Livro Eletrônico

www.igepp.com.br

Prof. Alvaro Costa Jr.
e Rodrigo Silva

DIRETO AO PONTO

[Concurso Público Nacional Unificado 2024]

Gestão de projetos: Metodologias ágeis

IGEPP
ONLINE

1. CONTEÚDO

3. Gestão de projetos: **Metodologias ágeis.**

2. CONCEITOS ESSENCIAIS

As metodologias ágeis foram inicialmente desenvolvidas para o desenvolvimento de software, mas ao longo do tempo, elas expandiram seu alcance e começaram a ser aplicadas em diversas áreas de gestão, tanto no setor público quanto no privado. A agilidade não se limita mais apenas ao desenvolvimento de software, mas também é aplicada em projetos e processos em várias outras áreas, como marketing, recursos humanos, operações e gestão de projetos em geral.

As metodologias ágeis foram, de certo modo, uma reação ao **método cascata**, também conhecido como modelo em cascata ou modelo sequencial linear. É uma abordagem de desenvolvimento de software que segue uma sequência linear e ordenada de etapas. Nesse modelo, cada fase do projeto deve ser concluída antes que a próxima fase possa começar, sem possibilidade de retorno às fases anteriores uma vez que estas foram finalizadas. As principais fases do método cascata geralmente incluem:

Requisitos: Nesta fase inicial, os requisitos do sistema são levantados e documentados em detalhes. Isso envolve a compreensão das necessidades do cliente e a definição clara dos recursos e funcionalidades desejados no software.

Análise: Os requisitos levantados são analisados em detalhes para garantir sua viabilidade técnica e operacional. Esta fase foca na compreensão dos requisitos do sistema e na identificação de quaisquer ambiguidades ou lacunas.

Projeto: Com base nos requisitos e na análise realizada, é desenvolvido um plano detalhado de design do sistema. Isso inclui a definição da arquitetura do software, a estruturação dos componentes e a especificação das interfaces entre eles.

Implementação: Na fase de implementação, o código-fonte do software é desenvolvido com base no projeto elaborado. Esta etapa envolve a escrita do código, a programação das funcionalidades e a integração dos diferentes componentes do sistema.

Testes: Uma vez que a implementação está completa, o software é submetido a testes rigorosos para garantir que ele atenda aos requisitos especificados e que funcione conforme o esperado. Os testes podem incluir testes unitários, testes de integração e testes de aceitação pelo usuário.

Implantação: Após a conclusão bem-sucedida dos testes, o software é implantado e disponibilizado para uso pelos usuários finais. Isso pode envolver a instalação do software em servidores, a distribuição para usuários finais ou a disponibilização em lojas de aplicativos.

Manutenção: A última fase do método cascata envolve a manutenção contínua do software, incluindo correções de bugs, atualizações de segurança e aprimoramentos de funcionalidades conforme necessário ao longo do tempo.

Embora o método cascata tenha sido amplamente utilizado no passado, ele tem algumas limitações, como dificuldade de adaptação a mudanças de requisitos, falta de flexibilidade e alto risco de falhas devido à abordagem sequencial e irreversível. Por esses motivos, muitas organizações têm adotado abordagens mais ágeis, como Scrum e Kanban, que oferecem maior flexibilidade e adaptabilidade aos projetos de desenvolvimento de software.

Essa expansão ocorreu porque os princípios e práticas ágeis demonstraram sua eficácia além do desenvolvimento de software. Muitas das características das metodologias ágeis, como **colaboração**, **entrega incremental**, **adaptação a mudanças** e **foco no valor entregue ao cliente**, são aplicáveis em uma ampla gama de contextos empresariais e governamentais.

Marcos Roque da Rosa e Eliane Nascimento Pereira (2021)[1] apontaram que a adoção de abordagens ágeis em projetos do setor público é considerada um desafio significativo para as instituições, pois exige a capacitação de profissionais em metodologias ágeis, a adaptação às particularidades da burocracia, a exploração de métodos de trabalho mais flexíveis e a promoção de mudanças na cultura organizacional. **A implementação do ágil pode contribuir para a melhoria da forma como a administração pública presta serviços e para aumentar a satisfação dos cidadãos.**

Quatro são os valores percebidos pela filosofia ágil:

[1] **Metodologias ágeis no contexto da administração pública**: análise de estudos de caso de implementação ágil.Revista do Serviço Público (RSP), Brasília 72 (2) 479-497 abr/jun 2021.

- **Indivíduos e interações**, em vez de processos e ferramentas
- **Software funcional**, em vez de documentação abrangente
- **Colaboração do cliente,** em vez de negociação de contratos
- **Responder mudanças**, em vez de seguir um plano

Com base nesses valores, a *Agile Alliance* estabelece 12 princípios de agilidade:

1. Garantir a satisfação do consumidor entregando rapidamente e continuamente softwares funcionais;
2. Softwares funcionais são entregues frequentemente (semanas, ao invés de meses);
3. Softwares funcionais são a principal medida de progresso do projeto;
4. Até mesmo mudanças tardias de escopo no projeto são bem-vindas. Os processos ágeis devem ser uma vantagem competitiva;
5. Cooperação constante entre pessoas que entendem do 'negócio' e desenvolvedores;
6. Projetos surgem através de indivíduos motivados, entre os quais existe relação de confiança.
7. Conversa aberta é a melhor maneira de transmitir informações para dentro e fora da equipe;
8. Atenção contínua para a excelência técnica;
9. Os processos ágeis promovem desenvolvimento sustentável. Desenvolvedores e usuários devem pode manter um ritmo constante de trabalho indefinidamente;
10. Simplicidade é essencial;
11. As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto organizam;
12. Colaboração com clientes mais do que negociação de contratos;

Seguidos esses princípios, existem algumas metodologias ágeis bastante conhecidas, como o **XP** e o **Scrum**. Vejamos:

3. XP [eXtreme Programming]

XP, oriunda de eXtreme Programming, é uma filosofia de programação que segue algumas práticas genéricas, cujo conhecimento considero válido. São elas:

- **Jogo de Planejamento** (*Planning Game*): O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planejamento. Nela, o cliente identifica prioridades e os desenvolvedores as estimam. O cliente é essencial neste processo e assim ele fica sabendo o que está acontecendo e o que vai acontecer no projeto. Como o escopo é reavaliado semanalmente, o projeto é regido por um contrato de escopo negociável, que difere significativamente das formas tradicionais de contratação de projetos de software. Ao final de cada semana, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção.
- **Pequenas Versões** (*Small Releases*): A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versões chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP.
- **Metáfora** (*Metaphor*): Procura facilitar a comunicação com o cliente, entendendo a realidade dele. O conceito de rápido para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas em tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.
- **Projeto Simples** (*Simple Design*): Simplicidade é um princípio da XP. Projeto simples significa dizer que caso o cliente tenha pedido que na primeira versão apenas o usuário "teste" possa entrar no sistema com a senha "123" e assim ter acesso a todo o sistema, você vai fazer o código exato para que esta funcionalidade seja implementada, sem se preocupar com sistemas de autenticação e restrições de acesso. Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código *simples* e código *fácil*. Nem sempre o código mais fácil de ser desenvolvido levará a solução mais simples por parte de projeto. Esse entendimento é fundamental para o bom andamento do XP. Código fácil deve ser identificado e substituído por código simples.
- **Time Coeso** (*Whole Team*): A equipe de desenvolvimento é formada por pessoas engajadas e de forma multidisclipinar (no sentido de incluir pessoas com cada uma das habilidades necessárias para o projeto).

- **Testes de Aceitação** (*Customer Tests*): São testes construídos pelo cliente e conjunto de analistas e testadores, para aceitar um determinado requisito do sistema.
- **Ritmo Sustentável** (*Sustainable Pace*): Trabalhar com qualidade, buscando ter ritmo de trabalho saudável (40 horas/semana, 8 horas/dia), sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto. Outra prática que se verifica neste processo é a prática de trabalho energizado, onde se busca trabalho motivado sempre. Para isto o ambiente de trabalho e a motivação da equipe devem estar sempre em harmonia.
- **Reuniões em pé** (*Stand-up Meeting*): Reuniões em pé para não se perder o foco nos assuntos, produzindo reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe.
- **Posse Coletiva** (*Collective Ownership*): O código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema.
- **Programação em Pares** (*Pair Programming*): é a programação em par/dupla num único computador. Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de defeitos. Com isto busca-se sempre a evolução da equipe, melhorando a qualidade do código fonte gerado.
- **Padrões de Codificação** (*Coding Standards*): A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros.
- **Desenvolvimento Orientado a Testes** (*Test Driven Development*): Primeiro crie os testes unitários (*unit tests*) e depois crie o código para que os testes funcionem. Esta abordagem é complexa no início, pois vai contra o processo de desenvolvimento de muitos anos. Só que os testes unitários são essenciais para que a qualidade do projeto seja mantida.
- **Refatoração** (*Refactoring*): É um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refabricar melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte;
- **Integração Contínua** (*Continuous Integration*): Sempre que produzir uma nova funcionalidade, nunca esperar uma semana para integrar à versão atual

do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte. Integrar de forma contínua permite saber o status real da programação.

4. SCRUM

Vamos examinar o **Scrum**, uma metodologia ágil amplamente empregada para a gestão de projetos, comumente associada ao desenvolvimento de software, mas também aplicável a uma variedade de outros tipos de projetos.

O Scrum enfatiza a colaboração da equipe, flexibilidade e entrega contínua de valor. Discutiremos os papéis essenciais no Scrum, seus princípios fundamentais e os benefícios dessa abordagem**. Scrum** não é um processo nem descreve o que se deve fazer em cada situação. Ele é usado para trabalhos complexos nos quais é difícil predizer tudo o que irá ocorrer.

O **Scrum** é um conjunto de passos que contém grupos de práticas e papéis pré-definidos. Os únicos papéis são:

1. **O Proprietário do Produto, ou Product Owner**, que representa os *stakeholders* e o negócio;
2. **o *Scrum Master***, que mantém os processos (normalmente no lugar de um gerente de projeto);
3. **a Equipe, ou *Team***, um grupo multifuncional com cerca de 7 pessoas e que fazem a análise, projeto, implementação, teste, etc.

Sprint

Um ***sprint*** é a unidade básica de desenvolvimento em Scrum. Sprints tendem a durar entre uma semana e um mês, e são um esforço dentro de uma "caixa de tempo" (ou seja, restrito a uma duração específica) de comprimento constante.

Cada **sprint** é precedido por uma reunião de planejamento, onde as tarefas para o sprint são identificadas e um compromisso estimado para o objetivo do sprint é

definido e seguido por uma reunião de revisão ou de retrospectiva, onde o progresso é revisto e lições para os próximos *sprints* são identificadas.

Durante cada ***sprint***, a equipe cria um incremento de produto potencialmente entregável (por exemplo, software funcional e testado). O conjunto de funcionalidades que entram em um sprint vêm do *backlog* do produto, que é um conjunto de prioridades de requisitos de alto nível do trabalho a ser feito. Quais itens do *backlog* entram para o sprint são determinados durante a reunião de planejamento do sprint. Durante esta reunião, o *Product Owner* informa a equipe dos itens no *backlog* do produto que ele ou ela quer concluídos. A equipe então determina quantos eles podem se comprometer a concluir durante o próximo sprint, e registram isso no *backlog* do sprint. Durante um **sprint**, ninguém está autorizado a alterar o *backlog* do sprint, o que significa que os requisitos são congelados para esse sprint. O desenvolvimento de cada sprint deve terminar na "caixa de tempo" prevista. Se os requisitos não são completados por qualquer motivo, eles são deixados de fora e voltam para o *backlog* do produto. Depois que um *sprint* é completado, a equipe demonstra como usar o software.

O ***Scrum*** permite a criação de equipes auto organizadas, encorajando a cooperação de todos os membros da equipe e a comunicação verbal entre todos os membros e disciplinas da equipe no projeto.

1. Artefatos

Product Backlog

Um *backlog* é uma lista de itens priorizados a serem desenvolvidos para um software. O *Product Backlog* é mantido pelo *Product Owner* e é uma lista de requisitos que tipicamente vêm do cliente. O *Product Backlog* pode ser alterado a qualquer momento pelo *Product Owner* ou por decisão deste.

Sprint backlog

O *Sprint backlog* é uma lista de itens selecionados do *Product backlog* e contém tarefas concretas que serão realizadas durante o próximo *sprint* para implementar tais itens selecionados. O *Sprint Backlog* é uma representação em tempo real do trabalho que o *Development Team* planeja concluir na *sprint* corrente, e ele pertence unicamente ao *Development Team*.

Planejamento de *sprint*

Antes de todo *sprint*, o Product Owner, o Scrum Master e a Equipe decidem no que a equipe irá trabalhar durante o próximo *sprint*. O Product Owner mantém uma lista priorizada de itens de *backlog*, o *backlog* do produto, o que pode ser repriorizado durante o planejamento do *sprint*. A Equipe seleciona itens do topo do *backlog* do produto. Eles selecionam somente o quanto de trabalho eles podem executar para terminar. A Equipe então planeja a arquitetura e o design de como o *backlog* do produto pode ser implementado. Os itens do *backlog* do produto são então destrinchados em tarefas que se tornam o *backlog* do *sprint*.

2. Reuniões

Daily Scrum

Cada dia durante o sprint, uma reunião de status do projeto ocorre. Isso é chamado de "*scrum* diário". Esta reunião tem diretrizes específicas:

- A reunião começa precisamente no horário marcado.
- Todos são bem-vindos, mas apenas "poucos" podem falar.
- O encontro tem duração determinada (*TimeBox*) e dura 15 minutos.
- A reunião deve acontecer no mesmo local e mesma hora todos os dias
- Durante a reunião, cada membro da equipe responde a três perguntas:
- O que você tem feito desde ontem?
- O que você está planejando fazer hoje?
- Você tem algum problema impedindo você de realizar seu objetivo?

É papel do Scrum Master para facilitar a resolução desses impedimentos. Normalmente, isso deve ocorrer fora do contexto do Daily Scrum para que a reunião possa durar menos de 15 minutos.

Reunião de Planejamento da Sprint (*Sprint Planning Meeting*)

No início do ciclo de sprint (a cada 7-30 dias), um *Sprint Planning Meeting* é realizado.

- Selecione o trabalho que está a ser feito.
- Prepare o Sprint Backlog que detalha o tempo que levará para fazer esse trabalho, com toda a equipe.
- Identificar e comunicar o quanto o trabalho é susceptível de ser feito durante o sprint atual.
- Dividida em duas partes:
- Parte 1 (Primeiras quatro horas): Team Product Owner: diálogo para priorizar o Product Backlog.
- Parte 2 (Próximas quatro horas): Team apenas: hash de um plano para a Sprint, resultando na Sprint Backlog.

No final de um ciclo de sprint, são realizadas duas reuniões: a "Sprint Review" e do "Sprint Retrospective".

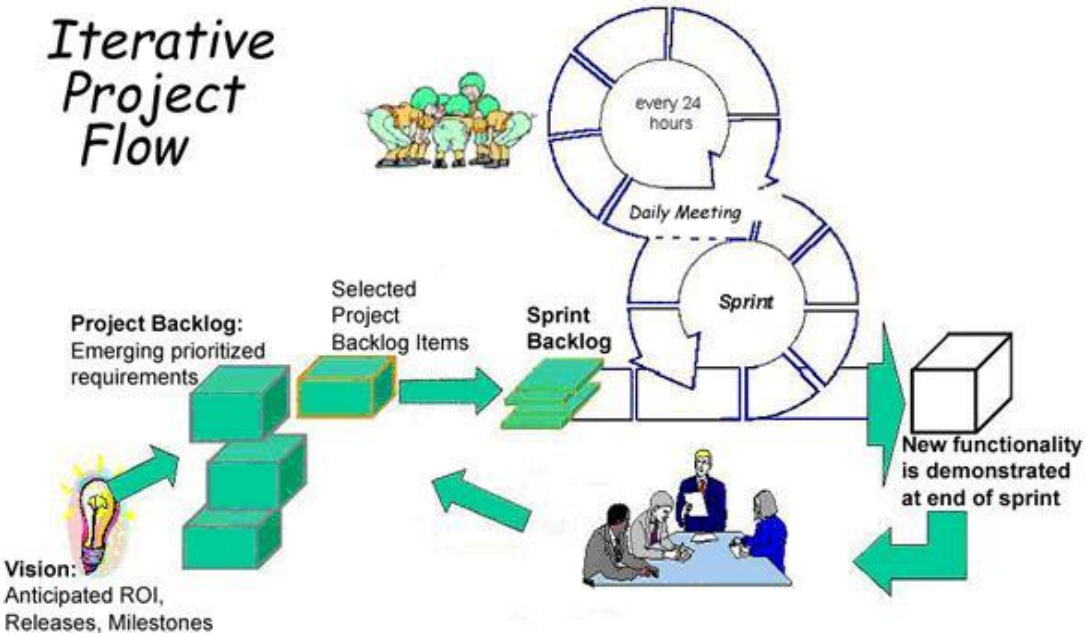
Reunião de Revisão da Sprint (*Sprint Review*)

- Rever o trabalho que foi concluído e não concluído.
- Apresentar o trabalho realizado para os interessados (ou "a demo"). Um trabalho incompleto não pode ser demonstrado.

Retrospectiva da Sprint (*Sprint Retrospective*)

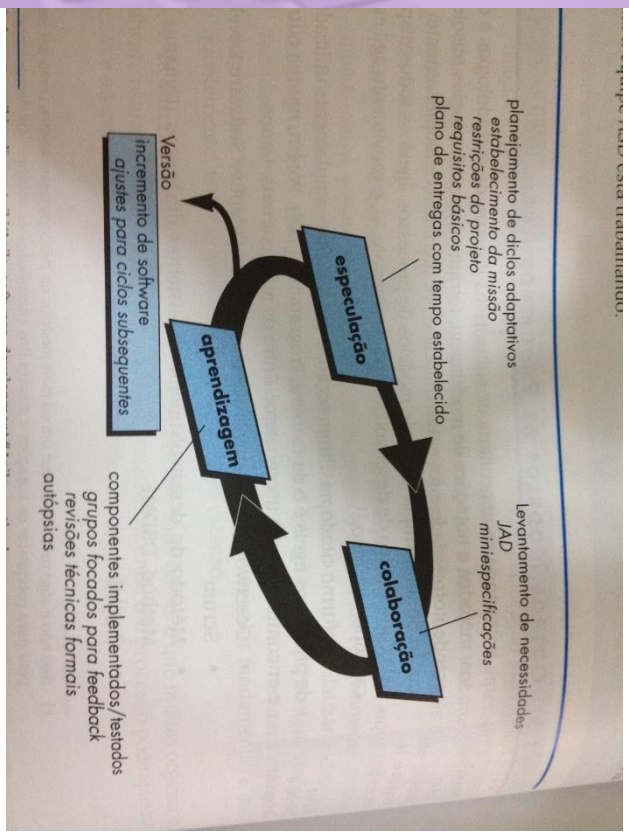
- Todos os membros da equipe refletem sobre a sprint passada.

Três questões principais são feitas na retrospectiva sobre o sprint: O que deu errado? O que deu certo? O que poderia ser melhorado para o próximo sprint?



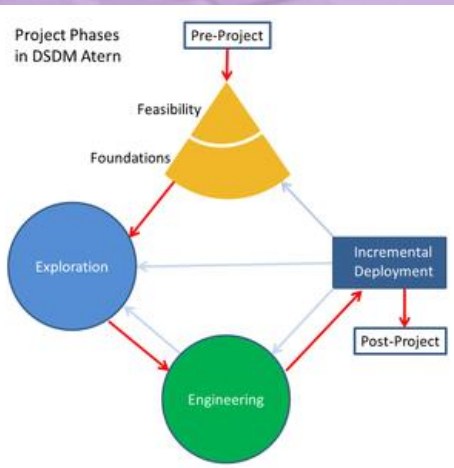
Existem outras metodologias ágeis, menos conhecidas como:

- **Desenvolvimento de Software Adaptativo (ASD)**: O apoio filosófico do ASD concentra-se na colaboração humana e na auto-organização. A auto-organização aparece quando agentes individuais independentes cooperam para criar resultados emergentes. Um resultado emergente é uma propriedade além da capacidade de qualquer agente individual. Ciclo de vida: Especulação: iniciação do projeto e planejamento adaptativo. Colaboração: pessoal motivado trabalha junto de um modo que multiplica seus talentos e resultados criativos. Aprendizagem que pode ocorrer de três modos: foco nos grupos; revisão técnicas formais; pós-conclusão.



ASD

- **Método de Desenvolvimento de Sistemas Dinâmicos**: É uma metodologia de desenvolvimento de software originalmente baseada em "Desenvolvimento Rápido de Aplicação" (RAD). DSDM é uma metodologia de desenvolvimento iterativo e incremental que enfatiza o envolvimento constante do usuário.



DSDM

Seu objetivo é entregar softwares no tempo e com custo estimados através do controle e ajuste de requisitos ao longo do desenvolvimento. DSDM é um dos modelos de Metodologia Ágil de desenvolvimento de software, e seu formato é propriedade da Agile Alliance.

- **Processo Unificado Ágil (AUP)**:Versão simplificada do RUP, adaptada à metodologia ágil.

5. KANBAN

O Kanban é uma metodologia de gestão visual originária do sistema de produção da Toyota, e tem sido amplamente adotado em diversos contextos, desde o desenvolvimento de software até a gestão de projetos, operações de manufatura e até mesmo em atividades pessoais.

Em sua essência, o Kanban é um sistema que visualiza o fluxo de trabalho, permitindo que as equipes entendam, controlem e melhorem o processo de forma contínua. O termo "Kanban" significa literalmente "cartão visual" em japonês. No método Kanban, as tarefas são representadas por cartões ou post-its e movem-se através de um quadro, dividido em colunas que representam os diferentes estágios do processo, desde a entrada até a conclusão.

Existem alguns princípios fundamentais que norteiam o Kanban:

Visualização do Fluxo de Trabalho: O quadro Kanban permite que todos na equipe vejam instantaneamente o que está sendo feito, onde está no processo e quem está trabalhando em cada item.

Limite de Trabalho em Progresso (WIP): O Kanban limita o número de tarefas que podem estar em andamento simultaneamente. Isso ajuda a evitar sobrecarga e congestionamento, focando na conclusão de tarefas antes de iniciar novas.

Feedback Contínuo e Melhoria: O Kanban promove a reflexão constante sobre o processo e incentiva a equipe a identificar oportunidades de melhoria. Ao monitorar o fluxo de trabalho e os tempos de ciclo, as equipes podem identificar gargalos e áreas de desperdício, buscando continuamente formas de otimizar o processo.

Gestão da Demanda: O Kanban ajuda a equilibrar a demanda com a capacidade da equipe, garantindo que o trabalho seja priorizado de acordo com o valor entregue ao cliente.

Flexibilidade e Adaptabilidade: O Kanban é altamente flexível e pode ser adaptado às necessidades específicas de diferentes equipes e contextos. Não há prescrições rígidas, permitindo que as equipes personalizem o processo para atender às suas necessidades únicas.

Além disso, o Kanban é conhecido por sua abordagem evolucionária, permitindo que as equipes introduzam mudanças gradualmente ao longo do tempo, em vez de realizar grandes transformações de uma só vez.

Em resumo, o Kanban é uma metodologia poderosa para visualizar, gerenciar e otimizar o fluxo de trabalho, promovendo a eficiência, colaboração e melhoria contínua nas organizações. Ele se destaca por sua simplicidade, flexibilidade e capacidade de se adaptar a uma ampla gama de contextos e tipos de projetos.

6. METODOLOGIA ÁGEIS NA ADMINISTRAÇÃO PÚBLICA

Rosa e Pereira (2021, Op.cit), apontam que, para conhecer a aplicação de métodos ágeis nas contratações para desenvolvimento de software na administração pública brasileira, o Tribunal de Contas da União (TCU) realizou um levantamento acerca das contratações, relatado no Acórdão nº 2314/2013 TCU/Plenário. As entidades consultadas foram: Tribunal Superior do Trabalho (TST); Banco Central do Brasil (Bacen); Instituto do Patrimônio Histórico e Artístico Nacional (Iphan); Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (Inep); Supremo Tribunal Federal (STF). O relatório de levantamento foi proposto pela Secretaria de Fiscalização de Tecnologia da Informação (SEFTI) com o objetivo de conhecer as bases teóricas do processo de desenvolvimento de software com métodos ágeis e práticas desse tipo de contratação (Tribunal de Contas da União, 2013).

De acordo com os autores, algumas conclusões foram apontadas:

Nas análises do TCU, destaca-se o aumento na adoção de métodos ágeis no desenvolvimento de sistemas, ainda pouco difundido em instituições públicas. Essas metodologias buscam simplificar o processo, eliminando desperdícios e entregando produtos mais rapidamente. No entanto, isso pode conflitar com as particularidades das contratações públicas, exigindo atenção aos princípios legais. Após análises, verificou-se que é possível alinhar métodos ágeis aos preceitos legais da esfera pública (TCU, 2013).

Identificaram-se riscos na adoção de métodos ágeis no setor público, como a possibilidade de falta de planejamento adequado e uso de formas de pagamento não alinhadas com resultados. Enquanto a remuneração por iterações é comum no setor privado, na esfera pública é tratada como exceção, conforme a súmula TCU 269, que estipula que o pagamento deve estar ligado a resultados ou níveis de serviço, com exceções justificadas (TCU, 2013).

O governo brasileiro promoveu o desenvolvimento ágil na administração pública ao criar um guia de projetos de software com práticas de métodos ágeis para o Sistema de Administração dos Recursos de Tecnologia da Informação (Sisp), conduzido pelo Ministério do Planejamento, Orçamento e Gestão. Esse guia foi elaborado como parte de uma série de iniciativas para facilitar e disseminar o uso dessa abordagem nas instituições públicas. Ele oferece um modelo de referência para a construção de projetos de software com práticas ágeis e terceirização do desenvolvimento, detalhando atividades, conceitos e exemplos. O guia baseia-se na definição de um modelo fundamentado em práticas ágeis, na execução de serviços de desenvolvimento, na gestão de projetos e na orientação ou adaptação de práticas que auxiliam no gerenciamento e fiscalização de projetos e contratos de desenvolvimento de software (Brasil, 2015).

Em uma pesquisa baseada na análise de estudos de casos de implementação de metodologias ágeis em instituições públicas publicadas em artigos científicos relacionados ao tema, os autores apontaram as seguintes conclusões:

6.1 Dificuldades na implementação de projetos ágeis

A falta de competências em metodologias ágeis pode ser vista como um desafio na implementação de projetos ágeis na administração pública. No caso do projeto Portal Brasileiro de Software Público, composto por uma equipe diversificada de professores, alunos de graduação e pós-graduação, e profissionais, foi necessária uma vasta experiência e formação adicional, geralmente indisponível para alunos de graduação. Para resolver esse problema, desenvolvedores experientes foram contratados para auxiliar em questões complexas e transferir conhecimento à equipe, embora em meio período devido a restrições financeiras, com a comunicação predominantemente online (Meirelles et al., 2017).

A aplicação completa de um framework com todas as suas práticas e técnicas pode enfrentar resistência na administração pública. Em um estudo relatado por Ribeiro e Domingues (2018), uma empresa pública portuguesa optou por um Scrum customizado devido à resistência à mudança, um dos principais desafios na implementação de metodologias ágeis no setor público. A falta de iniciativa das equipes para tomar decisões é outro desafio, como mencionado por Bogdanova et al. (2020), onde as equipes aguardam aprovação dos gerentes, embora os clientes/usuários dos serviços públicos deveriam desempenhar esse papel.

Em relação à cultura organizacional, resistência à mudança e baixo envolvimento das partes interessadas, Oliveira et al. (2020) identificaram esses obstáculos como significativos para a adoção de metodologias ágeis no setor público. Para projetos grandes e complexos, uma abordagem mais estruturada pode ser mais recomendada do que métodos ágeis, devido a desafios como tamanho do projeto, atrasos e questões contratuais.

Esses desafios estão alinhados com os dados do relatório global State of Agile Report, que destaca **a resistência à mudança, falta de participação da liderança e inconsistências nos processos e práticas de trabalho da equipe como os principais obstáculos para o sucesso das organizações na adoção de práticas ágeis** (DIGITAL.AI, 2020).

6.2 Benefícios da utilização de uma metodologia ágil

As estratégias de comunicação e gestão na metodologia ágil promovem uma colaboração eficaz entre todos os membros da equipe para alcançar o sucesso do projeto. No caso do Portal Brasileiro de Software Público, os benefícios dessas estratégias são destacados na implementação de um conjunto de ferramentas. Para a comunicação entre membros em diferentes locais, foram adotadas videoconferências com ferramentas de compartilhamento de tela, Internet Relay Chat (IRC) e listas de e-mail. Na gestão do projeto, uma página Wiki foi utilizada no próprio portal para validação, mantendo o gerenciamento próximo ao código-fonte e rastreando todos os recursos desenvolvidos durante o projeto (Meirelles et al., 2017).

Em outro estudo de caso apresentado por Soe e Drechsler (2018), que tratou da implementação de soluções digitais de transporte urbano entre as cidades de Tallinn e Helsinki, **os benefícios dos projetos ágeis foram associados à governança ágil e adaptativa, resultando em um aumento do valor público**. Os três domínios de valor público destacados foram a melhoria da qualidade dos serviços de mobilidade, a confiança nas instituições e a obtenção de resultados sociais.

No estudo de caso da Receita Estadual do Paraná, conduzido por Oliveira et al. (2020), que empregou técnicas ágeis de gestão de projetos baseadas no Scrum e Kanban, várias técnicas foram identificadas como impactantes para o projeto, como o uso do quadro Kanban para atividades por Squad, a definição de um Dono do Produto, reuniões diárias de status, levantamento de requisitos de histórias de usuário e a utilização da ferramenta de gestão do Backlog. **A pesquisa concluiu que as técnicas ágeis contribuíram positivamente para o desempenho dos projetos no setor público, aumentando as chances de sucesso.**

Concordando com esses resultados, o relatório global State of Agile Report destaca os cinco principais benefícios da adoção do ágil nas organizações, relacionados **à capacidade de gerenciar mudanças prioritárias, visibilidade do projeto, alinhamento entre negócios e tecnologia da informação, entrega rápida e melhoria do moral e engajamento da equipe de trabalho** (Digital.AI, 2020).

7. REFERÊNCIAS

AGILEMANIFESTO.ORG. Principles behind the Agile Manifesto, 2001. Disponível em: <http://agilemanifesto.org/principles.html>. Acesso em: 20 abril. 2024.

Brasil. Ministério do Planejamento, Orçamento e Gestão. Secretaria de Logística e Tecnologia da Informação. Guia de Projetos de Software com Práticas de Métodos Ágeis para o Sisp .Vol. 1. 2015. 90 p

Rosa, M. R e Pereira, E. N. Metodologias ágeis no contexto da administração pública: análise de estudos de caso de implementação ágil.Revista do Serviço Público (RSP), Brasília 72 (2) 479-497 abr/jun 2021.

Tribunal de Contas da União (TCU). Acórdão no 2314/2013 TCU/Plenário.2013. Disponível em:
<https://contas.tcu.gov.br/sagas/SvlVisualizarRelVotoAcRtf?codFiltro=SAGAS-SESSAO-ENCERRAD A&seOcultaPagina=S&item0=483300>.