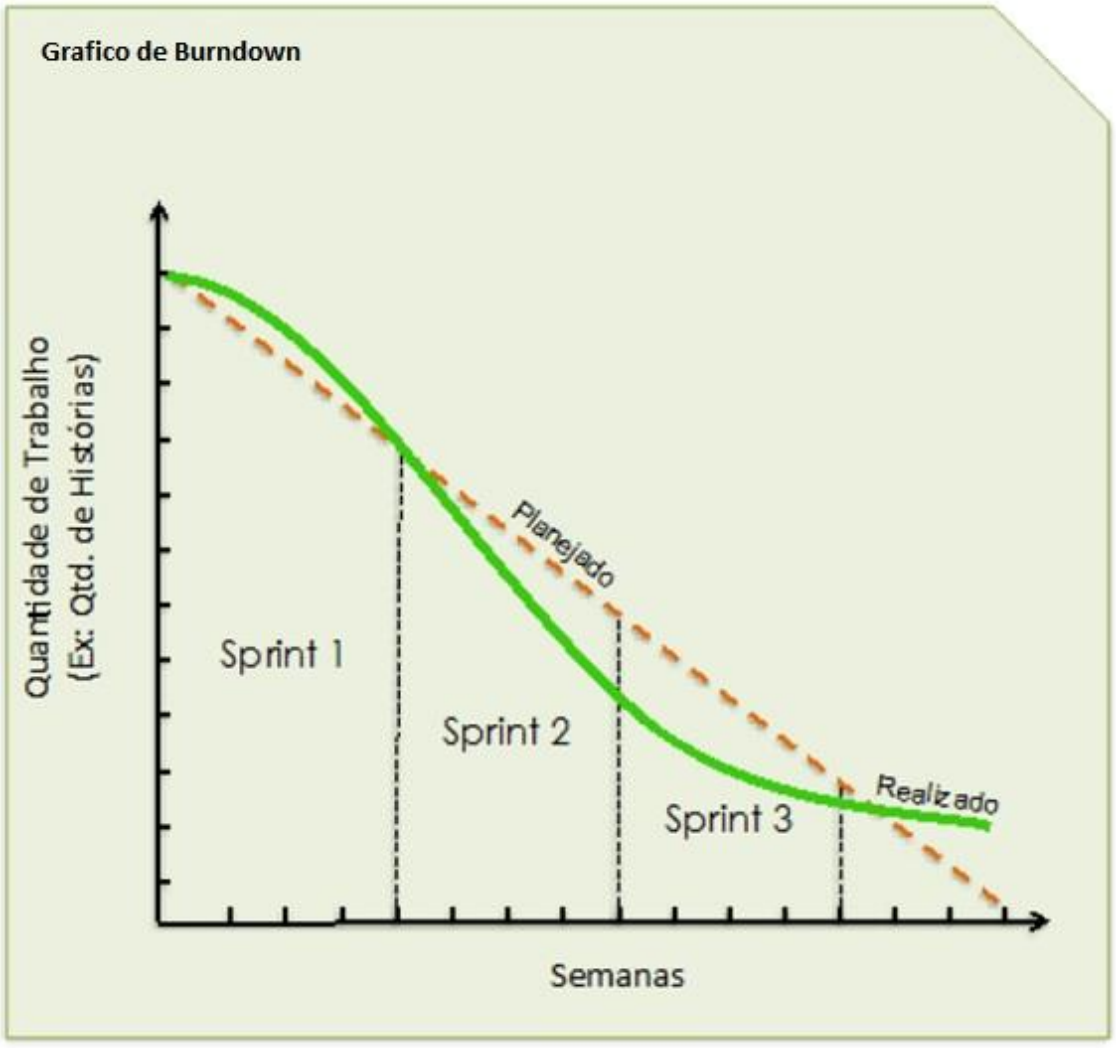

Figura 8 – Gráfico de Burndown

3.12. Paralelismo entre as Sprints

3.12.1. O modelo ágil permite que alguns passos entre as Sprints subsequentes sejam paralelizados. Este paralelismo depende da maturidade das equipes, do projeto e da disponibilidade do PO, dentre outros fatores. Nesta metodologia, por padrão, será adotado o paralelismo conforme Figura 9.

3.12.2. Neste modelo, **a fase de homologação acontece em paralelo à primeira semana da fase de Discovery da próxima Sprint**. Na referida figura, as semanas estão representadas em raias verticais, e as fases coloridas representam etapas de Sprint, conforme legenda.

3.12.3. Tal paralelismo implica, naturalmente, que quaisquer alterações negociais decorrentes desta fase de homologação (percepção de mudança de negócio, novas regras relacionadas às funcionalidades implementadas, etc.) podem ser incorporadas diretamente na Sprint em andamento.

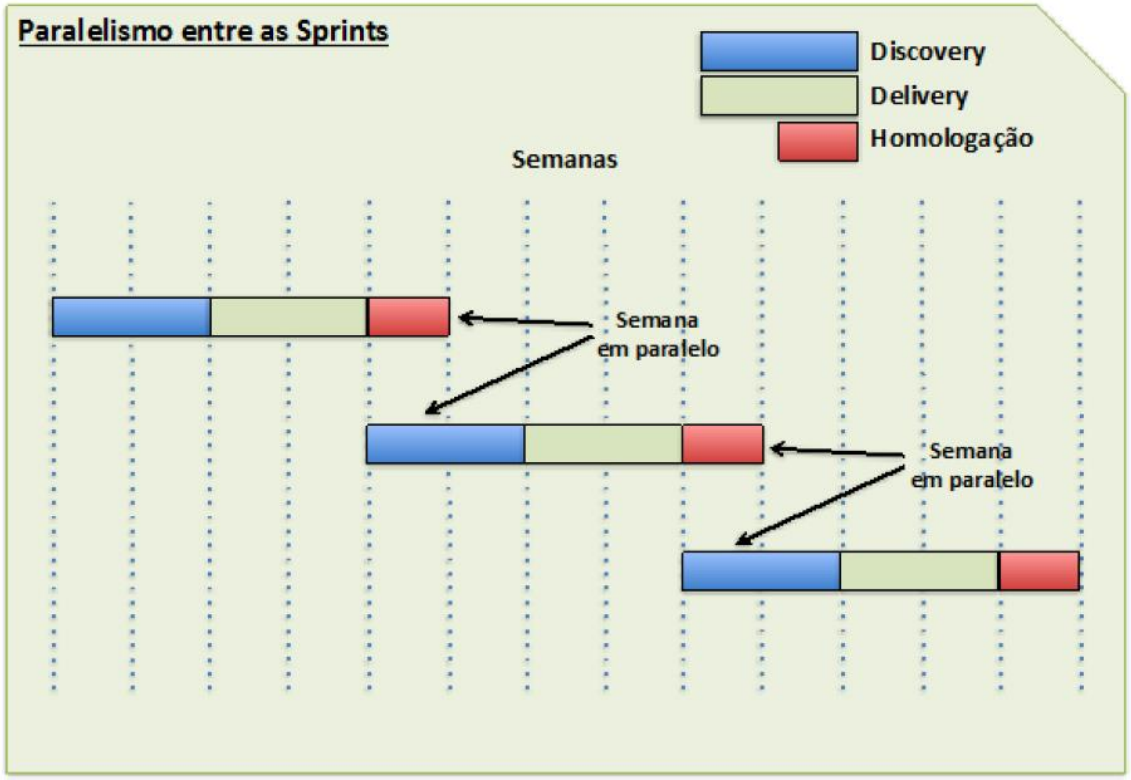

Figura 9 – Paralelismo entre as Sprints

3.13. Artefatos

3.13.1. Embora o Processo de Desenvolvimento Ágil preconize objetividade e código pronto, há uma série de artefatos a serem produzidos no processo, além do produto em si. Tais artefatos podem ser expressos em formato de documento, cujo template é fornecido, ou através de registro na ferramenta de projeto de software em uso no Órgão Contratante.

3.13.2. Ainda existem, além dos artefatos gerais, artefatos específicos para desenvolvimento de componentes e webservices, e por fim os guias operacionais, que são padrões tecnológicos e metodológicos a serem adotados pela contratada.

3.13.3. Artefatos Gerais de Desenvolvimento

Legenda: I – Iniciação **Di** – Discovery **De** – Delivery

GO – Guia Operacional **TP -** Template
WS – WebService **BI** – Business Intelligence

LISTA DE ARTEFATOS	**DESCRIÇÃO**	**NOME DOS TEMPLATES**	**I**	**Di**	**De**	**Obrigatório**
Documento de Visão	Descreve uma visão geral do projeto, seus produtos e suas características	TP - Sigla Projeto - Visão	X			X
Backlog do Produto	Lista das funcionalidades a serem desenvolvidas para o produto, em ordem decrescente de prioridade.	TP - Sigla do Projeto- Produto - Backlog	X	X	X	X
História de Usuário	Especifica uma necessidade do Product Owner e seus critérios de aceitação, além de esboços de tela (quando necessário, pode ser protótipo navegável).	TP - Sigla do Projeto – Funcionalidade – História de Usuário		X		X
Regras de Negócio	Especifica as regras de negócio que farão parte das Histórias de Usuário	TP - Sigla Projeto – Funcionalidade - Regras de Negócios	X	X		X

Protótipo	Apresenta uma demonstração estática ou funcional de solução	TP - Sigla Projeto - Funcionalidade - Protótipo				
Croqui	Esboço de desenho, planta ou projeto	TP - Sigla do Projeto – Funcionalidade - Croqui	X			X
MVP	Fundamentação detalhada escrita, acompanhada ou não de apresentação oral e/ou workshop	TP - Sigla do Projeto - MVP				
Modelo de dados	Projeto de modelo de dados para a aplicação	TP - Sigla Projeto – Modelagem de Dados		X	X	X
Manual de Implantação	Recursos de instalação, hardware, software	TP - Sigla Projeto - Manual de Implantação			X	X
Manual do Usuário	Descreve as funcionalidades do sistema para ajuda ao usuário.	TP - Sigla Projeto - Manual de Usuário			X	X
Relatório de Testes	Artefato gerado por ferramenta que execute os testes unitários e testes funcionais	Não se aplica			X	X
Relatório de Não Conformidade	Relatar as não-conformidades detectadas.	TP - Sigla Projeto – Funcionalidade - Relatório não Conformidade			X	
Especificação de Requisitos	Documento de especificação de requisitos, regras e indicadores	TP - Sigla Projeto – Funcionalidade - Especificação de Requisitos			X	X

3.13.4. Artefatos para Desenvolvimento de Business Intelligence

Especificação de Consulta e Relatórios	Especificação da consulta ou relatório	BI - Sigla Projeto - Especificação de Consulta e Relatórios			X	
Manual de Mapeamento de Origem Destino	Documento de mapeamento de origem destino	BI - Sigla Projeto – Funcionalidade -			X	X

		Mapeamento de Origem Destino				
Manual de Arquitetura de Solução de BI	Documento de arquitetura da solução de BI	BI - Sigla Projeto - Manual de Arquitetura BI			X	X
Documento de Projeto Físico do banco de dados	Projeto físico do banco de dados criado	BI - Sigla Projeto - Projeto Físico banco de dados			X	
Documento de Especificação do ETL	Especificação do ETL (fato ou dimensão)	BI - Sigla Projeto - Especificação do ETL			X	
Relatório de Rotinas e/ou Mapas de ETL	Rotina e/ou mapas de ETL de fatos ou dimensões	BI - Sigla Projeto - Relatório de Rotinas e/ou Mapas de ETL			X	
Documento de Banco de Dados	Banco de dados (Datamart ou Datawarehouse) atualizado	BI - Sigla Projeto - Documento de Banco de Dados			X	X
Documento de Especificação de Consultas	Documento de especificação de consultas/relatório/painel OLAP e/ou camada de apresentação	BI - Sigla Projeto - Documento de Especificação de Consultas			X	X
Documento de Consulta/Relatório/ Painel construído	Consulta/Relatório/ Painel construído	BI - Sigla Projeto - Documento de Consulta/Relatório/ Painel construído			X	X
Documento Painel ou planilha ou relatório	Painel ou planilha ou relatório	BI - Sigla Projeto - Documento Painel ou planilha ou relatório			X	
Manual do Produto customizado e documentado	Produto customizado e documentado	BI - Sigla Projeto - Manual do Produto customizado e documentado			X	X
Relatório de procedimentos realizados	Relatório de procedimentos realizados	BI -Sigla Projeto - Relatório de procedimentos realizados			X	
Manual do produto	Manual do produto entregue	BI - Sigla Projeto - Manual do produto			X	X

Documento de apoio às soluções de Business Intelligence e Datawarehouse	Documento de apoio às soluções de Business Intelligence e Datawarehouse	BI - Sigla Projeto - Documento de apoio às soluções de Business Intelligence e Datawarehouse			X	

3.13.5. Artefatos para Desenvolvimento de Componentes e WebServices

LISTA DE ARTEFATOS	**DESCRIÇÃO**	**NOME DOS TEMPLATES**	**OBRIGATÓRIO**
Especificação Técnica de Componente	Específicas linguagens de programação, componentes, logs e tratamento de erro.	WS - Sigla Projeto - Especificação Técnica Componente	X
Especificação Técnica para Consumo WebService	Especifica a arquitetura para consumo WebService	WS - Sigla Projeto - Especificação Técnica para Consumo Webservice	X
Especificação Técnica Webservice	Especifica a representação da arquitetura WebService	WS - Sigla Projeto - Especificação Técnica Webservice	X

3.13.6. Padrões Tecnológicos – Guias Operacionais

LISTA DE ARTEFATOS	**DESCRIÇÃO**	**NOME DOS TEMPLATES**	**OBRIGATÓRIO**
Manual de Padrão de Telas	Define um padrão de telas para os sistemas do ÓRGÃO CONTRATANTE	GO - Camada de Apresentação de Telas	X
Arquitetura de referência	Documento que define a arquitetura base para os sistemas	GO - Sigla Projeto Arquitetura de Referência Java	X

Arquitetura de Referência de Componentes Corporativos	Descrever a arquitetura padrão utilizada no desenvolvimento de componentes corporativos	GO Arquitetura de Referência Componentes Corporativos	X
Padrões de Nomenclaturas para Banco de Dados	Padronizar a Nomenclatura para Banco de Dados	GO – Nomenclatura para Banco de Dados ANM	
Orientações sobre a confecção de modelos de dados	Orientar a criação de modelos de dados	GO - Elaboração de Modelos de Dados	
Dicionário de dados	Dicionário de dados do sistema	GO - Dicionário de Dados	X
Padrão de codificação	Padrão de codificação	GO - Codificação <Nome Tecnologia>	X
Documentação de Sistemas Legados	Adoção de uma documentação mínima para os sistemas legados garantirá maior produtividade, qualidade e facilidade nos processos de manutenção	GO - Documentação de Sistemas legados	X
Versionamento de Código	Descrever as atividades do Processo de Controle de Versionamento de Programas a ser adotado no Processo de Desenvolvimento de Software utilizado no Órgão Contratante, de forma que a evolução dos sistemas ocorra de modo controlado.	GO – Versionamento de Código	X

4. PROCESSO DE SUSTENTAÇÃO DE SISTEMAS (PSS)

4.1. Trata-se do processo que envolve as atividades de sustentação de sistemas, tratando-se de manutenção continuada e estendendo-se desde sua implantação até o momento em que for substituído ou descontinuado.

4.2. Atividades de Sustentação

4.3. Dentro deste processo, estão englobadas uma série de atividades distintas, detalhadas a seguir.

4.3.1. Manutenção corretiva

4.3.1.1 Consiste na eliminação de comportamentos do software que divirjam de suas especificações ou que provoquem a interrupção inesperada de seu funcionamento.

4.3.2. Manutenção adaptativa tecnológica

4.3.2.1 Consiste na alteração do sistema para adaptá-lo às mudanças do ambiente computacional onde foi desenvolvido ou onde é executado, considerados aí os componentes tecnológicos passíveis de adaptação: Sistema Gerenciador de Bancos de Dados, Servidor de Aplicações, bibliotecas e/ou *frameworks* utilizados e as evoluções da própria linguagem computacional utilizada.

4.3.3. Manutenção cosmética localizada

4.3.3.1 Consiste em alteração de interface de usuário que não implique alteração das regras de negócio do Caso de Uso e que seja realizada de forma localizada, isto é, pela intervenção em um único arquivo ou em um pequeno conjunto de arquivos. Tal manutenção pode ser exemplificada da forma que se segue:

- Fontes de letra, cores, logotipos, mudanças de botões, alteração na posição de campos e texto na tela;

- Mudanças de texto em mensagens do sistema, título de um relatório ou *labels* de uma tela de consulta;

- Mudanças de texto estático em e-mail enviado pelo sistema.

4.3.4. Apurações especiais

4.3.4.1. Consiste na preparação de roteiros de execução em linguagem SQL ou outra adequada ao caso, destinados às extrações de dados não cobertas pelos relatórios do sistema, à correção de inconsistências nos dados mantidos pelo sistema e não realizáveis por meio das interfaces de usuário disponíveis (ou cujo volume inviabilize a sua execução de forma manual), ou à inserção de dados não automatizada no sistema.

4.3.5. Atendimento

4.3.5.1 Trata-se de prestação de esclarecimentos, ao ÓRGÃO CONTRATANTE, quanto à forma como foram implementados os requisitos do sistema, aos procedimentos requeridos ao seu correto funcionamento ou aos dados mantidos por ele. Em adição, também se enquadra nesta atividade o apoio à identificação e isolamento de falhas e problemas na execução do software.

4.3.6. Rotinas operacionais

4.3.6.1 Consiste na execução de quaisquer procedimentos operacionais rotineiramente requeridos pelo sistema em função de suas regras de negócio ou forma de construção.

4.4. Tratamento de Demandas de Sustentação

4.4.1. Todas as atividades descritas anteriormente seguem fluxos a serem tratados pela contratada, que têm como ponto de partida o registro do incidente ou solicitação no Sistema de Gestão de Demandas em uso no Órgão Contratante, que a partir de então deverá ser tratado pela contratada de acordo com as cláusulas de nível de serviço (que não são escopo desta metodologia) e tipo da demanda. Uma visão geral deste tratamento pode ser vista na Figura 10.

4.4.2. Manutenções Corretivas, Adaptativas e Cosméticas

4.4.3. Após a execução do serviço, a contratada deverá tomar as seguintes providências:

- Providenciar que o código com a mudança solicitada (correção, adaptação ou manutenção cosmética) seja enviado para geração de build em homologação, ou eventualmente diretamente em produção (dependendo da urgência ou tipo demanda, tratado a cada caso);
- Atualizar a versão do sistema conforme o guia de gestão de release do ÓRGÃO
  CONTRATANTE;
- Registrar tudo o que for realizado no sistema de gestão de demandas, relacionando, se possível, com casos semelhantes já conhecidos.

4.4.3.1 Artefatos Resultantes

4.4.3.2.1 O artefato resultante:

- Evidência de Teste.

4.4.4. Apurações Especiais

4.4.4.1 Após a execução do serviço, a contratada deverá tomar as seguintes providências:

- Providenciar que a intervenção em base de dados com a apuração solicitada (correção de registros, inclusão de dados em CODE TABLE, relatórios direto de base de dados, dentre outras) seja enviada para

execução em base de dados de homologação, ou eventualmente, diretamente em base de produção (dependendo da urgência ou tipo demanda, tratado a cada caso);

- Registrar tudo o que for realizado no Sistema de Gestão de Demandas, relacionando, se possível, com casos semelhantes já conhecidos.

4.4.5. Atendimento e Rotinas Operacionais

4.4.5.1 Após a execução do serviço, a contratada deverá tomar as seguintes providências:

- Anexar evidências, caso possível, do atendimento realizado ou rotinas executadas;
- Registrar tudo, inclusive entregáveis, que for realizado no Sistema de Gestão de Demandas,
- relacionando, se possível, com casos semelhantes já conhecidos.

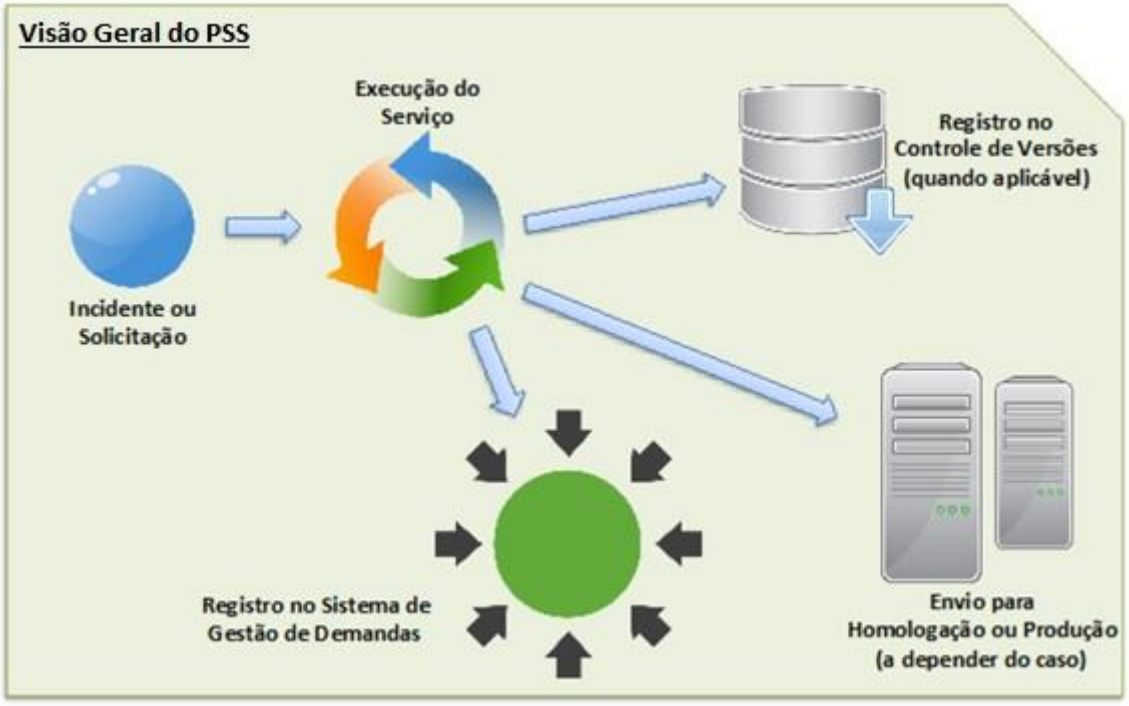

Figura 10 – Visão Geral do Processo de Sustentação de Sistemas

5. PROCESSO DE EVOLUÇÃO DE PEQUENO PORTE (PEP)

5.1. Trata-se de processo para lidar com situações em que as evoluções necessárias e/ou desejadas são de pequeno porte e não são elencadas como projeto. Cabe ressaltar que evoluções categorizadas como projeto são tratadas pelo PDA (Processo de Desenvolvimento Ágil).

5.2. Este processo, cuja visão geral pode ser observada na Figura 11, é composto de duas partes:

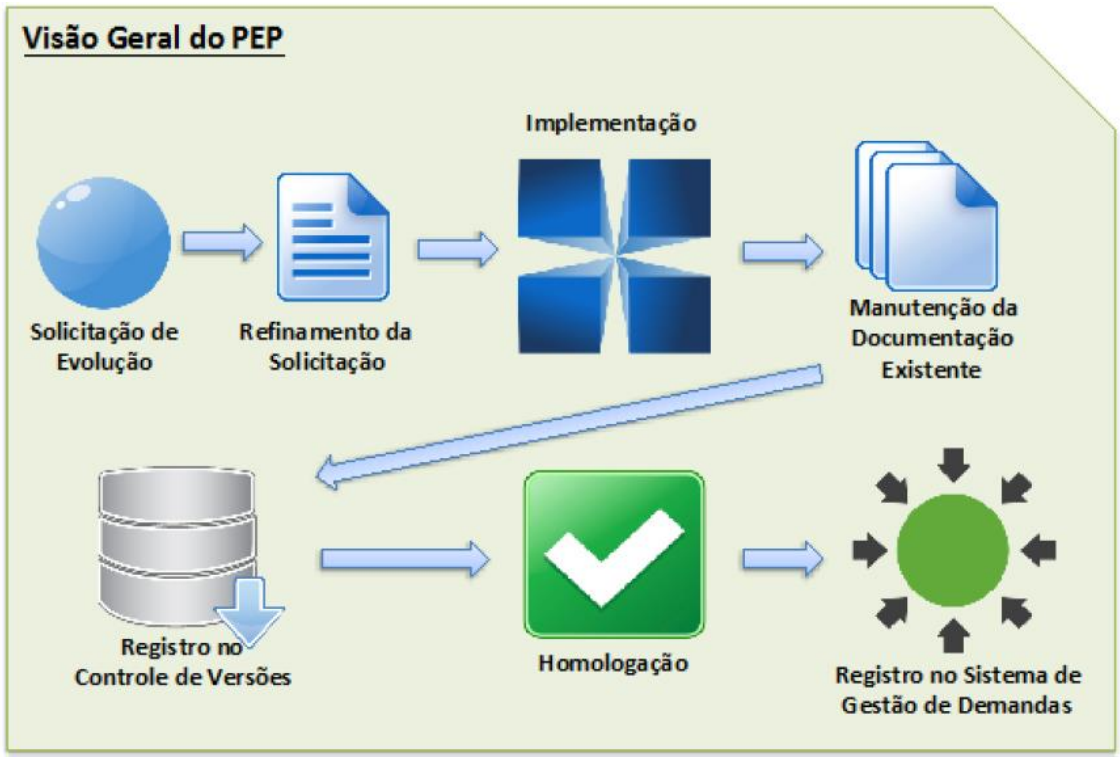

Figura 11 – Visão Geral do Processo de Evolução de Pequeno Porte

5.2.1. Abertura da solicitação

5.2.1.1 A necessidade de evolução deve ser relatada por um usuário demandante no Sistema de Gestão de Demandas, que tipicamente pode ser um Gestor ou Gerente de Sistema. Devido ao pequeno porte envolvido, não há formato fixo para tal relato, que é livre.

5.2.2. Refinamento da solicitação

5.2.2.1 Após o recebimento da solicitação, deve haver uma reunião para refinar o entendimento sobre seu teor. Nessa reunião, quaisquer detalhes devem ser coletados e registrados no caso aberto no Sistema de Gestão de Demandas.

5.2.3. Implementação

5.2.3.1 Seguinte ao refinamento da solicitação, deve haver então a implementação conforme solicitado. Neste passo, é importante seguir boas práticas tais como desenvolver ou evoluir testes unitários correspondentes, e seguir as premissas arquiteturais existentes no Órgão Contratante.

5.2.3.2 Ao final da implementação, devem ser executados os testes unitários correspondentes, cuja evidência deve ser anexada ao Sistema de Gestão de Demandas.

5.2.4. Manutenção da Documentação Existente

5.2.4.1 Em caso de haver documentação existente para o projeto em questão, esta deverá ser atualizada conforme as novas regras implementadas na evolução em questão.

5.2.5. Registro no Controle de Versões

5.2.5.1 O produto resultante da solicitação deve ser submetido ao Controle de Versões, e a ele atribuído versão conforme o Guia de Gestão de Release, anexo a esta MGDS.

5.2.6. Homologação

5.2.6.1 Deve ser gerado um executável (build) da versão correspondente à evolução em questão, e submetido ao ambiente de homologação para que o usuário demandante possa verificar se o artefato construído está de acordo com a solicitação. Todos entregáveis de software e documentação deverão estar versionados nos repositórios da CONTRATADA, inclusive o TAD (Termo de Aceite da Demanda).

5.2.7. Registro no Sistema de Gestão de Demandas

5.2.7.1 Todo o histórico e demais informações da solicitação devem estar devidamente registradas no Sistema de Gestão de Demandas após a homologação por parte do demandante.

5.2.8. Artefatos Resultantes

5.2.8.1. Os artefatos resultantes são:

- Análise Técnica da Demanda;
- Evidências de Teste.

6. COMO MEDIR PERFORMANCE NO SCRUM

6.1. As equipes SCRUM devem medir e acompanhar a sua performance. O SCRUM fornece uma estrutura empírica e leve para abordar problemas complexos. A medição da performance fornece os meios para avaliar o progresso e facilitar a identificação e resolução de questões de desenvolvimento.

6.2. A coleta e o exame superficial de dados são inadequados para adotar totalmente o desenvolvimento ágil e produzir melhores resultados com SCRUM. As equipes de desenvolvimento devem coletar, analisar e agir de acordo com as informações coletadas para melhorar substancialmente seu desempenho.

6.3. A empresa de consultoria Gartner (https://www.gartner.com/) contém um estudo detalhado das três principais formas de medir a performance no SCRUM. É um estudo que foca em analisar dados do SCRUM para identificar problemas de desenvolvimento e resolvê-los. Este estudo permite:

- Compreender padrões gráficos;

- Analisar os desvios de padrões;
- Identificar e implementar ações corretivas recomendadas.

6.4. O SCRUM é uma metodologia de fácil implementação e que permite o alcance de bons resultados desde o começo de sua utilização. Mas para entregar valores confiáveis com regularidade, as equipes precisam entender melhor as unidades de medidas a serem utilizadas.

6.5. Ao analisar o progresso no SCRUM, as equipes devem usar três gráficos de medição de desempenho em desenvolvimento ágil, cada um com propósito e público distintos.

6.6. A abordagem prevê que a equipe SCRUM colete os dados necessários, analise os padrões comumente vistos e use as orientações fornecidas para melhorar o desempenho.

6.7. As três ferramentas que permitem aumentar a capacidade da equipe são:

- Sprint Burndown Chart: este gráfico é usado pela equipe de desenvolvimento para analisar o progresso da sprint. A equipe de desenvolvimento atualiza o gráfico a cada SCRUM daily para acompanhar o trabalho nos dias restantes da sprint. A equipe analisa os dados coletados para melhorar sua proficiência técnica e aumentar sua capacidade de entregar o que está previsto na sprint.

- Velocity Chart: este relatório é usado pela equipe SCRUM para avaliar a quantidade de trabalho que pode ser entregue em cada sprint. O gráfico mostra a quantidade de trabalho prevista e a ser entregue em cada sprint sucessivamente. O Product Owner utiliza essas informações ao final de cada sprint para se planejar, enquanto a equipe de desenvolvimento as utiliza para melhorar a entrega de valor.

- Release Burndown Chart: este gráfico permite visualizar a quantidade de trabalho necessário para concluir a sprint. É utilizado na liberação pelo Product Owner para analisar o progresso com a equipe de desenvolvimento e as partes interessadas no sprint review.

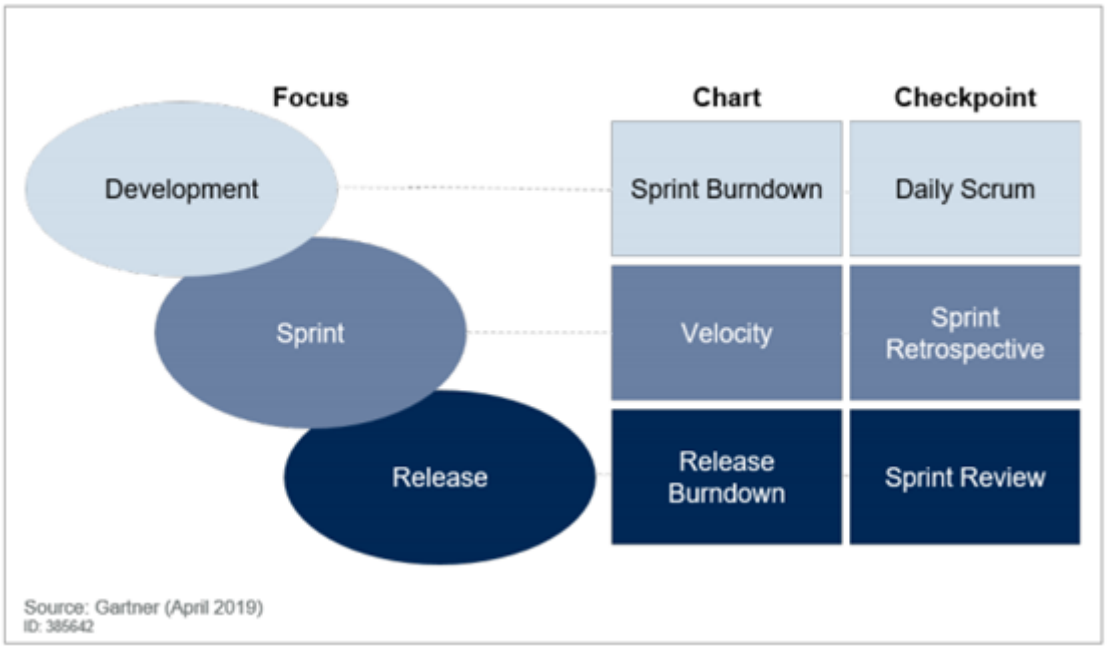

Figura 12 – Gráficos de Performance

6.8. O Quadro de Orientação

6.8.1. As equipes ágeis devem usar critérios para avaliar o progresso, identificar impedimentos e resolver problemas.

6.8.2. Processos empíricos coletam informações observando e experimentando, enquanto processos adaptativos alteram práticas de desenvolvimento baseadas em dados. A combinação dessas práticas possibilita às equipes ágeis coletar informações, analisar tendências e apoiar a rápida tomada de decisões, para otimizar o desempenho e produzir o melhor resultado possível.

6.8.3. Para a empresa de consultoria Gartner (https://www.gartner.com/) indicadores são a principal fonte de feedback, e o feedback é o objetivo principal das metodologias ágeis e de DevOps.

6.8.4. Equipes de desenvolvimento que coletam e analisam indicadores aproveitam sucessos, fracassos e oportunidades para o aprimoramento da equipe.

6.8.5. Os três principais instrumentos para acompanhar o progresso e o desempenho das equipes SCRUM são o gráfico Sprint Burndown, gráfico Velocity e o gráfico Release Burndown, conforme mostrado na figura abaixo.

Figura 13 – Gráficos de desempenho SCRUM

6.9. Sendo auto organizada e responsável, a equipe SCRUM automatiza a coleta de informações, avalia quantitativamente e impulsiona mudanças intencionais em seu processo a cada sprint. Equipes de desenvolvimento que usam índices de performance continuamente aperfeiçoam-se de forma iterativa e incremental.

6.10. Os gráficos Sprint Burndown, Velocity e Release Burndown ilustram os três pilares do SCRUM:

- Transparência: tornando aspectos significativos do processo visíveis para a equipe SCRUM. Uma organização mais ampla constrói um entendimento compartilhado das práticas e do progresso do desenvolvimento. A utilização de gráficos fomenta e facilita a análise crítica sobre o processo.

- Inspeção: as equipes SCRUM usam indicadores para melhorar continuamente sua capacidade de agregar valor de forma iterativa e incremental para seus clientes. A estrutura SCRUM fornece pontos de inspeção fixos para permitir que a equipe se concentre na produção de valor em sprints curtos.

- Adaptação: são feitos ajustes no processo com base em dados quantificáveis coletados automaticamente e analisados pela equipe SCRUM. Os curtos ciclos de aprendizado do SCRUM permitem que a equipe execute experimentos rápidos e melhore sua capacidade e habilidade, oferecendo melhor resultado.

6.11. Ao utilizar indicadores para melhorar o processo SCRUM, as equipes de desenvolvimento obtêm provas quantificáveis de que melhorias estão acontecendo, em vez de depender de suposições. Ao incorporar esses três gráficos no ciclo de desenvolvimento e analisá-los regularmente reduzimos os riscos e possibilitamos que o planejamento seja mais preciso e adaptável.

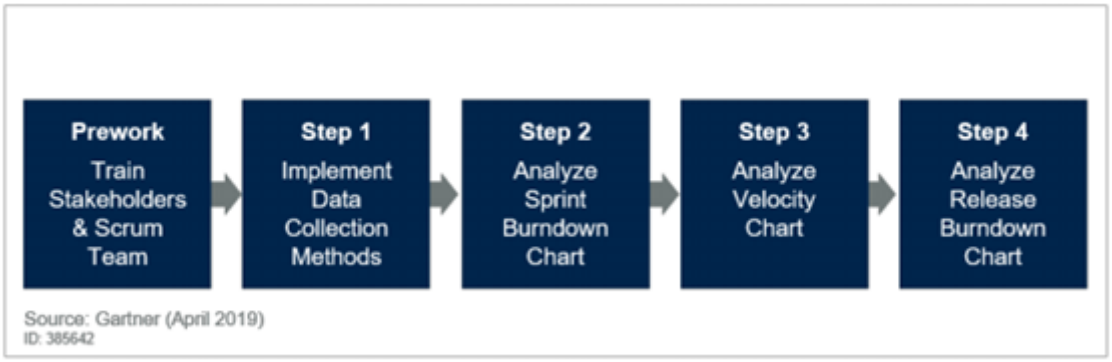

Figura 14 – Etapas de controle efetivo do SCRUM