Universidade do Vale do Itajaí

Modelagem de Processo de Software

MELISSA MOREIRA DE OLIVEIRA

ERICK LEMOS BARRETO

ANA CLARA KNISS

MARIANA FERREIRA

VICTOR HUGO CHRISOSTHEMOS TEIXEIRA

São José

Santa Catarina

2024

CONTEÚDO

1 DETALHAMENTO DAS FASES DO PROCESSO

Para definir o ciclo de vida de desenvolvimento de software será considerado ciclos iterativos e incrementais baseados na metodologia Scrum. Nesse sentido, é iterativo porque define a repetição de uma ação, no caso desenvolver, que gera artefatos superiores a iteração anterior (incrementais).

Portanto, os processos serão definidos em torno de sprints com tempo de 1 entre 3 semanas que vão definir um conjunto de funcionalidades que devem ser desenvolvidas. A ideia dessa segmentação terá como um denominador comum a entrega frequente de valor, a colaboração entre o time e os stakeholders, a adaptação e feedbacks

1.1 Fases

As fases para a dinâmica Scrum selecionada terá os seguintes aspectos.

1.1.1 Planejamento

A ideia norteadora dessa fase é desenvolver uma base para o projeto, com considerações sobre modelagem do negócio com o uso de BPMN, definição de um público-alvo, matriz SWOD, análise de concorrência, demanda de mercado, estratégia de produto para definir seu diferencial e criação de um catálogo de produtos.

A partir desses levantamentos, será definido a visão do produto pelo Product Owner, com os objetivos e valores esperados. Consolidado isso, será possível criar uma lista de funcionalidades que irá compor o Product Backlog, esse documento deverá ter prioridades de tarefas e possíveis melhorias, sendo constantemente revisitado e modificado.

Posteriormente, é criado o time Scrum, que irá desenvolver o sistema, é definido o Scrum Master, o responsável pela garantia que a equipe siga os princípios Scrum. Com isso, temos o Product Owner, o Scrum Master e o Time Scrum.

1.1.2 Desenvolvimento

O desenvolvimento será feito através das sprints. No início de cada sprint, será feito uma reunião de entender como será realizado os procedimentos necessários.

Nesse contexto, o Product Owner definir o que é prioritário no Product Backlog. Enfim, o Sprint Goal vai resumir o objetivo de entrega do ciclo. Lembrando que, sempre haverá Daily Scrum, um período de 15 minutos para revisar o progresso e definir o que será feito no dia.

No final de cada sprint, o time faz uma revisão e demonstra o as tarefas realizadas, há um feedback do Product Owner e possivelmente haverá ajustes no Backlog. Após, é feito uma retrospectiva para discutir pontos assertivos e pontos com possibilidades de melhorias em relação ao processo do desenvolvimento, contribuindo para melhorias e maior colaboração.

1.1.3 Entrega

A entrega será incremental e contínua, ou seja, a cada final de sprint, parte de projeto já estará com componentes funcionais e, com base no feedback e nas mudanças de requisitos, a adaptação vai moldando o desenvolvimento.

1.2 Objetivo

O objetivo do desenvolvimento é entregar valor contínuo ao cliente através de incrementos funcionais ao projeto, adaptação rápida às mudanças em parâmetros do negócio, colaboração entre as partes envolvidas e transparência no cotidiano em abranger as necessidades das partes envolvidas.

1.3 Atividades

Em relação às atividades, elas podem ser elencadas considerando as fases em que estão.

1.3.1 Planejamento

- Refinamento do backlog
- Planejamento da sprint

1.3.2 Desenvolvimento

- Criação das funcionalidades

- Scrum daily
- Entregas contínuas
- Revisão do sprint
- Retrospectiva do sprint

1.3.3 Entrega

- Suporte
- Gerenciamento do backlog
- Feedbacks dos stakeholders

.

1.4 Produtos de Entrada

Os produtos de entrada são artefatos, documentos, ou recursos necessários para iniciar uma fase específica do processo de desenvolvimento de software. Esses produtos fornecem as informações e bases necessárias para que a equipe possa começar a trabalhar na fase seguinte.

Exemplo: Visão geral do projeto, necessidades iniciais do cliente.

1.5 Produtos de Saída

Os produtos de saída são os resultados obtidos ao final de uma fase de processo. Esses produtos servem como base para as próximas fases e, em muitos casos, são utilizados como entrada para a fase seguinte.

Exemplo: Documentacão de Requisitos de Software (SRS).

1.6 Critérios de Entrada

Os critérios de entrada são as condições que devem ser atendidas antes de iniciar uma nova fase do processo de software. Esses critérios garantem que a fase anterior foi concluída com sucesso e que a equipe possui todas as informações e recursos necessários para continuar.

Exemplo: Reunião inicial com o cliente realizada, compreensão básica do escopo do projeto.

1.7 Critérios de Saída

Os critérios de saída são as condições que devem ser atendidas para considerar uma fase concluída com sucesso. Esses critérios asseguram que todos os aspectos essenciais da fase foram cumpridos antes de passar para próxima etapa do processo.

Exemplo: Requisitos aprovados pelo cliente e equipe de desenvolvimento.

2 CRIAÇÃO DE TEMPLATES

A criação de templates é útil para tarefas repetitivas, gera processos organizados e mantém a equipe consistente, eficiente e com clareza de suas atividades. Com a padronização proposta por templates, também promove o monitoramento do processo, contribuindo para a redução de erros e melhoria contínua estimada na entrega de valor.

2.1 Ata de Reunião

- **Cabeçalho**: Data, Hora, Local, Participantes.
- **Assuntos para debate**: Itens do Backlog selecionados.
- **Decisões tomadas**: Resumo das decisões acordadas.
- **Ações pendentes**: Listar tarefas e seus respectivos responsáveis.

2.2 Detalhamento dos casos de uso

- **Nome do caso de uso**: Dar um nome e identificar o ator principal.
- **Descrição**: Descrever um objetivo do caso de uso e o seu valor relacionado ao sistema.
- **Definição de fluxos**: Fluxo principal e fluxos alternativos.
- **Critérios de aceitação**: Definir as condições específicas a serem atendidas.

2.3 Detalhamento de Requisitos e Regras de negócio

- **Identificador do Requisito**: Nome do requisito.
- **Descrição**: Descrever de maneira clara o que o requisito deve cumprir e seu objetivo no sistema.
- **Regras de negócio**: Definir as regras de negócio.
- **Critérios de aceitação**: Deve-se definir as condições específicas a serem atendidas.

2.4 Prototipação das Telas

2.4.1 Planejamento

- **Definição de objetivos:** Definir quais os objetivos do protótipo, e o que deverá ser incluso para testes como, por exemplo, fluxo de navegação, usabilidade etc.
- **Definição de requisitos:** Analisar os requisitos a fim de que seja definido quais funcionalidades deverão ser implementadas.
- **Definição das ferramentas a serem usadas:** Definir quais ferramentas deverão ser usadas para criação dos protótipos como, por exemplo, Adobe XD, Balsamiq, Figma etc.

2.4.2 Criação dos protótipos

- **Prototipação de baixa fidelidade:** Cria-se essa prototipação para que ela sirva como um esboço simples, sendo rápido, para testar ideias e fazer a validação para definir se elas são funcionais e deverão ser incluídas ou não.
- **Prototipação de média fidelidade:** Essa prototipação é mais definida que a de baixa fidelidade, sendo criada para testar a interatividade com elementos da interface, fluxo do usuário, porém sem contar ainda com o detalhamento que será feito na prototipação de alta fidelidade.
- **Prototipação de alta fidelidade:** Essa prototipação servirá para mostrar como seria o produto funcionando, trazendo confiança para o produto, para isso precisa ser de certo modo interativo e demonstrar uma navegação similar ao que se espera do produto.

2.4.3 Testes

- **Refinação:** A fase de refinação deverá ocorrer de forma síncrona com as fases anteriores de criação de protótipo, pois após criação do protótipo inicial, o mesmo é avaliado para que possa ser refinado, processo esse que é feito novamente na criação do protótipo de alta fidelidade.
- **Revisão e Aprovação dos Stakeholders:** O protótipo deve ser apresentado aos stakeholders para sua avaliação, e caso se demonstre necessário, deve ser revisado fazendo os ajustes identificados.

2.5 Diagramas do Sistema

2.5.1 Planejamento

- **Definição de requisitos:** Coletar os dados relevantes para identificação dos requisitos, incluindo objetivo do sistema, quais funcionalidades deverá ter, e suas interações.
- **Escolha dos diagramas apropriados:** Escolher quais diagramas serão usados para representação do sistema.
- **Definição das ferramentas a serem usadas:** Escolher qual ferramenta será usada para a criação dos diagramas como, por exemplo, Visual Paradigm, Visio, Sparx Systems etc.

2.5.2 Criação dos diagramas

- **Identificar e criação elementos de cada diagrama:** Identificar os elementos pertinentes a cada tipo de diagrama como, por exemplo, no diagrama de casos de uso, será definido os atores e será feita uma representação básica do fluxo do sistema e de como serão realizadas as interações dos atores com o sistema. Após isso criar o diagrama. Isso deverá ser realizado para cada diagrama relevante ao projeto.

2.5.3 Documentação e entrega

- **Documentação dos diagramas:** Criar uma documentação para cada diagrama com explicações sua relevância, o porquê do uso desse diagrama, suas interações, fluxo, etc.
- **Revisão e Aprovação dos Stakeholders:** Os diagramas devem ser apresentados aos stakeholders para aprovação do fluxo, interações etc., para aprovação e possível revisão caso se demonstre necessária.

2.6 Lista de Defeitos de Design

2.6.1 Identificação dos defeitos

- **Análise do design:** Revisão do design buscando encontrar defeitos como inconsistências, decisões que afetem a funcionalidade etc.
- **Avaliação de Impacto:** Avaliação do impacto que o defeito encontrado pode causar ao projeto, considerando cronograma, custos, qualidade etc.
- **Classificação de Defeitos:** Classificação dos defeitos encontrados levando em consideração o impacto que pode causar, assim permitindo a priorização de defeitos de maior gravidade.
- **Identificação da causa do defeito:** Análise do defeito, buscando encontrar sua origem, o porquê ocorreu, e como evitar que defeitos similares ocorram novamente.

2.6.2 Correção dos defeitos

- **Desenvolvimento da solução:** Desenvolvimento da solução para o defeito encontrado, deve-se adotar métodos que previnam a ocorrência de outro defeito similar, deve-se também considerar outros defeitos que podem ocorrer em decorrência da solução criada.
- **Aplicação da solução:** Aplicação da solução criada, deve-se informar as partes interessadas que será feita a aplicação da solução para que seja feita de maneira controlada, e possa ser feito o monitoramento do impacto da correção.

2.6.3 Aprovação da correção

- **Aprovação dos stakeholders:** Deve-se comunicar as correções aos stakeholders para certificação de que as mudanças a serem realizadas estão de acordo com as expectativas, e garantir o alinhamento do projeto aos objetivos.

2.7 Lista de Defeitos de Implementação

2.7.1 Identificação dos defeitos

- **Teste de integração:** Realizar os testes de integração para identificar defeitos como incompatibilidades de interface, problemas de comunicação entre módulos e outros defeitos decorrentes do mau funcionamento da integração entre os componentes do sistema e os módulos.

- **Teste de unidade:** Realizar os testes de unidade para identificar defeitos ocorridos quando o comportamento difere do esperado.

2.7.2 Análise dos defeitos

- **Avaliação de Impacto:** Avaliação dos impactos dos defeitos encontrados sobre o sistema, o que é crucial para a classificação dos defeitos e a priorização das correções necessárias.

- **Classificação de Defeitos:** Classificação dos defeitos com base no impacto e na gravidade que eles possuem sobre o sistema, o que é crucial para a priorização adequada dos erros mais graves.

- **Identificação da causa do defeito:** Revisão do código para identificar a origem dos defeitos e avaliar as técnicas e métodos utilizados. Isso é essencial para evitar a recorrência dos mesmos erros que causaram o defeito.

2.7.3 Correção dos defeitos

- **Desenvolvimento de solução:** Após a análise dos defeitos, desenvolve-se a solução adequada, que pode incluir a modificação do código, a alteração da lógica do sistema, entre outros ajustes, com o objetivo de evitar a criação de novos defeitos.

- **Aplicação da solução:** A solução desenvolvida é aplicada ao código-fonte, e novos testes são realizados para verificar se o defeito foi corrigido e se a aplicação da solução não introduziu novos defeitos. Portanto, essa aplicação deve ser feita de forma controlada e monitorada.

2.7.4 Aprovação da correção

- **Aprovação dos stakeholders:** Após a correção, deve-se comunicar os stakeholders para que revisem e aprovem as mudanças. Isso garante que as alterações estejam alinhadas com os requisitos e objetivos do projeto.

2.8 Relatório de Testes Geral

- **Escopo dos testes:**
  - Definição do que será testado.
  - Ambiente de testes.
- **Critérios de aceitação:**
  - Critérios mínimos para que o sistema ou parte dele seja considerado aprovado.
  - Definição de métricas de sucesso, como número máximo de falhas permitidas, desempenho mínimo aceitável.
- **Procedimentos de teste:**
  - Objetivo do teste: O que se espera verificar.
  - Passos para execução: Sequência de ações para realizar o teste.
  - Resultados esperados: O que se espera que o sistema faça após a execução do teste.
- **Resultados dos testes:**
  - Registro dos resultados de cada caso de teste (sucesso ou falha).
  - Análise dos resultados obtidos, comparando-os com os resultados esperados.
  - Lista de defeitos identificados durante a execução dos testes.

2.9 Relatório de Testes de Casos de Uso

- **Escopo dos Testes:**
  - Definição dos casos de uso que serão testados.
  - Ambiente de testes específico para cada caso de uso.
- **Critérios de Aceitação:**
  - Critérios mínimos para a aprovação dos casos de uso testados.
  - Definição de métricas de sucesso específicas para cada caso de uso, como o comportamento esperado e a tolerância a falhas.
- **Procedimentos de Teste:**
  - Objetivo dos Testes: O que se espera verificar para cada caso de uso.
  - Passos para a Execução: Sequência detalhada de ações para testar cada caso de uso.
  - Resultados Esperados: Comportamento esperado do sistema após a execução de cada teste de caso de uso.
- **Resultados dos Testes:**
  - Registro dos resultados de cada teste de caso de uso (sucesso ou falha).
  - Análise dos resultados obtidos para cada caso de uso, comparando-os com os resultados esperados.
  - Lista de defeitos ou desvios identificados durante a execução dos testes de casos de uso.

2.10 Projeto do Banco de Dados

1. **Dicionário de dados:**

    - Representação visual do modelo de dados, mostrando as entidades e os relacionamentos entre elas através de um MER (Modelo Entidade Relacionamento).
    - Atributos de cada entidade e chaves primárias.
    - Lista de todas as colunas de uma tabela, descrição de seu significado, valores possíveis, quantidade típica de dados e constraints associadas.
    - Nomes de objetos dependentes, como stored procedures, triggers, views e funções, com suas respectivas funções, parâmetros e retornos.

2. **Planejamento da capacidade:**

    - Listar as necessidades de espaço de armazenamento, utilização de CPU, largura de banda e outros requisitos técnicos que possam impactar o banco de dados.

3. **Política de segurança:**

    - Descrição de procedimentos, responsabilidades e atribuições relacionadas à segurança das informações e ao controle de acesso.