FACULDADE DE TECNOLOGIA DE SÃO PAULO

Augusto Albuquerque Reis

SCRUMBAN - METODOLOGIA HÍBRIDA COM SCRUM E KANBAN PARA DESENVOLVIMENTO DE SOFTWARE

SÃO PAULO

2021

FACULDADE DE TECNOLOGIA DE SÃO PAULO

Augusto Albuquerque Reis

SCRUMBAN - METODOLOGIA HÍBRIDA COM SCRUM E KANBAN PARA DESENVOLVIMENTO DE SOFTWARE

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Professor Mestre David Tsai

SÃO PAULO

2021

LISTA DE SIGLAS E ABREVIATURAS

WIP	Work in Progress
SLE	Service Level Expectation
SM	Scrum Master
PO	Product Owner
CPD	Centro de Processamento de Dados
ISACA	Information Systems Audit and Control Association
ITIL	Information Technology Infrasructure Library
ITGI	Information Tecnology Governance Institute
PMBOK	Project Management Body of Knowledge
PRINCE	Project IN Controlled Environments
SEI	Software Engineering Institute
SGBD	Sistema Gerenciador de Banco de Dados
TI	Tecnologia da Informação
XP	Extreme Programming
DOD	Definition of Done

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

1. INTRODUÇÃO

O conceito de engenharia de software surgiu em 1968, durante a ‘crise do software’, conforme descrito por Sommerville (2008). O aumento na capacidade de hardware, proporcionada pela introdução de computadores baseados em circuitos integrados, permitiu a criação de softwares de maior complexidade e, por sua vez, o custo desse desenvolvimento.

E nesse novo cenário, o uso de métodos informais gerou uma crise no desenvolvimento de software, pois eles não eram suficientes para gerar previsões precisas em relação ao custo, prazo e escopo de desenvolvimentos mais complexos, portanto eram necessários novos métodos e técnicas que trouxessem maior controle sobre a complexidade desses novos sistemas.

Assim, surgiram metodologias que procuravam dividir as atividades do projeto em etapas, gerando documentações que definam e delimitem os requisitos do usuário e, dessa forma, mantendo o controle do desenvolvimento. Por outro lado, surgiram posteriormente as denominadas metodologias ágeis, que contestam o excesso de documentação, prezando pela implementação e resposta às mudanças de requisitos que surgem durante o projeto.

Com isso a agilidade tem se tornado a principal ferramenta na área de tecnologia, os times de desenvolvimento têm ganhos significativos com o uso de suas práticas. Existem metodologias ágeis para que facilite o nosso conhecimento e elas carregam regras e formas de como agir e pensar para que a eficácia dela seja maximizada.

As metodologias, tem como objetivo reduzir o máximo possível o tempo das tarefas de um analista e dar uma maior flexibilidade. A agilidade conta, principalmente, com duas metodologias, que será dado um foco o Scrum e o Kanban.

Ágil está se tornando cada vez mais popular e a maioria das empresas adapta suas metodologias. Embora Scrum seja de longe o framework Ágil usados com mais frequência, algumas organizações

acham difícil implementar e escolher o modelo Kanban para seu desenvolvimento de software.

Cada uma dessas estruturas tem seus próprios benefícios e desvantagens, e combinar dois métodos pode ser a melhor maneira de evitar quaisquer armadilhas de desenvolvimento. Esta combinação é conhecida como metodologia Scrumban.

“Scrum vs Kanban” é a batalha bem conhecida na comunidade Agile. Muitos gerentes de projeto e suas equipes tentaram definir a diferença entre Scrum e Kanban e muitas vezes seus esforços foram ineficazes. No entanto, existe uma solução lógica que tende a ser esquecida. É sobre Scrumban.

Será abordado o Scrumban e os benefícios de utilizar em times de tecnologia como metodologia ágil. Passando pelos pontos principais de flexibilidade e os principais relatórios e métodos de avaliação do time, com ênfase nos ganhos de utilização desse framework e como separadas as metodologias se tornam incompletas.

2. PROCESSO DE CRIAÇÃO DE SOFTWARE

O conceito de engenharia de software surgiu em 1968, durante a ‘crise do software’, contado por Sommerville (2008), ocorreu pelo aumento da complexidade e do custo para desenvolver um software, com isso, foi necessário desenvolver maneiras de estimar o tempo do desenvolvimento e do processo.

Com essa finalidade, nasceu as primeiras metodologias para organizar o processo de criação de software. Cada metodologia procura identificar as atividades estruturais que se apliquem a todos os projetos de software, mesmo com tamanhos e dificuldades diferentes.

Segundo Pressman (2011), uma metodologia genérica de processo de software abrange cinco atividades: comunicação, planejamento, modelagem, construção e emprego. E tais atividades podem realizadas repetidas vezes, durante um projeto, dependendo do número de incrementos de software ao longo do projeto.

2.1. Comunicação

A comunicação é a etapa de entendimento da necessidade do cliente para o software, é a parte onde levantaremos os requisitos que devemos desenvolver no decorrer do projeto.

Este conjunto de tarefas e técnicas que levam ao entendimento desses requisitos são denominados Engenharia de Requisitos, conforme Pressman (2011). Esse entendimento pode ser dividido nas seguintes atividades:

- Concepção: identificação da necessidade do negócio, da natureza da solução de software, das pessoas interessadas e do contato entre tais pessoas e a equipe de software;
- Levantamento: onde verifica como o software precisa atender as necessidades (requisitos) propostos;
- Elaboração: expansão das informações levantadas em um modelo de análise dos requisitos, permitindo a visualização do comportamento do programa;

- Negociação: análise de viabilidade (risco e custo) da implementação dos requisitos junto ao cliente.
- Especificação: documentação de todos os pontos a serem considerados no desenvolvimento de software;
- Validação: examina a documentação para viabilizar que os requisitos foram declarados de forma clara e sem inconsistências ou erros.

2.2. Planejamento

Segue para a definição do roteiro a ser seguido pela equipe de software para que os objetivos estratégicos e táticos sejam atingidos pelo projeto.

Para o início desta etapa, é necessário um entendimento do escopo do projeto para definir as atividades a ser realizadas pela equipe. O plano de projeto também deve conter cronograma de trabalho, com estimativas de custo, esforço das atividades e prazo.

2.3. Modelagem

Etapa de criação de modelos que representam o software a ser desenvolvido. Eles devem incluir a arquitetura do sistema, fluxo de informações, características desejadas e comportamentos desse sistema.

Além disso, os aspectos acima devem descrever o software tanto da perspectiva do cliente quanto técnico. Para atender essa demanda, Pressman (2011) menciona a possibilidade de utilização de um modelo de requisitos e de projeto.

2.4. Construção

Responsável pelas tarefas de codificação do software, assim como a realização de testes para detectar erros. As atividades de testes podem ser classificadas nos seguintes níveis:

- Teste de Unidade: verifica a qualidade do desenvolvimento;
- Teste de integração: verifica a integridade dos componentes;
- Teste de validação: avalia se os requisitos do usuário foram atendidos;
- Teste de aceitação: é conduzido pelo usuário, com objetivo de avaliar o funcionamento dentro do cenário de utilização.

2.5. Emprego

Etapa onde ocorre a disponibilização para a avaliação do cliente. Caso o software seja incremental, sua entrega pode ser dividida em ciclos, onde novas funcionalidades serão acrescentadas ao sistema no decorrer do tempo.

O usuário terá o suporte da equipe de desenvolvimento durante o período de avaliação. E após a avaliação, o usuário entregara um *feedback,* que servira de norte para futuras alterações na funcionalidade e que serão entregues no próximo ciclo.

3. Tecnologia da Informação (TI)

A Tecnologia da Informação (TI) é por definição um conjunto de processos, sistemas e soluções oriundas de recursos computacionais que tem como objetivo melhorar a produção, a forma de armazenamento, o meio de transmissão, acesso as informações e recursos e a segurança dos dados.

3.1. Surgimento

No final do século XIX surgiram os primeiros programas de computador e revolucionaram a transmissão de informações. Porém, foi Durante a Segunda Guerra mundial que ocorreu a transição para os computadores eletrônicos modernos e desde então evoluem rapidamente se tornando responsável por grandes avanços da humanidade.

3.2. Processamento de dados

Em 1960 os computadores começaram a se tornar importantes para as grandes e médias empresas, mas eram limitadas em relação as aplicações e incompatíveis entre si.

Os avanços da informática acompanhavam o desenvolvimento do hardware como por exemplo em custo, velocidade e as aplicações, onde esse construídos a partir do zero absoluto, pelo fato de não haver empresas empenhadas no desenvolvimento destes pacotes.

Em 1970, as linhas telefônicas possibilitaram o acesso a terminais remotos de computadores, fazendo das telecomunicações a base tecnológica aliada aos seus avanços, trazendo automatização das atividades burocráticas realizadas pelas empresas da época.

Toda execução de processos ocorria nos então denominados CPD (Centro de Processamento de Dados) que eram responsáveis pela manipulação e tratamento dos dados, onde o acesso era realizado através de relatórios gerados pelo sistema ou computadores conectados

a central. No entanto, havia resistência dos usuários para utilização de um novo sistema e concentração das operações.

3.3. Sistemas de informações

Por volta do ano de 1970 os avanços tecnológicos passaram a possibilitar o melhoramento e adequação dos sistemas atendendo as necessidades das empresas, no entanto neste período havia muita centralização das informações.

Os terminais passaram a ser flexíveis, permitindo o computador executar diversas tarefas simultaneamente inúmeros usuários conectados. Então surgiu também os pacotes de software, onde era combinada a flexibilidade dos terminais, estimulando uma série de inovações que passariam a ser conhecidas como “sistemas de apoio à decisão”.

De acordo com Keen (1996, p. XXXVII), “a maior evolução técnica dessa época foi a passagem do processamento de transações para o gerenciamento de banco de dados." Surgiu a partir daquele momento os sistemas gerenciadores de banco de dados (SGBDs), que organizava os dados de maneira eficiente e eficaz, inibindo a duplicidade permitindo rápidas análises através de consultas. E então os CPDs passaram se tornar bibliotecas de informações.

3.4. Inovação e vantagem competitiva

Em 1980, ocorreram diversas mudanças tecnológicas, em especial nas tecnologias utilizadas em escritórios e nos microcomputadores, foi então que o termo “Tecnologia da Informação” se tornou mais utilizado.

O SGBD foi disponibilizado nos computadores pessoais e nos softwares de baixo custo que dominaram o mercado, de forma a concentrar as atenções do mercado em buscar novas estratégias baseadas nas tecnologias de TI. As telecomunicações e os computadores permitiram a utilização da TI em empresas ao redor do mundo.

Foram então criados os programas de “consciencialização gerencial” para os altos executivos e o Centro de Suporte ao Usuário (CSU) ou o denominado Help Desk, onde todos os usuários podem esclarecer dúvidas, além também de oferecer a consultoria na área tecnológica, ambos para permitir acesso e conhecimento às ferramentas de TI utilizadas nas empresas em uma maior escala de aceitação.

3.5. Reestruturação de negócios

Durante os anos de 1990 a 1999, os sistemas abertos, integração e modelos se tornaram recursos primordiais para os departamentos de sistemas extinguindo a incompatibilidade. Esta integração tecnológica aumentaram os índices de performance flexibilizando e facilitando a troca e o acesso às informações otimizando o funcionamento das empresas.

“A TI é reconhecida como fator crítico de capacitação, principalmente através das telecomunicações, que permite eliminar barreiras impostas por local e tempo às atividades de coordenação, serviço e colaboração”. (KEEN, 1996, p. XLIX). A mudança em quase todas as áreas do negócio e da tecnologia ocorreu de forma mais acelerada. As modificações e a utilização das ferramentas de TI se tornaram mundialmente conhecidas e as distinções entre computador e comunicação são extintas, tendo um enorme impacto no mundo Business. O computador se torna elemento de TI indispensável em uma organização.

4. GOVERNANÇA

A Governança corporativa é o conglomerado de regras, processos práticas, políticas, leis, regulamentos e instituições que regulam as diretrizes da empresa. O termo sugere o estudo sobre as relações entre diversas pessoas envolvidas, denominadas tecnicamente de “stakeholders” que tem por objetivos orientar a forma em que a empresa é dirigida. Os principais atores podem ser acionistas, a alta administração e o conselho de administração.

Outros participantes, como por exemplo os funcionários também são incluídos, fornecedores, clientes, bancos e outros credores, instituições reguladoras e a comunidade em geral. É possível notar que esta palavra é muitas vezes utilizada de maneira errônea com um sinônimo de Administração. No entanto a governança tem um foco voltado aos stakeholders e a Administração é voltada, de maneira geral, para a gestão interna das corporações ou empresas.

A Governança corporativa utiliza conceitos de múltiplas abordagens e uma de suas preocupações é assegurar a aceitação dos stakeholders aos códigos de conduta previamente acordados, valendo-se de mecânicas que se esforçam para eliminar os conflitos de interesse.

Outros temas em governança corporativa são abordados, tais como a preocupação com o ponto de vista dos outros stakeholders que não os acionistas e dos diversos modelos de governança corporativa. O governo das sociedades é constituído por uma agregação de mecanismos e regras que estabelecem modelos de controle na gestão das sociedades de capital aberto. A governança corporativa tem o objetivo de diminuir problemas que eventualmente podem surgir na relação entre gestores e acionistas.

Existe um interesse no assunto de governança corporativa desde meados de 2001, devido aos colapsos de grandes corporações norte-americanas, foi a partir daí que o governo federal da américa do norte aprovou a Lei *Sarbannes-Oxley*, a fim de restaurar a confiança do público em geral na governança corporativa.

4.1. Governança de TI

Governança de TI ou Governança de Dados, de acordo com a RNP, está fortemente relacionada ao constante desenvolvimento de estruturas de competências e habilidades estratégicas voltada aos profissionais de TI que são responsáveis pelo planejamento, implantação, controle e monitoramento de programas e projetos de governança, considerado um requisito fundamental para as corporações e empresas, tanto nos aspectos operacionais, quanto nos aspectos legais.

De acordo com o ITGI (*Information Tecnology Governance Institute*) “Governança de TI é uma parte integral da Governança Corporativa e é formada pela liderança, estruturas organizacionais e processos que garantem que a TI sustenta e melhora a estratégia e objetivos da organização”.

As diretrizes da governança em TI fazem parte de um complexo sistema de métricas utilizados pelas empresas, que tem o intuito de alcançar um diferencial competitivo diante do mercado B2B (*business to business*). As empresas que detém níveis de governança elevados conseguem melhores índices de reputação na segurança de dados e através disto alcançam a eficiência e eficácia desejada em suas operações.

5. MODELOS DE PROCESSO DE SOFTWARE

Conforme descrito na Introdução, à medida que o software ganhava maior complexidade, metodologias começaram a surgir para organizar seu processo de criação. E a forma esse processo é representado poderá variar de uma abordagem para outra, gerando diferentes modelos de processo.

A seguir, serão descritos diferentes tipos de modelos de processo, baseando- se na divisão utilizada por Pressman (2011).

5.1. Modelos de Processos Prescritivo

São modelos que prescrevem elementos do processo de software, além da maneira como estão inter-relacionados. Esses elementos seriam as atividades metodológicas, as ações de engenharia de software, as tarefas envolvidas, os produtos gerados durante o processo, o controle de qualidade, e como o projeto irá se adaptar às mudanças de escopo.

5.1.1. Modelo Cascata

Surgido em 1970, o modelo em cascata considera o desenvolvimento de software como um processo linear. Por isso, se houver mudanças de requisitos durante o desenvolvimento, será revisar todas as etapas do processo para incluir as alterações levantadas.

De acordo com Sommerville (2008), o modelo pode ser dividido em cinco fases sequenciais:

- Definição de requisitos: levantamento das necessidades e expectativas do cliente em relação ao software;
- Projeto de sistema e software: etapa onde são mapeados os dados de entrada e saída, a estrutura do sistema, as interfaces entre usuário e o sistema, além da interação entre os diferentes módulos do sistema.

- Implementação e teste de unidade: construção do software através de uma linguagem de programação, realizando testes de cada componente desenvolvido para verificar sua adequação ao que foi especificado.
- Integração e teste de sistema: execução dos componentes do software de forma integrada para identificação de possíveis erros na codificação e para verificar se os requisitos foram atendidos.
- Operação e manutenção: etapa de correção de erros que tenham ocorrido após a entrega do software (onde o software já estará em operação pelo cliente), além do levantamento de melhorias que podem gerar um novo ciclo de desenvolvimento.

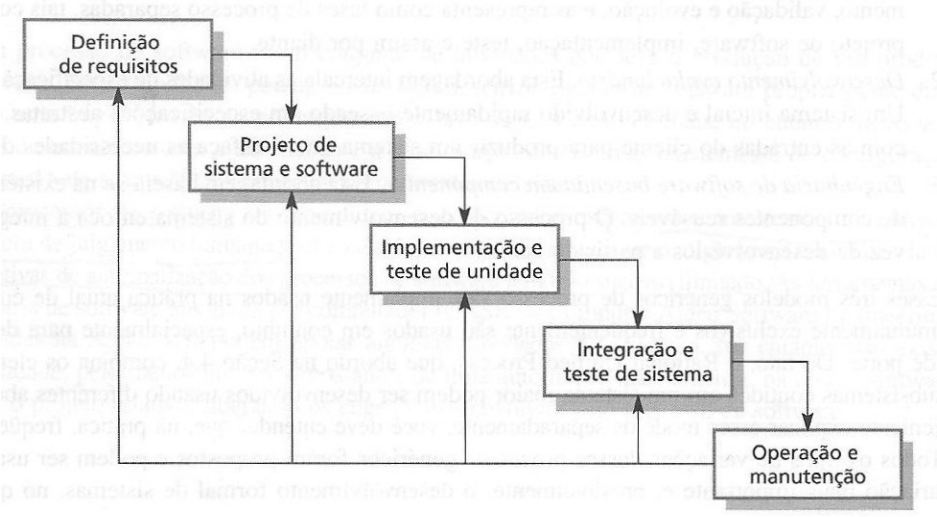

Figura 1 – Ciclo de Software do Modelo Cascata

5.1.2. Modelo Incremental

Este modelo pressupõe que novas funcionalidades serão agregadas ao software no decorrer do tempo. Para cada incremento de software, o processo terá um modelo de fases linear, semelhante ao modelo em cascata, e representam entregas de software durante o período do projeto.

Por considerar que software será incrementado ao longo do tempo, o projeto implementará inicialmente as principais funcionalidades do sistema, no incremento descrito por Pressman (2011) como Produto Essencial. As funcionalidades serem construídas durante os próximos incrementos serão agregadas a esse núcleo principal.

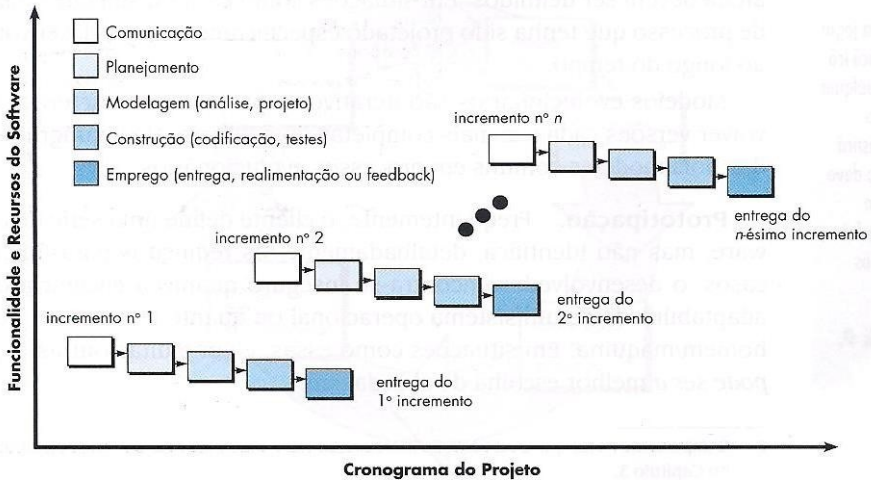

Figura 2 – Modelo Incremental

5.1.3. Modelo Evolucionário

Trata-se de um modelo onde o haverá a produção de novas versões do sistema de software no decorrer do projeto, tornando-se mais completo a cada nova iteração. Essa característica de evolução do software vem para lidar com situações em que os requisitos desse sistema ainda não estão completamente definidos pelo usuário.

Pressman (2011) apresenta dois tipos de modelos para processos evolucionários: Prototipação e Modelo Espiral.

5.1.3.1. Prototipação

Utiliza protótipos como forma de levantamento de requisitos junto aos usuários. O protótipo seria uma versão do programa com o objetivo de dar ao usuário uma prévia do sistema de software.

Para a construção do protótipo, são levantados os requisitos iniciais, verificando quais áreas necessitam de uma definição mais ampla do escopo. Após identificada a necessidade de uma iteração do software, segue a modelagem e a construção dessa versão do sistema, que será focada nos aspectos visíveis aos usuários.

Com a apresentação do protótipo, as partes interessadas terão uma noção do funcionamento do sistema final, enquanto a equipe de projeto terá um *feedback* para aprimorar os requisitos levantados.

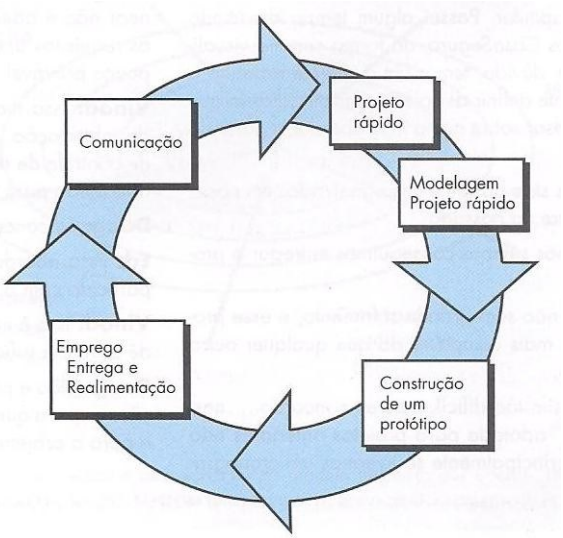

Figura 3 – Processo de Prototipação

5.1.3.2. Modelo Espiral

O modelo espiral caracteriza-se por ter uma abordagem cíclica quanto ao desenvolvimento de software. No decorrer dos ciclos, haverá pontos de controle para validar o que foi produzido, definindo como o software deve evoluir no desenvolvimento do próximo ciclo.

A disposição dos ciclos do modelo espiral pode ser vista na figura 4. O primeiro ciclo é iniciado no centro da espiral, que segue de dentro para fora. Quando houver a necessidade de uma nova iteração, um ciclo será acrescentado à espiral, aumentando sua extensão que, como ao todo, representa o ciclo de vida do software.

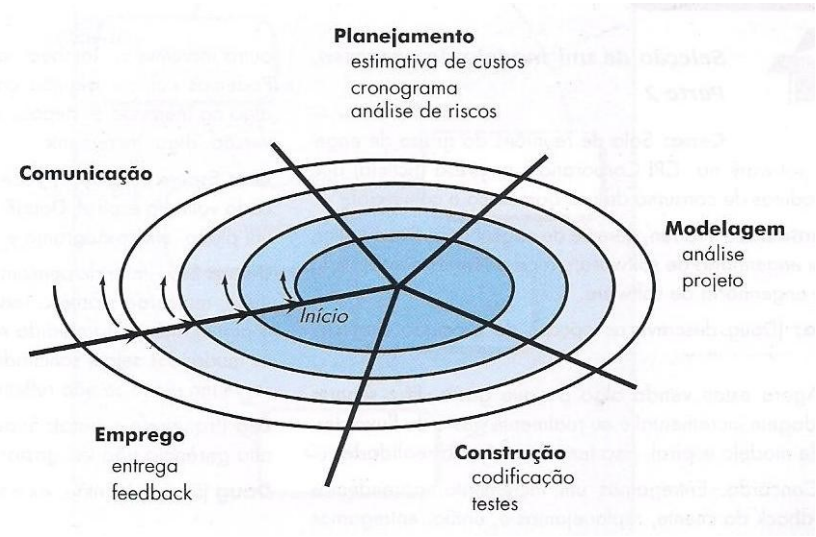

Figura 4 – Modelo Espiral

5.1.4. Modelo Concorrente

Modelo onde as atividades metodológicas do desenvolvimento são representadas por estados, que definem a situação daquela atividade num dado momento. Esses estados estão relacionados de maneira concorrente dentro da atividade, e a transição de um estado para outro ocorrerá de acordo com os eventos pré-definidos pela equipe responsável.

E neste modelo, essas atividades existem concorrentemente, mas cada uma pode estar num estado diferente. Além disso, o estado de uma das atividades pode levar à mudança de estado numa outra (se assim for determinado na relação de eventos dessa última atividade);

A figura 5 exemplifica uma atividade de modelagem neste modelo, que permanece no estado “Inativo” até que a atividade de comunicação com o cliente tenha terminado, transitando para o estado “Em desenvolvimento” os requisitos estiverem definidos, podendo transitar para o estado “Aguardando modificações” se novas informações precisarem ser levantadas.

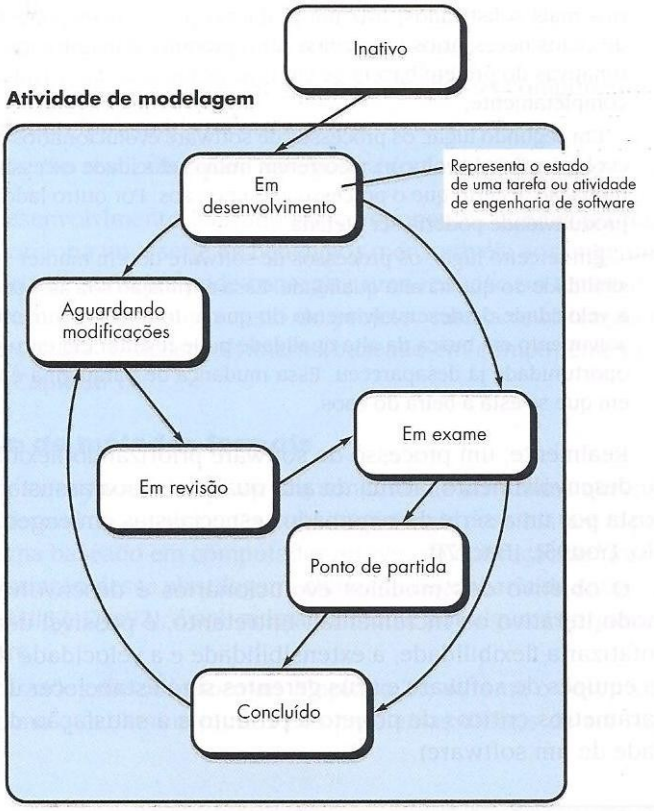

Figura 5 – Modelo Espiral

5.2. Modelos de Processos Especializado

São modelos que utilizam características dos modelos tradicionais, mas também adicionam uma nova abordagem para uma ou mais etapas do processo de software, quando há a necessidade de um foco especializado ou uma meta específica para o desenvolvimento em questão.

5.2.1. Desenvolvimento Baseado em Componentes

Neste modelo de processo, as funcionalidades a serem incorporadas ao sistema são divididas em componentes. Assim, o desenvolvimento de software utilizará uma abordagem iterativa para incorporar tais componentes de software pré-definidos ao sistema. As

atividades metodológicas referentes à modelagem e construção do sistema irão identificar os componentes a serem incluídos.

Os componentes poderão ser desenvolvidos como classes orientadas a objetos ou pacotes de classe, segundo Pressman (2011). Esse modelo busca a reutilização de software, diminuindo o tempo e o custo da implementação.

O desenvolvimento por componentes possui as seguintes etapas:

- Pesquisa de componentes disponíveis e se podem ser aplicados à situação;
- Avaliação sobre a forma de integração dos componentes;
- Projeção da arquitetura do software, considerando a inclusão dos componentes;
- Integração dos componentes ao sistema.
- Teste das funcionalidades implementadas.

5.2.2. Modelo de Métodos Formais

Modelo que utiliza técnicas baseadas na matemática para especificar o sistema de software. Por meio da teoria dos conjuntos e da notação lógica, o objetivo desse modelo é criar uma definição mais clara dos requisitos, evitando que o sistema seja especificado de forma ambígua, incompleta ou inconsistente. Dessa forma, procura-se garantir que o software seja desenvolvido de forma mais apurada.

5.2.3. Desenvolvimento Orientado a Aspectos

Segundo Pressman (2011), para o desenvolvimento de software orientado a aspectos, o termo “aspectos” pode ser descrito como restrições impostas pelo cliente que se estendem por funções, recursos e informações desse sistema. Para atender a essas necessidades, esse desenvolvimento propõe uma abordagem de métodos e processos para definição, especificação, projeção e construção de aspectos.

5.3. Processo Unificado

O Processo Unificado, também como conhecido como *Rational Unified Process* (RUP), é um modelo de processo de software orientado a objetos que possui características de metodologias tradicionais, mas também é iterativo e incremental. Ele utiliza o UML como ferramenta para a representação de requisitos, durante a modelagem de dados.

Quanto ao incremento de software, ele pode ser iniciado antes mesmo da finalização da iteração anterior, fazendo com que as fases do modelo ocorrem de maneira concorrente e escalonada no decorrer do projeto. Estas fases são:

- Concepção: levantamento das necessidades de negócio, assim como a identificação dos recursos e riscos envolvidos no projeto, fornecendo subsídios para avaliar sua viabilidade. Nesta fase também é proposta uma arquitetura para o sistema, e é definido um cronograma para o projeto.
- Elaboração: detalhamento da arquitetura do sistema, representando-a em modelo de caso prático, modelo de requisitos, modelo de projeto, modelo de implementação e modelo de emprego. À partir dessa análise, o planejamento da fase anterior pode sofrer alterações para que o projeto continue atendendo ao escopo, prazo e riscos levantados.
- Construção: Codificação dos componentes previstos na arquitetura e referentes ao incremento de software em questão, incluindo testes de unidades e de integração.
- Transição: Entrega do software para testes do usuário, que poderá fornecer feedback sobre problemas a serem corrigidos e/ou solicitações de mudanças. Ao final desta fase, será disponibilizada uma versão utilizável do sistema, juntamente com materiais contendo informações de apoio.
- Produção: acompanhamento da utilização do software lançado, fornecendo suporte ao usuário e registrando erros e mudanças desejadas.

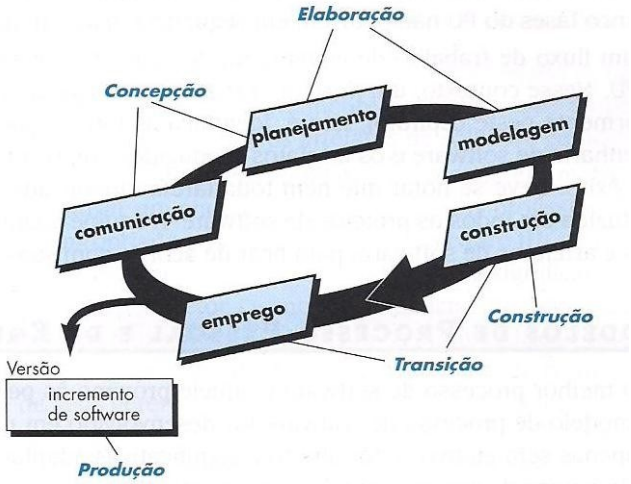

Figura 6 – Modelo Espiral

5.4. Processo de Software pessoal

Modelo que propõe treinamento aos profissionais para conhecer o processo, a arquitetura do software gerado e como medir sua qualidade. Dessa forma, esse profissional terá subsídios para planejar o projeto, assim como a identificação e compreensão dos tipos de erros passíveis de ocorrer nos artefatos de software produzidos no projeto.

As atividades desse modelo são as seguintes:

- Planejamento: identificação dos requisitos e estimativa do tamanho do software, dos recursos necessários e de possíveis erros. Esta atividade também envolve a definição das atividades referentes ao desenvolvimento e o estabelecimento de um cronograma.
- Projeto em alto nível: Especificação externa dos componentes de software a serem desenvolvidos, sendo solicitada a criação protótipos caso haja incerteza nessa especificação.
- Revisão de projeto de alto nível: utilização de métodos formais para verificar possíveis erros no projeto.
- Desenvolvimento: Codificação dos componentes de software, que é revisada, compilada e testada.

- Autópsia: Análise da eficácia do processo, além de propor melhorias para aumentar sua eficácia

5.5. Processo de Software em equipe

O Processo de Software em Equipe tem por objetivo criar equipes de projetos que consigam se organizar para produzir software com qualidade, conforme descrito por Pressman (2011). Essas equipes serão capazes de entender suas metas e objetivos, definir papéis e responsabilidades para seus membros, além de poder medir sua produtividade e qualidade.

Para ter essas equipes autodirigidas, o modelo de processo utiliza roteiros, formulários e padrões para orientar seus membros durante as atividades. Isso inclui tanto as atividades metodológicas, que são similares à do Processo de Software Pessoal, quanto as mais específicas para cada equipe.

6. MÉTODOS: ÁGEIS E TRADICIONAIS

Metodologias de gestão, embora cada vez mais em voga no mundo corporativo, não são uma novidade tão grande assim. Na verdade, métodos de se gerenciar recursos e projetos estão presentes há milhares de anos na sociedade, embora não tenham a mesma formalização que metodologias como PDCA, GPD (Gestão por Diretrizes) e Scrum.

Foram esses conhecimentos que permitiram a execução de grandes projetos da humanidade, como por exemplo a construção da Grande Muralha da China. Sem um método de organizar a força de trabalho, horários e os recursos necessários para a construção, tal grandiosidade nunca teria sido concluída.

Durante muitos anos, os meios mais avançados de gestão eram de conhecimento quase que exclusivo dos governos. Foi a partir da Revolução Industrial que houve um ‘boom’ de empresas e produtos, o que aumentou a competitividade do mercado, exigindo cada vez mais eficiência nos processos e na organização interna.

Nesse contexto, surgem as metodologias de gestão modernas, que atraem a atenção dos profissionais independente de sua área, especialmente pela capacidade de otimizar processos e reduzir custos, indicadores de sucesso para qualquer projeto.

Novamente estamos passando por uma revolução na forma como os negócios são feitos e as empresas são geridas. As metodologias ágeis, embora não tão novas, mas bastante atuais, propõem mudanças de paradigmas que tangem desde o planejamento de projetos até organogramas e a forma como são tomadas decisões estratégicas (confira também o nosso texto de metas Bottom Up e Top Down e entenda melhor do que estamos falando).

Podemos então dividir os principais métodos de gestão da atualidade em dois grupos: as metodologias tradicionais e as metodologias ágeis.

6.1. Ágil

A metodologia ágil pode ser definida como um método de gerenciamento de projeto colaborativo, capaz de segregar grandes projetos a fim de possibilitar à equipe o trabalho em conjunto.

A principal característica do método ágil é a flexibilidade e adaptabilidade que permitem mudanças em todo o ciclo de desenvolvimento do projeto, trabalhando com constantes iterações (sprints) que podem ser entregues parcialmente em curto prazo (em média de uma a três semanas, possibilitando o feedback do cliente e abrindo margem para alterações no projeto e em cada entrega parcial.

O manifesto ágil possuí 4 valores fundamentais e 12 princípios:

ÍNDICE	VALORES
1	Os indivíduos e suas interações acima de procedimentos e ferramentas.
2	O funcionamento do software acima de documentação abrangente.
3	A colaboração com o cliente acima da negociação e contrato.
4	A capacidade de resposta a mudanças acima de um plano pré-estabelecido.

Tabela 1 - 6.1 - Tabela de valores ágeis

ÍNDICE	PRINCÍPIO
1	Garantir a satisfação do cliente, entregando rápida e continuamente software funcional;
2	Até mesmo mudanças tardias de escopo no projeto são bem-vindas.
3	Software funcional é entregue frequentemente (semanal ou mensal - o menor intervalo possível);
4	Cooperação constante entre as pessoas que entendem do 'negócio' e os desenvolvedores;
5	Projetos surgem por meio de indivíduos motivados, devendo existir uma relação de confiança.
6	A melhor forma de transmissão de informação entre desenvolvedores é através da conversa 'cara a cara'
7	Software funcional é a principal medida de progresso do projeto;
8	Novos recursos de software devem ser entregues constantemente. Clientes e desenvolvedores devem manter um ritmo até a conclusão do projeto.
9	Design do software deve prezar pela excelência técnica;
10	Simplicidade – a arte de maximizar a quantidade de trabalho que não é feito – é essencial;
11	As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
12	Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento.

Tabela 2 – 6.1 - Tabela de princípios ágeis

6.2. Tradicional

A metodologia tradicional tem conceitos baseados em projetos com fases progressivas, logo a organização do projeto obtém um modelo ou formato de cascata (Waterfall).

Geralmente este método é composto e divido em 5 fases, sendo elas: Requisitos, Design, Implementação, Verificação e Manutenção, respectivamente.

ORDEM	PROCESSO
1	Requisitos
2	Design
3	Implementação
4	Verificação
5	Manutenção

Tabela 3 - 6.2 - tabela de processos tradicionais

Neste modelo cada fase deverá ser concluída antes de seguir para a próxima, e têm características mais detalhadas e rígidas e é projetado de forma a evitar que após a conclusão das fases haja retorno da equipe para desenvolvimentos posteriores.

Eventualmente pode ocorrer problemas futuros, caso não estejam claras as metas e tarefas antes do início da fase. Entretanto, o gerenciamento do projeto baseado nesta abordagem é comprovado, pois visualiza cada fase do projeto de forma lógica, facilitando a compreensão.

6.3. Visão geral

As metodologias aqui abordadas são categorizadas como tradicionais e ágeis, e cada uma delas possuí diferentes aspectos em relação a abordagem no gerenciamento da TI. A “Tabela - Tabela abordagem dos métodos” pode identificar o tipo de abordagem de cada Framework aqui analisado:

METODOLOGIA	FRAMEWORK
Ágil	Kanban
Ágil	Scrum
Ágil	ITIL
Ágil	XP
Tradicional	PMBOK
Tradicional	CMMI
Tradicional	COBIT
Tradicional	PRINCE2

Tabela 4 – 6.3 - Tabela abordagem dos métodos

As metodologias Tradicionais e Ágeis possuem inúmeras características fundamentadas em conceitos e métodos no qual podemos destacar as seguintes abordagens: Foco, Domínio, Documentação, Qualidade, Processo, Organização, Planejamento, Perspectiva de mudança, Requisitos, Gerenciamento, Liderança, Desempenho e Retorno.

ABORDAGEM	ÁGIL	TRADICIONAL
Foco	Pessoas	Processos
Domínio	Exploratório	Previsível
Documentação	Mínima (necessária)	Completa
Qualidade	Voltada ao cliente	Voltada ao processo
Processo	Iterativo	Linear
Organização	Auto-organizada	Gerenciada
Planejamento	Baixo	Alto
Perspectiva de mudança	Adaptabilidade	Sustentabilidade
Requisitos	Constantemente atualizado	Fixados no plano do projeto
Gerenciamento	Descentralizado	Autocrático
Liderança	Colaborativa	Comando e controle
Desempenho	Valor de negócio	Conformidade com o plano
Retorno	Constante durante o projeto	Final do projeto

Tabela 5 – 6.3 - Tabela características ágeis e tradicionais

7. METODOLOGIA HÍBRIDA

A metodologia híbrida é a utilização de múltiplos métodos e ferramentas que tem como por objetivo trazer aspectos positivos das metodologias tradicionais e ágeis, com o intuito de obter os melhores resultados durante o planejamento, execução ou monitoramento dos projetos.

Os métodos ágeis se tornaram um grande aliado para os gestores de projetos que almejam soluções mais rápidas e inovadoras, entretanto, há um grande desafio em integrar metodologias tradicionais ao método ágil. Com a metodologia ágil é possível ter ganhos em termos de tempo, porém no aspecto de documentação e planejamento pode não ser tão bem estruturado quanto as abordagens tradicionais. Ou seja, os métodos ágeis e tradicionais possuem características que podem ter uma melhor aplicação em diferentes cenários, que pode variar de acordo com a necessidade de cada projeto.

Foi diante deste cenário que surgiu o conceito de metodologia híbrida, que pode suprir a necessidade de diferentes etapas de um projeto com o aproveitamento das abordagens tradicionais nas etapas de planejamento e documentação e com a abordagem ágil para um ciclo de soluções rápidas e entregas parciais e interativas.

Ao adotar uma metodologia híbrida é fundamental entender o contexto do projeto e da empresa e avaliar a urgência na entrega em que o prazo deve estar definido, pois as metodologias híbridas podem combinar a necessidade de rapidez de implementação (adquirida através dos métodos ágeis) e uma documentação bem estruturada do projeto (caraterística dos métodos tradicionais).

8. KANBAN

Kanban é um sistema visual de gestão de trabalho, que busca conduzir cada tarefa por um fluxo predefinido de trabalho.

Em geral, o conceito de Kanban pode ser definido pelos seguintes itens:

- **O sistema visual**: um processo, definido em um quadro com colunas de separação, que permite dividir o trabalho em segmentos ou pelo seu status, fixando cada item em um cartão e colocando em uma coluna apropriada para indicar onde ele está em todo o fluxo de trabalho.
- **Os cartões**: que descrevem o trabalho real que transita por este processo.
- **A limitação do trabalho em andamento**: que permite atribuir os limites de quantos itens podem estar em andamento em cada segmento ou estado do fluxo de trabalho.

O Kanban pode ser considerado também como uma metodologia ágil exatamente por ter o objetivo de evitar a procrastinação e render mais no dia a dia.

Isso acontece porque todo o sistema é pautado de uma forma organizada para tornar o workflow mais produtivo.

Essa lógica de pensamento é muito vista na metodologia lean, onde se evitam desperdícios.

A junção desses conceitos pode ser chamada de “lean Kanban”, onde o fluxo de trabalho segue por etapas de forma enxuta.

Portanto, o método Kanban pode ser descrito como um processo para melhorar gradualmente tudo o que você faz.

8.1. História

O Kanban foi introduzido pela primeira vez por Taiichi Ohno, na indústria de manufatura japonesa em 1940, dentro do Sistema Toyota de Produção. Na época, ele era um cartão para sinalizar conclusão de um processo, fazendo com que novas demandas fossem puxadas para produção.

8.2. Objetivo

O Kanban tem o foco em priorizar a produtividade e organização das entregas, com objetivo de proporcionar um trabalho transparente e reduzir tempos de inatividades durante o processo.

8.3. Método

Trata-se de um sistema de controle e gestão de estoque e fluxo, onde utiliza-se de pequenos cartões.

Em projetos de criação de software, são comuns para demonstrar o fluxo das atividades e ter um controle de quem e o que está sendo executado naquele momento. O sistema é uma das variantes mais conhecidas do Just in Time.

8.4. Tipos de Kanban

O Kanban não se limita a apenas um estilo de implementação ou atuação, ele pode se adequar ao que sua empresa precisa no momento. A metodologia é representada por um board, conhecido como quadro Kanban.

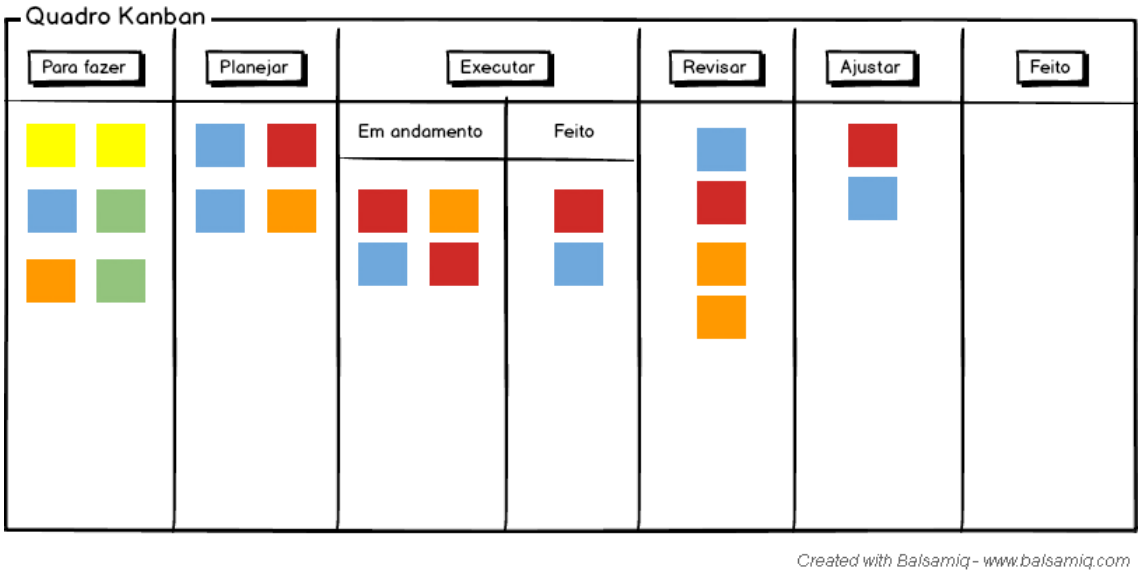

Figura 7 – Quadro de Atividades

8.4.1. Kanban de Produção

O modo de Produção, é fundamental que os colaboradores identifiquem a sequência do projeto, já por meio de murais ou softwares especializados. Dentro do quadro fazemos uma divisão em:

- To do: o que deve ser feito;
- Doing tarefas em execução;
- Done: tarefas concluídas.

8.4.2. Kanban de Movimentação

É utilizado por departamentos envolvidos em uma determinada tarefas, onde, recebem uma notificação para iniciarem a produção ou aguardam um alerta para realizá-la:

8.5. Benefícios

Como mostrado o Kanban apresenta diversos aspectos e benefícios no ambiente de trabalho, uma vez que se adapta a sua empresa. De acordo com o The State of Agile, de 2018, 55% das empresas que optaram por adotar metodologia ágil o fizeram melhorar sua produtividade.

8.5.1. Otimizar a Produtividade

O Kanban disponibiliza de dados atuais para cada projeto, permitindo maior controle dos processos e atividades. Isso potencializa a produtividade, por encurtar caminhos e torna cada execução mais direcionada.

Ele serve como forma de dar autonomia aos profissionais, organizando seu processo produtivo, evitando que os funcionários fiquem ociosos.

8.5.2. Priorizar Tarefas Importantes

Por ter um apelo visual, a metodologia destaca o fluxo de trabalho e explora atividades paradas e que foram finalizadas. O Kanban, possui um sistema hierárquico, que é composto por sinalizações de prioridade nos cartões, permitindo balancear as demandas, destacando as mais importantes até as de menor importância no momento.

8.5.3. Reduzir Custos

Independente do porte ou segmento de atuação de seu negócio a empresa trabalha com eficácia e estabelece um fluxo de trabalho, o que impacta a redução de gastos cortando o desperdício de materiais.

8.5.4. Incentivar a Comunicação Entre a Equipe

A colaboração é a peça-chave para o sucesso da metodologia. O Kanban preza pela comunicação interna, com troca de ideias, ajudas e feedbacks, com foco na produção.

8.6. Aplicado a Desenvolvimento de Software

Em desenvolvimentos de software o Kanban é comum pelo seu jeito simples e direto, reduzindo principalmente o custo e ociosidade do analista. Em grande parte, o Kanban é aplicado em board ou software, nas divisões de backlog, design, desenvolvimento, teste, deploy e pronto.

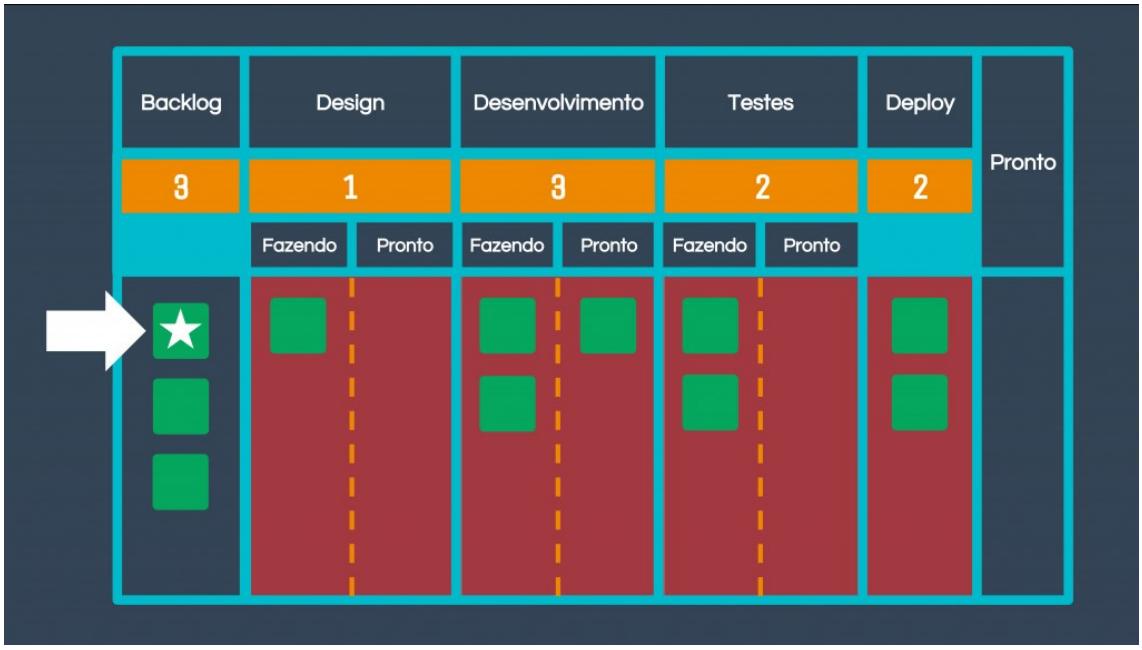

Figura 8 - Quadro em Desenvolvimento de Software

8.6.1. Backlog

Backlog é a parte onde fica armazenadas as atividades que precisam ser desenvolvidas, geralmente são organizadas em uma ordem de prioridade, sempre do mais crítico para o menos crítico.

8.6.2. Design

Etapa onde o analista pega a demanda vinda do backlog e faz o entendimento da tarefa para execução. Quando essa atividade começa a ser realizada a etapa dela sai do backlog e vai para a etapa de Design fazendo, quando o analista completa a tarefa a mesma e modificada no board para o status pronto, aguardando sua próxima fase do processo.

8.6.3. Desenvolvimento

Nesta etapa o analista é responsável por desenvolver a melhoria solicitada pelo cartão em que está executando. O mesmo processo ocorre em todas as etapas, ao receber a demanda o status dele é modificado para fazendo e ao ser finalizado muda para pronto.

8.6.4. Testes

Nesta etapa, uma pessoa diferente de quem desenvolveu assume a tarefa, ela é responsável por realizar um teste da função solicitada e em como isso afeta a estrutura do projeto, se impactou em outro momento do software.

8.6.5. Deploy

Ao finalizar os testes, a etapa de deploy ocorre para enviar essa melhoria ou correção, até o cliente em questão. Nesta fase do Kanban, ela pode ser realizada pelo teste ou pelo desenvolvedor, uma vez que foi tudo aprovado.

8.6.6. Pronto

Melhoria estando correta, ela é enviada para o destinatário ou integrada ao sistema de produção, onde estará disponível para uso total.

8.6.7. Erros

É possível que após a realização de alguma etapa encontramos erros, este pode ser relacionado a qualquer uma das etapas. Com o erro encontrado a tarefa (cartão) não volta para a etapa anterior, ela permanece no status atual e um novo cartão é criado. Com o novo cartão a vista, esta passa por todas as etapas, porém com uma diferença, ele não é iniciado no backlog, uma vez que é um defeito, ele começa diretamente na etapa de design, onde o analista de teste e o desenvolvedor tem o entendimento do problema. A criação do novo cartão, não faz o antigo sumir, ele é apresentado com um impedimento, onde os dois cartões (original e defeito) andem juntos até o encerramento.

8.7. Papeis

O Kanban não precisa necessariamente ter responsáveis pelo seu papel, por isso é comum a usabilidade em diversas áreas de uma empresa. Entretanto, nos últimos anos dois papeis foram emergindo nas empresas adequas ao Kanban.

8.7.1. Service Request Manager

O servisse request (SRM) manager é responsável por compreender as necessidades e expectativas do cliente. Ele lidera o processo de seleção dos itens de trabalho. Eles iniciam discussões que levam a soluções de tomada de decisão. Embora tenha semelhança com o product owner do scrum, ele não é equivalente tendo, dentro de seus projetos, ações diferentes.

O principal objetivo dos SRM é servir como um facilitador e gerente de risco em Kanban, contendo em sua maioria os principais desempenhos:

- Solicitar itens de trabalho da lista de pendencias do produto e facilitar a priorização.
- Apresentar políticas para o sistema que combinam decisões.
- Melhorar a governança corporativa, consistência dos processos.
- Reduzir riscos pessoais relacionados a indivíduos.

Para ter um destaque maior como SRM deve conhecer muito bem seus clientes e entender suas necessidades. Em geral o SEM, se adequa perfeitamente a uma pessoa capaz de interação e compreensão do cliente.

8.7.2. Service Delivery Manager

O service delivery Manager (SDM) é responsável por melhorar a eficiência do fluxo de trabalho. Essa função também é conhecida como gerenciador de fluxo ou mestre de fluxo. Este papel não é equivalente a um scrum master, embora tenha suas semelhanças.

O principal objetivo do SDM é garantir um fluxo de trabalho limpo. Ele ajuda o time a fazer sem trabalho sem interrupções ou problemas externos, seja de uma pessoa da equipe ou problemas de uma equipe externa, com isso a equipe ganha por estar mais focada sem interrupções, reduz os custos e encurta a resposta em suas demandas.

As equipes de Kanban precisam de um gerente de entrega de serviço para supervisionar a qualidade de sua entrega no serviço. Portanto, essa pessoa importante garante uma resposta adequada as demandas do cliente. O SDM normalmente assume as principais responsabilidades:

- Facilitar iniciativas de melhoria contínua.
- Verifique regularmente o quadro Kanban e certifique-se de que nenhum item de trabalho foi bloqueado.
- Executa reuniões Kanban regulares
- Controla todos os processos para trabalhar a satisfação do cliente.

Dirige à propriedade de cada tarefa, caso esta esteja atrasada

- Monitora e gerência os riscos ao projeto
- Certifica que as políticas sejam seguidas pela equipe.

Diferente do scrum, no Kanban, não há nenhuma regra rígida a ser seguida. Essa função pode ser preenchida por um membro da equipe ou desempenhada por toda a equipe.

8.8. Métricas

Com a metodologia Kanban, temos três principais métricas:

8.8.1. WIP – Work in Progress

WIP significa Work in Progress (trabalho em progresso). Ele representa a quantidade de tarefas que já foram iniciadas, mas ainda não foram concluídas, ou seja, toda tarefa que fica em fila.

Muito comum os projetos de desenvolvimento de softwares que utilizam Kanban, terem um wip limitado. Isso ocorre para estabelecer um máximo de tarefas iniciadas, mas não concluídas, em cada etapa do fluxo de trabalho. Com essa limitação é possível ver benefícios como:

- Melhor gerenciamento das tarefas;
- Maior foco na entrega;

- Trabalho entregue com maior qualidade;
- Minimizar ou evitar GAPS no fluxo;
- Evitar sobrecarga de serviço para o time;
- Localizar gargalos no processo;
- Melhora na Produtividade;
- Entregar Valor ao cliente com maior velocidade.

8.8.2. Lead Time

Lead Time é o prazo da entrega, esse conceito trata-se do período compreendido entre a data de realização de um pedido até o dia da entrega ao cliente. Essa medida permite que a liderança saiba a produtividade do time em tempo, muito utilizada para passagem de prazos, por exemplo, a cliente.

Calcular esse tempo de entrega é um dos fatores mais importantes para garantir a capacidade competitiva de uma empresa e ampliar suas possibilidades de expansão de mercado.

8.8.2.1. Importância

Dentro do Kanban, temos a mentalidade de aumentar a produtividade e diminuir o tempo, para obter os melhores resultados. Por conta disso, saber como ter um bom gerenciamento do lead time é importante porque possibilita a otimização dos processos de produção, sobretudo a diminuição do tempo gasto no desenvolvimento de features, desde a entrada do projeto até sua entrega.

Ter a noção do lead time dos projetos, contribui com uma flexibilidade do wip e com os planejamentos organizacionais mais eficientes, pois, com o tempo gasto calculado, os gestores permitem analisar e planejar medidas a serem adotadas para reduzir ainda mais esse período.

O cálculo é um longo processo, porém, após algumas semanas realizando o cálculo do lead time, é possível ter uma base de tempo.

8.8.2.2. Cálculo

Para que o cálculo seja feito de maneira eficiente e resultado assertivo algumas etapas em andamento do projeto precisam ser analisadas e relatadas. O primeiro passo é dividir todo o projeto em micro etapas iguais ou semelhantes para que se tenha uma visibilidade clara e fazer uma média de tempo de cada etapa.

No começo o processo pode parecer empírico já que a equipe não está acostumada a tais medidas e analises. Após as entregas alguns requisitos podem ter tempos variáveis, uma vez que podem ser de manutenção ou inovação e seu nível de complexidade variável. Passado um período, reúna essa análise e calcule o tempo médio de cada etapa.

O lead time é a soma de todas as etapas do projeto (Backlog, Design, Desenvolvimento, Teste e Deploy). Com o passar do tempo e a maturidade da equipe esse número tende a ficar mais real a produtividade da equipe

8.8.3. Throughput

Traduzido literalmente como vazão, o throughput é a métrica que mede a capacidade de entrega do seu time em um ciclo de tempo. Ou seja, com ela é possível analisar quantas tarefas ou histórias de usuário seu time consegue entregar dentro de um período. Com essa metrificação, é possível verificar qual a média de rendimento do seu time de tecnologia para novos produtos a partir dos projetos que já foram entregues ao cliente.

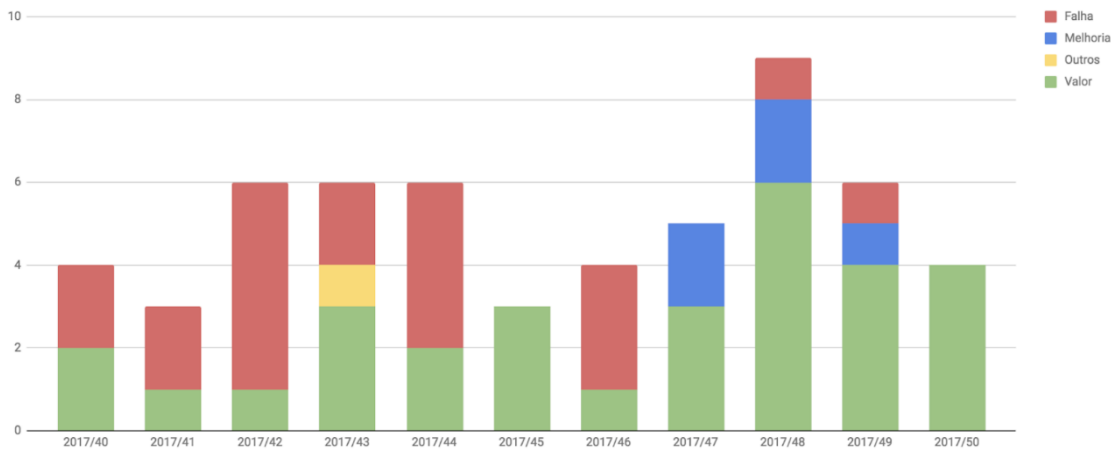

Figura 9 - Quadro de Troughput

8.8.3.1. Importância

Atuando em conjunto com o wip, é possível medir com clareza o tempo de produção de desenvolvimento dos itens, podendo servir, para reorganizar de uma maneira que seja mais produtiva e ágil para a equipe. Além disso com a Análise de dados é possível encontrar outros aspectos importantes como a confiabilidade do projeto e a eficiência da produção.

O throughput é importante também para analisar a categorização de atividades do seu Kanban, assim, é possível perceber onde o esforço da equipe está sendo investido, seja em desenvolvimento de manutenção ou de melhorias para o produto

8.8.4. Throughput vs Lead Time

Ambos são métricos de análise de fluxo, porém, para coisas diferentes, o throughput tem o foco em saber qual o esforço do time com foco nas tarefas, se a maior parte das tarefas entregues aos clientes foi uma manutenção, uma inovação ou uma documentação. Já o lead time, ela vê principalmente as tarefas de manutenção quanto tempo ela demora no fluxo, desde sua entrada na área do suporte até a entrega efetiva da solução para o cliente.

9. SCRUM

A ferramenta Scrum é uma das mais comuns na área de tecnologia, pois muito comum os projetos de tecnologia utilizarem projetos em etapas, resolvendo demandas, delegando responsáveis e determinando os seus respectivos prazos de maneira organizada.

Scrum, é um framework com solução voltada para gerenciamento de projetos onde, profissionais conseguem trabalhar juntos na resolução de problemas complexos de forma criativa, adaptável e garantindo resultados mais eficientes e de alto valor,

O Scrum foi desenvolvido por Jeff Sutherland, um piloto da aeronáutica, onde, foi criado pois cada decolagem em um pouso nunca será igual, não havendo uma fórmula. Com isso, em cada operação os pilotos devem fazer ajustes de acordo com as necessidades apontadas.

9.1. Objetivo

A metodologia Scrum Agile, mostra que o projeto deve ser dividido em pequenos ciclos de atividades, que incluem reuniões de alinhamento. Em cima disso, as tarefas de desenvolvimentos são geradas, porém, durante o processo evolutivo do projeto, alguns ajustes fora do escopo levantado nas reuniões, serão realizados para chegar no resultado, sempre de forma ágil e sem padronizações ou ações engessadas. Com o objetivo de buscar uma solução mais rápida e eficiente.

9.2. Papeis

Para que a técnica seja aplicada corretamente, o Scrum conta com papeis bem definidos. Com o tamanho do projeto e ou complexidade, é possível adicionar outras funções ou mesmo reduzir algumas.

9.2.1. Product Owner

Product Owner, pode ser traduzido com dono do produto, é o maior interessado no desenvolvimento do produto. Ele é responsável por definir as funcionalidades esperadas pelo produto, ordem de prioridade e suas expectativas.

9.2.2. Scrum Master

O Scrum Master exerce uma função de liderança importante. Ele coordena as atividades e atua como facilitador e implantador da metodologia para a equipe. Quaisquer dificuldades encontradas no projeto ele irá buscas a solução ideal. Por mais que possua uma função de liderança, ele não atua a nível de autoridade ligado ao projeto final e sim a metodologia e como ela é aplicada.

9.2.3. Time Scrum

O time é composto por desenvolvedores e responsáveis pelas etapas, como tester. Normalmente, reúnem habilidades especificas para concluir as tarefas definidas para atingir as expectativas estabelecidas pelo product owner.

9.3. Execução em Desenvolvimento de Software

Para que o framework possa ser executado da melhor maneira, é essencial respeitar algumas atividades especificas. O Scrum segue o Scrum Guide, onde é definido regras para execução, nela podemos perceber que são regras flexíveis podendo se organizar de acordo com a complexidade do projeto.

O scrum mesmo sendo composto por regras, a implementação nos times de desenvolvimento de software pode se adaptar facilmente. Os pilares do scrum, transparência, inspeção e adaptação, podem ser um ponto de atenção a usabilidade nos primeiros meses

9.3.1. Product Backlog

O product backlog serve como uma lista com todas as funcionalidades e características desejadas do produto. É definida de acordo com prioridades, em que as entregas mais importantes são feitas logo no início.

Nesta etapa é organizado com a criticidade de uma tarefa, onde funcionalidades mais importantes e essenciais são executadas no início para que possa fluir a execução do projeto.

9.3.2. Sprints

O product backlog, normalmente, apresenta muitos passos e exigências, conforme o projeto vai evoluindo. Para garantir o cumprimento, existe os chamados sprints. Eles são pacotes de desenvolvimentos, com tempo definido. Em geral, duram entre 2 a 4 semanas e o ideal é que durante todo o projeto seja mantida esse tempo, para acompanhamento de métricas, velocidade de desenvolvimento e evolução da equipe.

É essencial que, ao final de cada sprint, haja uma entrega de valor para o cliente. Nesta etapa não precisa ser algo muito grande, mas é necessário ter um valor para o cliente.

9.3.3. Sprint Planning

Após o encerramento completo de uma sprint, é preciso planejar como será a próxima. A reunião dura em média entre 2 horas a 4 horas, e nela é discutido tudo que foi feito e nos futuros desenvolvimentos pelas próximas semanas bem como a maneira de agir.

O scrum master tem como papel fundamental nessa etapa, ele coordena esse planejamento marcando as reuniões, levando as métricas de destaques e de e as fragilidades encontradas nesse período.

9.3.4. Daily Scrum

Diariamente e, de preferência, no mesmo horário, deve ser feita uma reunião. Conhecido como daily scrum, dura, no máximo, 15 minutos. Ela serve para identificar o andamento das tarefas e o alcance da meta da Sprint específico, durante a reunião é necessário responder a três questões:

- O que eu fiz ontem que me ajudou no alcance das metas?
- O que vou fazer hoje para ajudar a completar a Sprint?
- Quais são os impedimentos ou dificuldades para a tarefa?

9.3.5. Definition of Done

Diferentemente do Kanban, a definição de pronto, é um documento que defini se uma sprint pode ser finalizado. É essencial que este documento seja elaborado com ajuda de toda a equipe, é recomendado, ainda que, o formato deste documenta seje em checklist.

9.3.6. Sprint Review

A revisão da sprint ou sprint review ocorre no encerramento da sprint. Ela marca a entrega oficial, com apresentação para os clientes das funcionalidades mais relevantes desenvolvidas no período.

Na reunião é possível fazer no formato de apresentação onde o time de desenvolvimento apresenta para o cliente mostrando suas funcionalidades apenas ou com a participação do cliente onde ele passa um feedback.

9.3.7. Sprint Retorspective

Já na retrospectiva da Sprint tem um funcionamento parecido com o da revisão, a diferença ocorre, pois, é uma reunião focada no desenvolvimento da equipe no processo de entrega. Normalmente,

nessas reuniões são mostrados os impedimentos que tiveram durante o ciclo e é discutido uma forma de na próxima evitar ou contornar tal ação.

9.4. Vantagens

Essa ferramenta aumenta consideravelmente a produtividade e as chances de sucesso da sua equipe no desenvolvimento de projetos. Ela ajuda a organizar os processos, o que é imprescindível para identificar e eliminar eventuais gargalos, o que faz com que possíveis problemas sejam eliminados antes que tornem as situações irreversíveis.

9.4.1. Transparência

Como os ciclos de atividades envolvem diferentes profissionais, a transparência das informações dos projetos é um de seus pilares, ou seja, para que os resultados esperados sejam atingidos, é preciso que os colaboradores envolvidos em um projeto conheçam todos os elementos que podem afetar seu andamento.

9.4.2. Adaptabilidade

Como ferramenta, o scrum é flexível e não exige que uma linha única de ações seja tomada. Elas devem ser adaptadas conforme a necessidade que o problema apontar. Isso traz mais leveza para sua aplicação, o que, por consequência, garante mais agilidade nas resoluções.

9.4.3. Feedback Continuo

O feedback contínuo está totalmente alinhado com o princípio da transparência do scrum. Nele, a ideia é apontar quais atividades são desempenhadas corretamente e quais podem ser melhoradas.

Seu foco, no entanto, não está relacionado a motivação e gestão de pessoas, e sim com a manutenção da transparência das informações.

Como parte de um projeto, um profissional precisa entender o impacto de suas contribuições. Para isso, as reuniões regulares, ou sprint reviews, são determinantes.

9.4.4. Melhoria Continuo

Tomando como base as correções recorrentes e o aprendizado a partir da observação sobre elas, a tendencia é de que os processos e profissionais envolvidos estejam em constante aprimoramento de seus conhecimentos e qualidade. Isso é percebido tanto nos resultados práticos quanto no mindset da equipe, que passara a valorizar o método e que ele proporciona para as rotinas de trabalho.

9.4.5. Eficiência

O scrum tem algumas diretrizes que buscam moldar o processo com eficiência. Dessa forma, apesar de flexíveis e adaptáveis ao modelo da empresa, o limite de tempo para as reuniões diárias, formatos como os conteúdos devem ser compartilhados e outras padronizações garantem a eficiência do processo no longo prazo de sua utilização.

9.4.6. Motivação

Diferentemente do que é comum, ver nas empresas, é motivador trazer transparência para as informações dos projetos e dar autonomia aos colaboradores. Como as reuniões diárias de conscientização de toda a equipe sobre o andamento dos projetos são objetivas quanto as informações, o scrum e seus processos garantem um ambiente e condições de trabalho engajadoras. Isso demonstra que a empresa e seu gestor têm confiança em seus funcionários e suas habilidades para resolução dos problemas apontados.

9.5. Principais Erros

Com o aumento da popularidade do scrum, é normal algumas pessoas quererem implementar em seus projetos de desenvolvimentos de software, porém, temos que tomar cuidado com algumas peculiaridades, uma delas é a empresa não se adaptar e reclamar que os métodos não funcionam. Geralmente, isso acontece devido aos erros relacionados ao uso da ferramenta.

9.5.1. Não Combater a Resistencia Organizacional

Assim como a maioria das novidades, a adoção do scrum pode enfrentar certa resistência organizacional no começo. Essa situação pode acontecer em função da falta de conhecimento sobre a importância do Scrum em todos os níveis da companhia, o que pode ser um reflexo do receio de mudar. Quando uma organização passa a ter uma mentalidade ágil, apresentando equipes auto-organizadas, com maior autonomia e uma estrutura organizacional horizontal, os gestores podem pensar que se tornarão irrelevantes para o ambiente de trabalho.

A Atuação do Scrum master é de grande utilidade para situações como essa. Isso porque, as empresas mais bem-sucedidas no uso da ferramenta são aquelas que conseguem gerenciar e eliminar o clima de medo e incertezas entre os líderes. Para isso, é necessário repensar as funções e responsabilidades dos gerentes no processo de desenvolvimento das atividades.

9.5.2. Não Considerar os Princípios Ágeis

Muitas vezes, os profissionais que estão aplicando o scrum focam em tarefas simples, como a realização de reuniões todos os dias, o uso de ferramentas certas, e o preenchimento dos papeis da plataforma. Isso é importante, mas é apenas uma pequena parte do trabalho.

Seguir os princípios ágeis é essencial para o bom funcionamento dele. Devido ao descumprimento ou a má execução pode causar um desinteresse da empresa para investimento.

9.5.3. Não Documentar

Uma etapa muito importante do scrum é a documentação, dos requisitos, processos da metodologia e etapas dos desenvolvimentos do software.

Além disso, muitas pessoas optam por uma documentação rasa para que o excesso de detalhes não acabe atrapalhando a criatividade dos colaboradores responsáveis por executar as atividades e para otimizar o tempo, uma vez que por meio do scrum se está trabalhando com ciclos menores.

Entretanto, a documentação é essencial. O mais indicado é estabelecer uma boa comunicação com os integrantes do seu time para definir quais requisitos devem ser documentados em uma sprint. Com isso, todos poderão indicar quais são as informações necessárias para que as tarefas sejam desenvolvidas.

O product owner deve priorizar e organizar o backlog continuamente e na etapa de refinamento precisa contar todos os detalhes, podendo fazer possíveis mudanças até chegar ao sprint planning.A partir desses detalhes, a equipe pode entender melhor a história no decorrer doo sprint planning. Para tanto, pode-se usar vídeos, áudios ou explicações. Também é nessa etapa que o Dev Team estima as histórias dos usuários e, se já tiver métricas de velocidade, poderá determinar quantas histórias serão possíveis atender no sprint.

10. SCRUMBAN

O termo Scrumban foi primeiramente mencionado por Corey Ladas, um praticante de Lean-Kanban que por meio de diversos estudos criou um método de migrar do Scrum para o Kanban. Apesar de novo, os estudos de Ladas levaram a crer que a adoção da mescla seria muito benéfica para os já praticantes do framework Scrum, pois preenchiam lagunas deixadas em aberta.

À medida que mais pessoas se interessam pelas ideias Lean e sua aplicação ao trabalho do conhecimento e gerenciamento de projetos, é útil encontrar maneiras que tornem mais fácil começar ou aprender alguns conceitos básicos que podem levar a insights mais profundos posteriormente. De uma forma ou de outra, os usuários Scrum são um importante constituinte do público Kanban. Visto que o Scrum pode ser descrito como uma declaração na linguagem que usamos para descrever os sistemas Kanban, também é bastante fácil elaborar esse caso para descrever os híbridos Scrum / Kanban. Isso pode ser útil para times Scrum existentes que procuram melhorar sua escala ou capacidade. Também pode ser útil para novos usuários mais cautelosos que encontram conforto em um método “estabelecido”. (Apesar do fato de que a ideia Kanban é pelo menos 40 anos mais velha.)

O Scrumban, como o híbrido de dois métodos ágeis, fornece às equipes de desenvolvimento de software a flexibilidade para se adaptar e alterar as partes interessadas e os requisitos de produção sem sobrecarga. Esta é uma abordagem versátil para gerenciamento de fluxo de trabalho, pois fornece a estrutura Scrum completa com a visualização e flexibilidade de Kanban. Você pode facilmente aplicar o Scrumban Agile para passar do Scrum para o Kanban sem dor.

10.1. Scrumban versus Scrum

O Scrumban tira do Scrum tal tomada de decisão, como descobrir quanto trabalho pode ser feito em um sprint e priorizar qual é a tarefa mais importante a ser trabalhada em seguida.

Mas esse trabalho não é feito até que a análise necessária seja concluída, que se enquadra na definição do Scrum de pronto. Essa lista pronta é usada como uma ponte para organizar tarefas entre o backlog do produto e o estágio de execução.

10.2. Scrumban versus Kanban

Kanban entra em Scrumban para melhorar o processo de gerenciamento de projeto e visualizar o fluxo de trabalho. Primeiro, o Scrumban usa painéis Kanban, que geralmente são chamados de painéis Scrumban quando usados em uma metodologia Scrumban.

As equipes Scrumban também usam processos Kanban, como o sistema pull, que fornece um fluxo de trabalho contínuo. Ou seja, as tarefas são puxadas para a coluna de execução quando a equipe está pronta para executar.

O método Kanban também ajuda o Scrumban, limitando quantos itens estão em andamento a qualquer momento, o que aumenta o foco em tarefas específicas e ajuda na produtividade. Ao contrário do scrum, no Kanban as funções individuais não são claramente definidas, portanto, isso também adiciona alguma flexibilidade.

A análise just-in-time é herdada do Kanban, o que cria lead times mais curtos, em vez do processamento em lote para estimativas de planejamento de iteração, que é usado no Scrum. Além disso, as fraquezas nos processos podem ser expostas por meio do uso de buffers de processo e diagramas de fluxo . Essas ferramentas Scrumban ajudam a identificar áreas que podem ser melhoradas e reduzir gargalos.

10.3. Scrum Guide

Usar Kanban junto ao Scrum, vai trazer mais clareza de como aplicar na prática conceitos do Scrum guide e não os substituir.

Falando de papeis, mantemos o fluxo do Scrum, sendo que o Scrum master vai ter de aprender algumas coisas que são responsabilidades do Service Delivery Manager e o product owner, líder do produto, tem de estudar um pouco sobre o papel de Service Request Manager, ambos papeis do Kanban.

Como o Kanban é um método mais metrificado que o Scrum, você terá mais métricas e a presença de quadros para insumos nas cerimonias propostas no Scrum, como reuniões diárias (daily), cerimonias de planning e retrospectiva.

A daily passara a ser utilizado um quadro físico ou um software para acompanhar as tarefas, verificando tarefa por tarefa comentado sua dificuldade e impedimentos.

Planning, usara das técnicas de métricas e previsibilidade obtidas pelo SLE (Service Level Expectations – Expectativas de Nivel de Serviço) para incrementar sua assertividade.

A review terá um momento para ver as métricas do time, com como a retrospectiva pode usar as mesmas para iniciar discussões.

10.4. Fluxo de Valor

O coração do método Kanban é o fluxo de valor, as vezes também é chamado de fluxo de trabalho. Todo o time Scrumban, deve ter mapeado e definido o seu fluxo de valor, as etapas necessárias para um Product Backlog Item ser entregue como DONE segundo sua definição de pronto.

- Além do famoso quadro de atividades, que ajuda a definir esse fluxo, é importante que o time documento e torne explicito, a definição de fluxo valor do time deve ter:

- As etapas que o fluxo de valor do time possui, ao menos considerando espera (TO DO), execução (DOING) e finalizado (DONE).
- Regras claras de quando um Product Backlog Item pode ser movido de uma etapa para outra. Qual será a unidade de medida de valor entregue ao usuário. Enquanto em Scrum o mais comum seja utilizar Story Points, o mais comum em Kanban é apenas contar o número de Product Backlog Itens finalizados (DONE).
- Limitar o trabalho em progresso (WIP) durante a sprint.Um conjunto de métricas chamado de SLE que serve para ter previsibilidade e transparência do trabalho de time.

10.5. WIP – Work in Progress

Uma prática de times Scrumban é possuir o limite de trabalho em progresso. Nas semanas anteriores ao adotar a prática conjunta, é realizado um estudo para prever uma mediana de quantos itens do backlog o time conseguiu entregar, esse número é a vazão estimada (Throughput) e define o wip limit total dessa sprint. Para novos itens entrarem, os itens já pertencentes a sprint precisam ser finalizados ou de priorizados. Conforme novas sprints acontecem o time poderá aumentar ou diminuir o WIP já com ele em prática.

O Kanban sugere que para aplicar o WIIP por etapas do fluxo de trabalho. Essas restrições mais granulares ajudam a educar e disciplinar o time de desenvolvimento, garantindo melhor foco, objetividade e entregas completas.

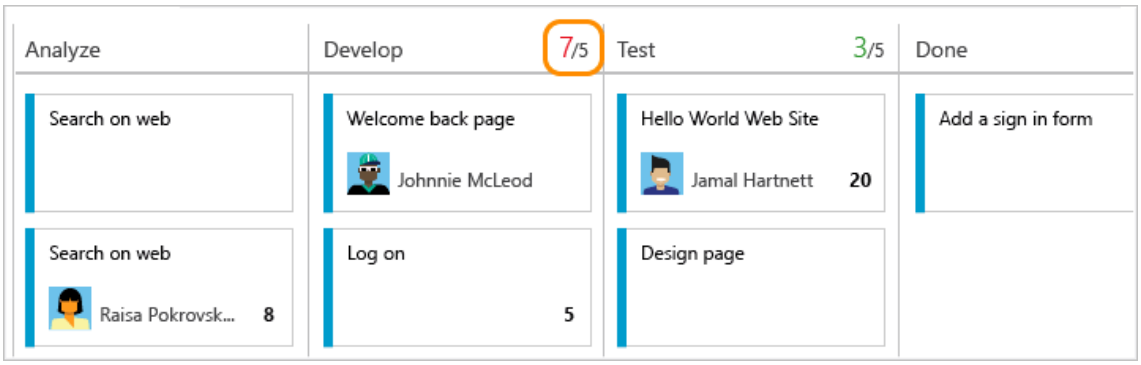

Figura 10 - Quadro Kanban com WIP

10.6. SLE - Service Level Expectations

O SLE é um dashboard de gráficos e números que o time Scrumban deve ter anexo ao quadro de atividades. A base para as métricas é o cycle time, ou seja, tempo decorrido entre início e fim do desenvolvimento de um backlog. O primeiro passo é anotas em cada PBI o dia que o time começa a desenvolver até o término.

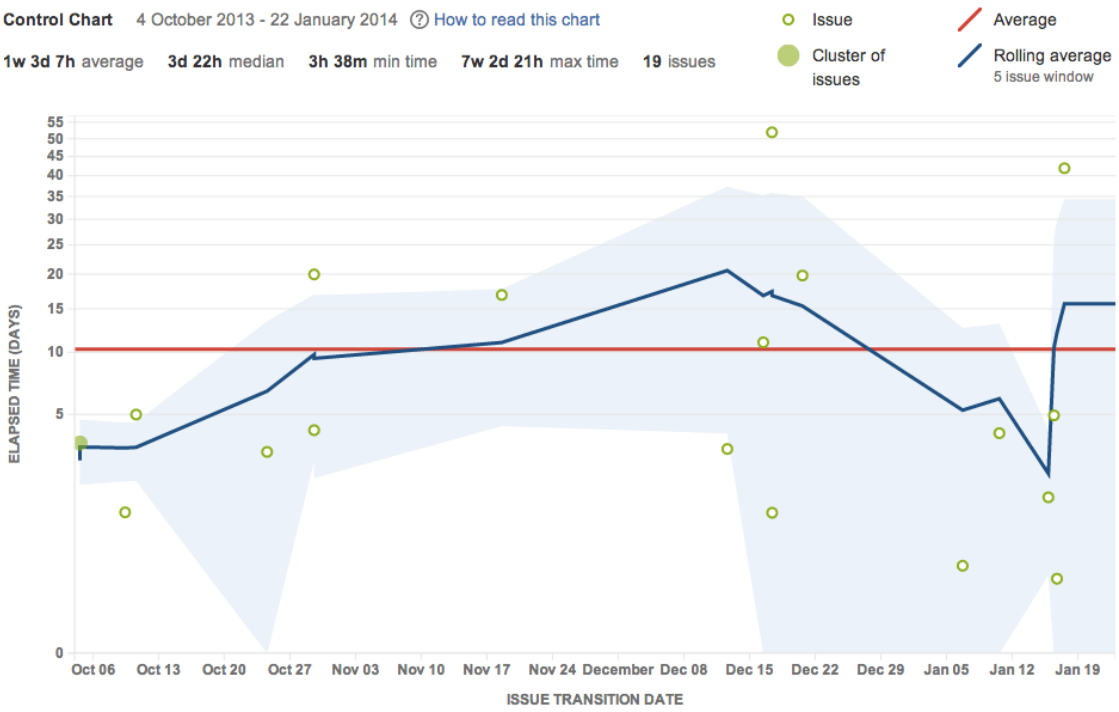

Figura 11 – Gráfico SLE

Outro elemento importante é registrar o Throughput do time. Esse número serve como um limitador de tarefas em progresso (WIP).

Weeks	1	2	3	4
Cards Delivered	10	8	12	10

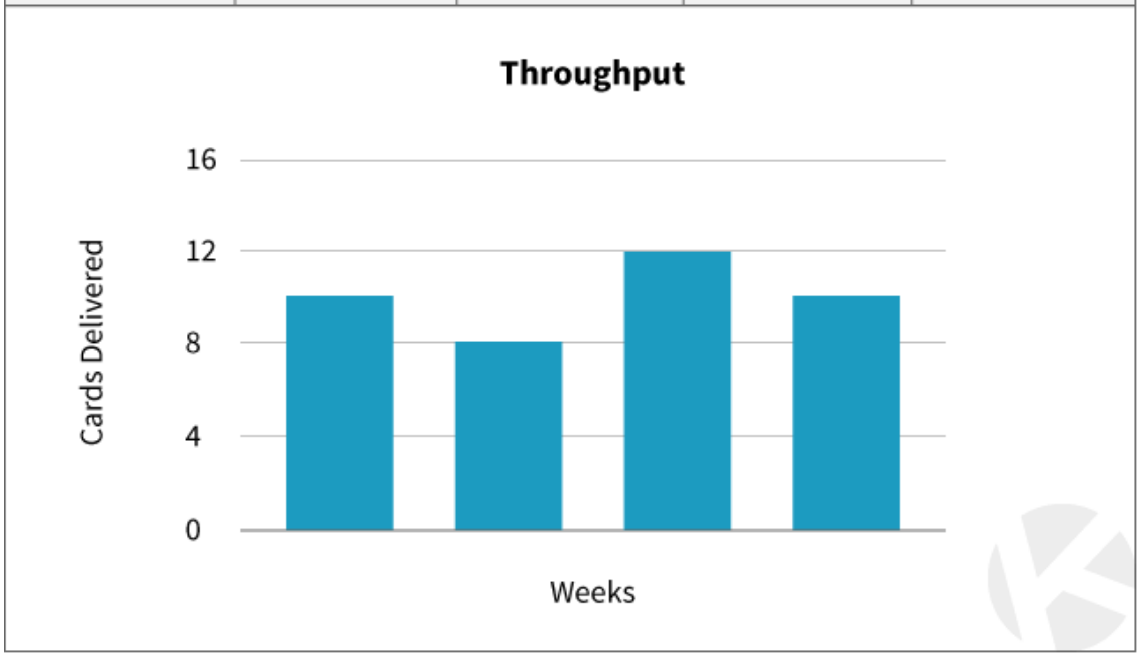

Figura 12 – Throughput

Burndown Chart, é um gráfico que mostra o fluxo de cada etapa do desenvolvimento, mostrando quanto tempo cada uma demorou e onde estão os principais gargalos do time.

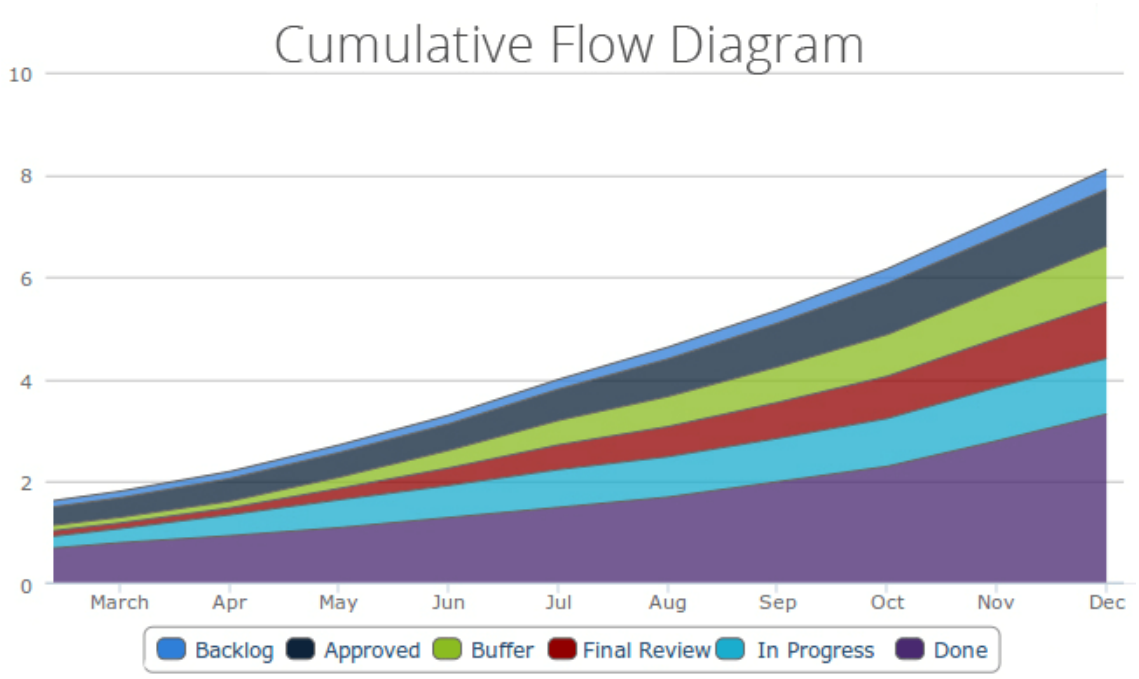

Figura 13 – BurnDown

10.7. Vantagens da Metodologia Scrumban

Scrumban possui muitos dos pontos fortes e poucos dos pontos fracos que atormentam seus predecessores, Scrum e Kanban. Mas sua flexibilidade pode ser intimidante para equipes novas no Ágil e sem um coach ou defensor interno. Se a estrutura e a certeza ajudarem você a começar, a adaptabilidade do Scrumban pode ser um prejuízo. Para que a metodologia consiga fluir sem medo, é recomentado que a equipe já trabalhe em uma das outras metodologias, pois a familiarização com os novos processos pode ser mais rápida e menos tenebrosa. Como citado, o Scrumban tem pontos fortes de ambas as metodologias, por isso é recomendado quando já se tem uma maturidade da equipe e da empresa.

10.7.1. Economia de Tempo

Uma das vantagens do processo Scrumban é que ele economiza tempo. Isso porque não há planejamento de sprint a cada duas semanas. Os planos só são feitos quando há uma demanda da equipe para fazê-los, como quando o trabalho em andamento fica abaixo de um limite predefinido.

10.7.2. Compartimentalização

Projetos maiores também são ideais para a metodologia Scrumban. Quanto maior o projeto, mais recursos e tarefas associados a ele. Essas entregas são necessárias ao longo de meses, senão anos. O Scrumban pode ser distribuído em vários intervalos de tempo e priorizado em iterações mais curtas para melhor gerenciar esses projetos de longo prazo.

10.7.3. Detectar Obstáculos com Scrumbran

Os gargalos são a ruína dos projetos. Eles atrasam o trabalho, bagunçam os horários e perdem tempo e dinheiro. Um quadro Scrumban é uma ótima maneira de encontrar esses gargalos no fluxo de trabalho e resolvê-los antes que se tornem um problema. Como os quadros Kanban, um quadro Scrumban permite que os gerentes de projeto vejam onde estão a maioria das tarefas e resolvam a desaceleração com antecedência e eficácia.

10.7.4. Clareza

Todos estão na mesma página no Scrumban. Novamente, por causa da transparência dos quadros Kanban, todos os membros da equipe podem ver onde eles e o projeto estão em termos de fluxo de trabalho. Eles atualizam seus status e todos os veem.

11. CONCLUSÃO

Neste trabalho pôde-se mostrar como a evolução da metodologia ágil impacta diretamente nas áreas de desenvolvimento de software. Nota-se que as metodologias Kanban e Scrum conseguem trabalhar de forma satisfatória separadamente, porém um novo patamar é alcançado ao ser utilizadas juntas.

O Scrum, uma das metodologias mais utilizadas no desenvolvimento de software, mostra pequenas fraquezas quando olhamos o fluxo de trabalho, enquanto o Kanban peca principalmente na parte de documentação de processos, como o DOD.

Quando as empresas percebem que precisam mudar para uma forma mais enxuta e ágil de trabalhar, o Scrumban é uma maneira fácil de eliminar qualquer empecilho que acaba surgindo eventualmente, além de permitir que as empresas continuem a trabalhar de maneira mais eficiente, mas ao mesmo tempo implementem as melhores práticas enxutas e ágeis específicas para o ambiente de *streaming*.

O Scrumban pode fornecer o nível apropriado de rigor de que a empresa precisa, bem como a flexibilidade, eficiência e visibilidade de Kanban e Lean.

12. REFERENCIAS BIBLIOGRÁFICAS

The Agile Manifesto for Software Development. Disponível em: https://www.agilealliance.org/agile101/the-agile-manifesto/. Acesso em: 10 set. 2021.

What is Scrumban. Disponível em: https://kanbantool.com/kanban-guide/what-is-scrumban Acesso em: 15 set. 2021.

What is Scrumban? How It Differs from Scrum & Kanban. Disponível em: https://www.projectmanager.com/blog/what-is-scrumban Acesso em: 15 set. 2021.

Scrumban. Disponível em: https://www.productplan.com/glossary/scrumban/ Acesso em: 15 set. 2021.

What is Scrumban. Disponível em: **What is Scrumban**. Disponível em: https://kanbantool.com/kanban-guide/what-is-scrumban Acesso em: 15 set. 2021.

What is Scrumban? How It Differs from Scrum & Kanban. Disponível em: https://www.projectmanager.com/blog/what-is-scrumban Acesso em: 15 set. 2021.

Funcionamento do Scrumban. Disponível em: https://www.productplan.com/glossary/scrumban/ Acesso em: 15 set. 2021.

Acesso em: 19 set. 2021.

Entenda o funcionamento do Scrum. Disponível em: https://rockcontent.com/br/blog/scrum/ Acesso em: 19 set. 2021.

Kanban. Disponível em: https://www.totvs.com/blog/negocios/kanban/ Acesso em: 20 set. 2021.

Metodologia Agil. Disponível em: https://www.totvs.com/blog/negocios/metodologia-agil/ Acesso em: 19 set. 2021.

Lead Time. Disponível em: https://ezdevs.com.br/lead-time-saiba-a-importancia-de-reduzi-lo/ Acesso em: 21 set. 2021.

Funcionamento do Scrumban. Disponível em: https://www.productplan.com/glossary/scrumban/ Acesso em: 21 set. 2021.

Acesso em: 20 set. 2021.

Entenda o funcionamento do Scrum. Disponível em: https://rockcontent.com/br/blog/scrum/ Acesso em: 22 set. 2021.

Scrum ou Kanban? Disponível em: https://www.luiztools.com.br/post/o-que-e-melhor-scrum-ou-kanban/ Acesso em: 22 set. 2021.

A Arte de Fazer o Dobro do Trabalho na Metade do Tempo. SUTHERLAND, Jeff. Scrum. 2ª Ed, Leya, 2016.

Throughput. Disponível em: https://ezdevs.com.br/throughput-o-uso-da-metrica-no-gerenciamento-de-fluxo-do-seu-projeto/ Acesso em: 30 set. 2021.