Uma imagem com texto, Tipo de letra, captura de ecrã, documento

Descrição gerada automaticamente

1. O SDLC é o ciclo de vida de desenvolvimento de sistemas, sendo as etapas: Planeamento, Análise, Design, Implementação e Manutenção.
2. Planeamento: Definir claramente os objetivos do projeto e criar um plano de ação detalhado.

Análise: Compreender e documentar as necessidades e requisitos do sistema.

Design: Desenhar a arquitetura do sistema e detalhar como ele será construído.

Implementação: Construir o sistema de acordo com as especificações do design.

1. "processo de software" refere-se ao conjunto de atividades, métodos, práticas e transformações que são realizados para desenvolver e manter software. O processo de software é um framework que define como o desenvolvimento de software deve ser conduzido para assegurar que os produtos finais sejam de alta qualidade, entregues dentro do prazo e dentro do orçamento.

* Exemplos:
* **Modelo em Cascata (Waterfall):** Um processo linear e sequencial onde cada fase deve ser concluída antes de passar para a próxima.
* **Modelo Incremental:** O software é desenvolvido e entregue em pequenas partes funcionais.
* **Modelo Iterativo:** O desenvolvimento é realizado em ciclos, permitindo revisões e melhorias contínuas.
* **Scrum:** Um framework ágil que enfatiza entregas rápidas e iterativas com feedback contínuo.

**Diferenças Fundamentais**

1. **Foco:**
   * **Análise de Domínio:** Enfoca na compreensão do contexto e necessidades do negócio.
   * **Especificação de Software:** Enfoca na tradução dessas necessidades em requisitos técnicos e detalhados.
2. **Participantes:**
   * **Análise de Domínio:** Envolve stakeholders, usuários finais e analistas de negócios.
   * **Especificação de Software:** Envolve engenheiros de software, arquitetos de sistemas e analistas técnicos.
3. **Resultado:**
   * **Análise de Domínio:** Produz uma compreensão clara e modelada do domínio de aplicação.
   * **Especificação de Software:** Produz documentos técnicos detalhados que guiam a implementação do sistema.
4. **Metodologias e Ferramentas:**
   * **Análise de Domínio:** Utiliza entrevistas, observações, workshops, e técnicas de modelagem conceitual.
   * **Especificação de Software:** Utiliza documentos de requisitos, diagramas técnicos, e ferramentas de modelagem de dados e processos.
5. O analista de sistemas analisa a situação do negócio, identifica oportunidades de melhorias e projeta um sistema de informação para implementá-las.

O principal objetivo de um analista de sistemas não é criar um sistema “topo de gama”, mas criar valor para a organização.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

**1. Descrever a estrutura do UP/OpenUP (fases e objetivos; iterações)**

O Unified Process (UP) e sua variação Open Unified Process (OpenUP) são processos de desenvolvimento de software iterativos e incrementais que são divididos em várias fases principais:

**Estrutura do UP/OpenUP:**

* **Fase de Iniciação:**
  + **Objetivos:** Definir o escopo do projeto, estabelecer um caso de negócios, e identificar os principais riscos.
  + **Atividades:** Coleta de requisitos iniciais, criação de um esboço da arquitetura, planejamento de alto nível, e elaboração de um estudo de viabilidade.
* **Fase de Elaboração:**
  + **Objetivos:** Refinar os requisitos, mitigar os riscos principais, e desenvolver a arquitetura básica do sistema.
  + **Atividades:** Modelagem dos casos de uso detalhados, desenvolvimento de protótipos, refinamento da arquitetura, e planejamento detalhado das próximas fases.
* **Fase de Construção:**
  + **Objetivos:** Desenvolver a maior parte do software, construir os componentes e funcionalidades principais do sistema.
  + **Atividades:** Implementação de funcionalidades, testes unitários e de integração, documentação do sistema, e preparação para a entrega do produto.
* **Fase de Transição:**
  + **Objetivos:** Transferir o sistema para o ambiente de produção e garantir que os usuários estejam preparados para utilizá-lo.
  + **Atividades:** Testes de aceitação, correção de defeitos, treinamento de usuários, e implantação do sistema.

**2. O OpenUP pode ser considerado “método ágil”?**

Sim, o OpenUP pode ser considerado um método ágil porque incorpora princípios e práticas ágeis, como:

* **Iterações curtas e incrementais:** Desenvolvimento em ciclos curtos que permitem feedback constante e ajustes rápidos.
* **Foco na colaboração:** Envolvimento ativo dos stakeholders e comunicação contínua com a equipe.
* **Entrega contínua de valor:** Priorização de funcionalidades que entregam valor imediato aos usuários.

**3. Por que é que o UP se assume como “orientado por casos de utilização, focado na arquitetura, iterativo e incremental”?**

* **Orientado por casos de utilização:** Os requisitos são capturados e definidos principalmente através de casos de uso que descrevem as interações entre os usuários e o sistema.
* **Focado na arquitetura:** A ênfase está na criação de uma arquitetura robusta e flexível desde o início, o que facilita a adaptação e a evolução do sistema.
* **Iterativo e incremental:** O desenvolvimento é realizado em pequenos ciclos (iterações), onde o software é continuamente refinado e expandido com base no feedback recebido, permitindo ajustes frequentes e entregas parciais de funcionalidade.

**4. Identificar características distintas dos processos sequenciais, como a abordagem waterfall.**

* **Fases rígidas e sequenciais:** Cada fase do desenvolvimento (requisitos, design, implementação, testes, manutenção) deve ser concluída antes de passar para a próxima.
* **Pouca flexibilidade:** Alterações nos requisitos ou no design são difíceis e caras de implementar depois que as fases iniciais são concluídas.
* **Baixa adaptação a mudanças:** Não acomoda bem mudanças durante o ciclo de desenvolvimento, o que pode resultar em problemas se os requisitos mudarem.

**5. Identificar as práticas distintas dos métodos ágeis (o que há de novo no modelo de processo, comparando com a abordagem “tradicional”?).**

* **Iterações curtas:** Ciclos de desenvolvimento curtos e frequentes (sprints) permitem entregas rápidas e feedback contínuo.
* **Colaboração contínua:** Envolvimento constante dos stakeholders e comunicação direta e frequente entre todos os membros da equipe.
* **Adaptação a mudanças:** Flexibilidade para alterar requisitos e planos de acordo com o feedback recebido e as mudanças no ambiente de negócios.
* **Foco em entregas de valor:** Priorização de funcionalidades que agregam valor imediato aos usuários, com foco em entregar software funcional desde as primeiras iterações.

**6. Distinguir projetos (de desenvolvimento de software) sequenciais de projetos evolutivos.**

* **Projetos Sequenciais:** Seguem uma abordagem linear e faseada (ex.: waterfall), onde cada fase deve ser completada antes de passar para a próxima. É menos flexível e difícil de acomodar mudanças.
* **Projetos Evolutivos:** Seguem uma abordagem iterativa e incremental (ex.: UP, métodos ágeis), onde o software é desenvolvido em ciclos curtos com feedback constante e a capacidade de adaptar-se a mudanças de requisitos e prioridades.

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

**1. Modelos Funcionais**

**Objetivo:** Representar as funcionalidades do sistema e como elas interagem com os usuários ou outros sistemas.

**Principais Diagramas:**

* **Diagrama de Casos de Uso (Use Case Diagram):**
  + **Descrição:** Mostra as interações entre os atores (usuários ou outros sistemas) e o sistema, destacando as funcionalidades que o sistema oferece.
  + **Elementos:** Atores, casos de uso, e associações.
  + **Exemplo:** Um diagrama que mostra como um usuário pode realizar operações bancárias como "Depositar Dinheiro" e "Retirar Dinheiro".

**Características dos Modelos Funcionais:**

* Foco nas funções e serviços que o sistema deve fornecer.
* Facilita a comunicação com stakeholders, pois é de fácil compreensão.
* Ajuda na definição e validação dos requisitos funcionais do sistema.

**2. Modelos Estáticos**

**Objetivo:** Representar a estrutura estática do sistema, mostrando os componentes e suas relações.

**Principais Diagramas:**

* **Diagrama de Classes (Class Diagram):**
  + **Descrição:** Mostra as classes do sistema, seus atributos, métodos e os relacionamentos entre elas.
  + **Elementos:** Classes, atributos, métodos, associações, generalizações, e dependências.
  + **Exemplo:** Um diagrama que mostra classes como "Conta Bancária", "Cliente", e "Transação", e como essas classes estão relacionadas.
* **Diagrama de Objetos (Object Diagram):**
  + **Descrição:** Uma instância do diagrama de classes que mostra objetos específicos e suas relações em um momento particular.
  + **Elementos:** Objetos (instâncias de classes) e links (instâncias de associações).
* **Diagrama de Componentes (Component Diagram):**
  + **Descrição:** Mostra os componentes do sistema e suas interdependências.
  + **Elementos:** Componentes, interfaces, e dependências.
* **Diagrama de Pacotes (Package Diagram):**
  + **Descrição:** Organiza elementos em grupos chamados pacotes para mostrar a estrutura modular do sistema.
  + **Elementos:** Pacotes e dependências.

**Características dos Modelos Estáticos:**

* Foco na estrutura e organização do sistema.
* Ajuda a definir a arquitetura e o design detalhado do sistema.
* Fundamental para a implementação, pois detalha os componentes e suas interações.

**3. Modelos de Comportamento**

**Objetivo:** Representar o comportamento dinâmico do sistema, mostrando como o sistema responde a eventos e interage ao longo do tempo.

**Principais Diagramas:**

* **Diagrama de Sequência (Sequence Diagram):**
  + **Descrição:** Mostra a interação entre objetos no tempo, enfatizando a ordem das mensagens trocadas.
  + **Elementos:** Objetos (ou lifelines), mensagens, e ativações.
  + **Exemplo:** Um diagrama que ilustra o processo de login de um usuário, mostrando as mensagens trocadas entre o "Usuário", "Interface de Login", e "Sistema de Autenticação".
* **Diagrama de Atividades (Activity Diagram):**
  + **Descrição:** Representa o fluxo de atividades ou o workflow dentro do sistema.
  + **Elementos:** Atividades, transições, estados de decisão, e estados iniciais/finais.
  + **Exemplo:** Um diagrama que mostra o processo de aprovação de um pedido de compra, desde a submissão até a aprovação final.
* **Diagrama de Estados (State Diagram):**
  + **Descrição:** Mostra os estados pelos quais um objeto passa durante seu ciclo de vida e os eventos que causam a transição entre esses estados.
  + **Elementos:** Estados, transições, eventos, e ações.
  + **Exemplo:** Um diagrama que descreve os estados de uma "Conta Bancária" como "Aberta", "Suspensa", e "Fechada", e os eventos que causam as mudanças de estado.
* **Diagrama de Comunicação (Communication Diagram):**
  + **Descrição:** Similar ao diagrama de sequência, mas foca nas relações entre os objetos e as mensagens trocadas.
  + **Elementos:** Objetos, mensagens, e links de comunicação.

**Características dos Modelos de Comportamento:**

* Foco nas interações e no fluxo de controle dentro do sistema.
* Útil para entender a dinâmica do sistema e como os componentes colaboram.
* Importante para o design de sistemas orientados a eventos e fluxos de trabalho.

**Resumo das Diferenças**

* **Modelos Funcionais:** Enfocam "o que" o sistema deve fazer (funcionalidades e interações com usuários).
* **Modelos Estáticos:** Enfocam "como" o sistema é estruturado (componentes e suas relações).
* **Modelos de Comportamento:** Enfocam "como" o sistema se comporta ao longo do tempo (interações dinâmicas e fluxos de controle).

Uma imagem com texto, captura de ecrã, Tipo de letra, documento

Descrição gerada automaticamente

**3-**

**Abordagens Centradas em Cenários (Utilização)**

**Objetivo:** Focar na experiência dos usuários e nas interações que eles têm com o sistema para determinar os requisitos.

**Características:**

1. **Cenários e Casos de Uso:**
   * **Definição:** Um cenário é uma descrição detalhada de uma sequência de eventos que um usuário segue para alcançar um objetivo específico utilizando o sistema.
   * **Casos de Uso:** Utilizam casos de uso (use cases) para descrever interações entre atores (usuários ou outros sistemas) e o sistema, destacando as funcionalidades necessárias.
2. **Envolvimento dos Usuários:**
   * **Participação Ativa:** Os usuários e stakeholders estão fortemente envolvidos no processo de definição de requisitos, fornecendo insights sobre como eles interagem com o sistema.
   * **Workshops e Entrevistas:** São comuns para capturar as necessidades e expectativas dos usuários.
3. **Foco em Experiência do Usuário (UX):**
   * **Usabilidade e Acessibilidade:** Grande ênfase é colocada em como o sistema será usado na prática, incluindo aspectos de usabilidade e acessibilidade.
4. **Iterativo e Incremental:**
   * **Desenvolvimento Iterativo:** Permite a construção e revisão contínua dos cenários, ajustando requisitos conforme o entendimento do sistema e das necessidades dos usuários evolui.

**Benefícios:**

* **Maior Alinhamento com as Necessidades dos Usuários:** Garante que o sistema desenvolvido atende às expectativas e necessidades reais dos usuários.
* **Facilita a Validação:** Cenários claros e detalhados facilitam a validação dos requisitos com os stakeholders.
* **Melhoria na Experiência do Usuário:** Foco na interação e usabilidade melhora a satisfação do usuário final.

**Exemplos:**

* **Sistemas Interativos:** Aplicativos móveis, sistemas de comércio eletrônico, plataformas de mídia social.
* **Sistemas Orientados a Serviços:** Portais de atendimento ao cliente, sistemas de reservas online.

**Abordagens Centradas no Produto**

**Objetivo:** Focar nas características e funcionalidades do produto em si para determinar os requisitos.

**Características:**

1. **Especificações de Produto:**
   * **Definição de Funcionalidades:** Detalha as funcionalidades, características técnicas e capacidades que o produto deve ter.
   * **Requisitos Não Funcionais:** Inclui desempenho, segurança, escalabilidade, compatibilidade e outros aspectos técnicos.
2. **Visão de Produto:**
   * **Roadmap de Produto:** Define uma visão clara e um roadmap para o desenvolvimento do produto, incluindo versões futuras e melhorias planejadas.
   * **Documentação Detalhada:** Criação de documentos de especificação detalhados que guiam o desenvolvimento técnico.
3. **Foco na Arquitetura e Design:**
   * **Desenho da Arquitetura:** Grande ênfase na arquitetura do sistema e no design detalhado dos componentes.
   * **Padrões e Frameworks:** Utilização de padrões de design e frameworks para assegurar a qualidade técnica do produto.
4. **Planejamento e Gestão de Projetos:**
   * **Planejamento Rigoroso:** Foco no planejamento detalhado, cronogramas e gestão de recursos para garantir a entrega do produto conforme especificado.

**Benefícios:**

* **Clareza Técnica:** Fornece uma visão clara e detalhada das capacidades técnicas e das especificações do produto.
* **Controle de Qualidade:** Facilita o controle de qualidade e assegura que o produto atende aos requisitos técnicos e padrões estabelecidos.
* **Planejamento a Longo Prazo:** Ajuda no planejamento estratégico e na definição de versões e melhorias futuras do produto.

**Exemplos:**

* **Produtos de Software:** Sistemas operacionais, suítes de produtividade, ferramentas de desenvolvimento.
* **Hardware e Sistemas Integrados:** Dispositivos eletrônicos, sistemas embarcados, produtos de IoT (Internet das Coisas).

**Comparação e Contexto de Uso**

* **Cenários de Utilização:**
  + **Quando Usar:** Ideal para sistemas interativos, onde a experiência do usuário é crítica e as interações são complexas e variadas.
  + **Vantagens:** Alinhamento com as necessidades dos usuários, facilidade de validação, melhoria na experiência do usuário.
* **Produto:**
  + **Quando Usar:** Adequado para produtos técnicos com requisitos bem definidos, onde a arquitetura e o desempenho são críticos.
  + **Vantagens:** Clareza técnica, controle de qualidade, planejamento a longo prazo.

Em resumo, a escolha entre uma abordagem centrada em cenários e uma abordagem centrada no produto depende do tipo de sistema que está sendo desenvolvido e das prioridades do projeto. As abordagens centradas em cenários são mais focadas na experiência do usuário e nas interações práticas, enquanto as abordagens centradas no produto são mais detalhadas tecnicamente e orientadas para a visão e evolução do produto.

S. M. A. R. T. → Specific, Measurable, Attainable, Relevant and time-sensitive (Específico, Mensurável, Atingível, Relevante e Rastreável no tempo)

Uma imagem com texto, captura de ecrã, Tipo de letra, software

Descrição gerada automaticamente

Ultima pergunta: O processo de determinação de requisitos é, sem dúvida, um desafio de interação humana porque depende fortemente da comunicação eficaz, compreensão mútua, resolução de conflitos e construção de relacionamentos de confiança. As habilidades técnicas são importantes, mas sem a capacidade de interagir e compreender as pessoas envolvidas, a elicitação de requisitos pode falhar em capturar o que realmente é necessário para o sucesso do sistema.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

* Os seis princípios propostos por Ivar Jacobson para a adoção de casos de utilização no "Use Cases 2.0" visam criar um processo de desenvolvimento de sistemas mais ágil, eficiente e alinhado com os objetivos de negócio. Ao focar no valor, adotar um desenvolvimento incremental, manter modelos leves, usar casos de uso como fio condutor, revisar continuamente e promover a colaboração efetiva, as equipes de desenvolvimento podem melhorar significativamente a qualidade e a eficácia de seus projetos.
* A relação entre requisitos e casos de utilização é intrínseca e complementar. Os casos de utilização servem como um meio estruturado e narrativo para capturar, especificar, validar e comunicar requisitos funcionais. Eles fornecem uma visão clara de como o sistema deve interagir com seus usuários, garantindo que as necessidades dos stakeholders sejam atendidas de maneira eficaz e eficiente.

Uma imagem com texto, Tipo de letra, captura de ecrã, documento

Descrição gerada automaticamente

**1. Início da Arquitetura (Architecture Inception)**

Esta fase inicial é crucial para estabelecer as bases da arquitetura do sistema. As atividades típicas incluem:

* **Identificação de Requisitos de Arquitetura:** Compreensão dos requisitos não funcionais (performance, escalabilidade, segurança, etc.) e funcionais que impactam decisões arquiteturais.
* **Definição da Visão Arquitetural:** Estabelecimento de uma visão inicial da arquitetura, alinhada com os objetivos de negócio e estratégias técnicas da organização.
* **Identificação dos Stakeholders:** Identificação das partes interessadas que terão impacto ou interesse na arquitetura do sistema.
* **Seleção de Tecnologias e Plataformas:** Avaliação e seleção de tecnologias, plataformas e frameworks que serão utilizados na implementação da arquitetura.

**2. Elaboração da Arquitetura (Architecture Elaboration)**

Nesta fase, a arquitetura é elaborada com mais detalhes. As atividades principais incluem:

* **Modelagem Arquitetural:** Desenvolvimento de modelos arquiteturais, como diagramas de componentes, diagramas de pacotes, diagramas de implantação, entre outros, que representam a estrutura e o comportamento do sistema.
* **Prototipação e Experimentação:** Construção de protótipos para validar decisões arquiteturais e explorar soluções técnicas para desafios críticos.
* **Gestão de Riscos Arquiteturais:** Identificação e mitigação de riscos relacionados à arquitetura, garantindo que as decisões arquiteturais sejam robustas e sustentáveis.
* **Definição de Interfaces e Contratos:** Especificação de interfaces entre componentes e módulos, definindo contratos claros para a comunicação e integração entre partes do sistema.

**3. Construção da Arquitetura (Architecture Construction)**

Durante esta fase, a arquitetura é implementada e integrada ao restante do sistema. As atividades típicas incluem:

* **Implementação de Componentes Arquiteturais:** Desenvolvimento dos componentes principais da arquitetura, garantindo que eles sigam os padrões e diretrizes estabelecidos.
* **Integração e Testes Arquiteturais:** Integração dos componentes arquiteturais e realização de testes para verificar a conformidade com a arquitetura definida e garantir a interoperabilidade adequada entre os módulos.
* **Validação da Arquitetura:** Verificação de que a arquitetura implementada atende aos requisitos funcionais e não funcionais definidos anteriormente, através de revisões e validações com os stakeholders.

**4. Transição da Arquitetura (Architecture Transition)**

Nesta fase final, a arquitetura é entregue e preparada para o uso operacional. As atividades principais incluem:

* **Documentação da Arquitetura:** Elaboração de documentação detalhada da arquitetura, incluindo manuais, guias de design e outras informações necessárias para suportar a operação e manutenção do sistema.
* **Treinamento e Suporte:** Preparação e fornecimento de treinamento para equipes que utilizarão e manterão a arquitetura do sistema.
* **Avaliação Pós-Implantação:** Monitoramento inicial da performance e eficácia da arquitetura em produção, identificando áreas de melhoria contínua e refinamento.

**Considerações Finais**

No OpenUP, o desenvolvimento da arquitetura de software é iterativo e evolutivo, integrando-se de forma contínua com outras disciplinas de desenvolvimento de software, como análise, design, implementação e teste. As atividades descritas acima são adaptadas e ajustadas conforme necessário para atender às necessidades específicas do projeto, mantendo um foco constante na entrega de valor e na qualidade do sistema desenvolvido.

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

**Distinção Resumida:**

* **CI:** Garante que todas as alterações de código sejam integradas continuamente e testadas para detectar problemas de integração o mais cedo possível.
* **CD (Entrega Contínua):** Mantém o software sempre em um estado de pronto para ser entregue ao cliente, automatizando a criação de artefatos de software e garantindo que qualquer versão do software possa ser implantada manualmente.
* **CD (Implantação Contínua):** Automatiza o processo de implantação para que cada alteração de código bem-sucedida seja automaticamente implantada no ambiente de produção.

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Histórias de usuário e casos de utilização são técnicas utilizadas para capturar requisitos funcionais em projetos de software, mas possuem diferentes enfoques e níveis de detalhe. Vamos comparar esses dois conceitos em relação aos pontos comuns e diferenças e discutir como podem ser usados de forma complementar.

**Pontos Comuns:**

1. **Captura de Requisitos:**
   * Ambos são utilizados para capturar o que o sistema deve fazer do ponto de vista do usuário.
   * Servem como base para a comunicação entre as partes interessadas (stakeholders) e a equipe de desenvolvimento.
2. **Foco no Usuário:**
   * Ambos se concentram nas necessidades e objetivos dos usuários finais.
   * Ajudam a garantir que o sistema atenda aos requisitos reais dos usuários.
3. **Base para Testes:**
   * Podem ser utilizados para definir critérios de aceitação e desenvolver casos de teste.
   * Ajudam a validar se o sistema implementado cumpre os requisitos especificados.

**Diferenças:**

1. **Nível de Detalhe:**
   * **Histórias de Usuário:** São descrições breves e simples das funcionalidades desejadas, geralmente com o formato "Como [tipo de usuário], eu quero [ação] para [benefício]". São menos detalhadas e mais focadas em captar rapidamente a intenção do usuário.
   * **Casos de Utilização:** São descrições detalhadas das interações entre o usuário e o sistema. Incluem o fluxo principal, fluxos alternativos, pré-condições, pós-condições e exceções.
2. **Formato e Estrutura:**
   * **Histórias de Usuário:** Geralmente consistem em uma ou duas frases que descrevem a necessidade do usuário e podem ser acompanhadas de critérios de aceitação.
   * **Casos de Utilização:** Incluem um diagrama de casos de uso (em UML) e uma descrição narrativa detalhada de cada interação.
3. **Enfoque:**
   * **Histórias de Usuário:** Focam na entrega de valor de negócio de maneira incremental e são utilizadas frequentemente em metodologias ágeis, como Scrum.
   * **Casos de Utilização:** Focam na compreensão completa dos requisitos do sistema e nas interações detalhadas entre os atores e o sistema, sendo mais comuns em metodologias tradicionais ou híbridas.
4. **Utilização no Ciclo de Vida do Projeto:**
   * **Histórias de Usuário:** São utilizadas para o planejamento de iterações/sprints e para gerenciar o backlog do produto em metodologias ágeis.
   * **Casos de Utilização:** São utilizados durante a fase de análise de requisitos e design detalhado, proporcionando uma visão abrangente das funcionalidades do sistema.

**Uso Complementar:**

Histórias de usuário e casos de utilização podem ser usados de forma complementar para maximizar os benefícios de ambas as abordagens:

1. **Iniciação com Histórias de Usuário:**
   * Comece com histórias de usuário para capturar rapidamente as necessidades e objetivos do usuário em um formato simples e ágil.
   * Use as histórias de usuário para o planejamento de iterações/sprints e priorização de funcionalidades.
2. **Detalhamento com Casos de Utilização:**
   * Para funcionalidades complexas ou críticas, elabore casos de utilização a partir das histórias de usuário para obter um entendimento mais detalhado e abrangente das interações do sistema.
   * Use casos de utilização para documentar fluxos alternativos e exceções que podem não ser capturados nas histórias de usuário.
3. **Critérios de Aceitação e Testes:**
   * Utilize os critérios de aceitação das histórias de usuário como base para a criação de casos de teste iniciais.
   * Complemente com casos de teste derivados dos casos de utilização para garantir que todas as interações, inclusive as alternativas e exceções, sejam validadas.

**Conclusão:**

Histórias de usuário e casos de utilização são ferramentas valiosas com seus próprios pontos fortes. As histórias de usuário são ótimas para capturar rapidamente as necessidades do usuário e priorizar funcionalidades em um ambiente ágil. Os casos de utilização fornecem uma visão detalhada das interações do sistema, útil para análise e design aprofundados. Usados de forma complementar, eles garantem que tanto os objetivos de negócio quanto os detalhes técnicos das interações do sistema sejam adequadamente capturados e atendidos.

Uma imagem com texto, Tipo de letra, captura de ecrã, documento

Descrição gerada automaticamente

**Explique o objetivo da “Daily Scrum meeting”**

A **Daily Scrum meeting**, também conhecida como "stand-up meeting", é uma reunião diária de 15 minutos onde a equipe Scrum sincroniza suas atividades e cria um plano para as próximas 24 horas. O objetivo principal é:

1. **Sincronização:** Garantir que todos os membros da equipe estejam cientes do progresso e dos obstáculos.
2. **Planejamento:** Ajustar o trabalho do dia para otimizar a colaboração e eficiência.
3. **Identificação de Bloqueios:** Detectar e comunicar impedimentos que possam atrasar o progresso.

**Relacione os conceitos de sprint e iteração e discuta a sua duração esperada**

* **Sprint:** No Scrum, um sprint é um ciclo de desenvolvimento com duração fixa, geralmente entre 1 a 4 semanas, durante o qual uma quantidade específica de trabalho deve ser completada e estar potencialmente pronta para entrega.
* **Iteração:** É um termo mais geral usado em metodologias ágeis que representa um ciclo de desenvolvimento repetitivo e incremental. No contexto do Scrum, cada sprint é uma iteração.
* **Duração:** A duração típica de um sprint é de 2 semanas, embora possa variar conforme a complexidade do projeto e as necessidades da equipe.

**Explique o método de pontuação das histórias (e critérios aplicados)**

* **Método de Pontuação:** A pontuação das histórias geralmente é feita usando "story points", que são uma medida abstrata de esforço necessário para implementar uma história de usuário.
* **Critérios Aplicados:**
  + **Complexidade:** O grau de dificuldade técnica.
  + **Volume de Trabalho:** A quantidade de tarefas envolvidas.
  + **Incertezas:** Grau de desconhecimento ou variabilidade nos requisitos.
  + **Riscos:** Potenciais problemas que podem surgir.

**Identificar os papéis numa equipe de Scrum e as principais "cerimônias"**

* **Papéis:**
  + **Product Owner:** Responsável por maximizar o valor do produto e gerenciar o backlog do produto.
  + **Scrum Master:** Facilita o processo Scrum, ajuda a remover impedimentos e assegura que a equipe segue as práticas Scrum.
  + **Development Team:** Conjunto de profissionais que trabalham na construção do incremento do produto.
* **Cerimônias:**
  + **Sprint Planning:** Planejamento do trabalho a ser feito no sprint.
  + **Daily Scrum:** Reunião diária de 15 minutos para sincronização.
  + **Sprint Review:** Revisão do trabalho completado e demonstração do incremento ao Product Owner e outras partes interessadas.
  + **Sprint Retrospective:** Reflexão sobre o sprint passado para identificar melhorias no processo.

**Relacione as práticas previstas no SCRUM e os princípios do "Agile Manifest": em que medida estão alinhados?**

* **Colaboração:** Scrum promove reuniões diárias e comunicação constante, alinhado ao princípio ágil de colaboração entre as partes interessadas.
* **Entrega Contínua:** A prática de sprints curtos e revisões regulares está alinhada ao princípio ágil de entrega contínua de software funcional.
* **Adaptação:** O Scrum incorpora a adaptação constante através de reuniões de retrospectiva e planejamento, alinhado ao princípio ágil de responder a mudanças.

**Pode-se considerar o SCRUM como a "silver bullet" (solução universal) para o sucesso dos projetos de desenvolvimento?**

* **Não, o Scrum não é uma "silver bullet".** Embora seja uma metodologia eficaz para muitos projetos, não é adequada para todos os contextos. Sua eficácia depende de fatores como a maturidade da equipe, a natureza do projeto e a cultura organizacional. Cada projeto pode exigir adaptações ou até a combinação de diferentes metodologias.

**Identifique alguns desafios/bloqueios à implementação eficaz do SCRUM.**

* **Resistência à Mudança:** Membros da equipe ou stakeholders podem resistir à adoção de novas práticas e processos.
* **Falta de Compromisso:** A falta de comprometimento dos membros da equipe ou do Product Owner pode comprometer a eficácia do Scrum.
* **Impedimentos Não Resolvidos:** Bloqueios identificados durante as Daily Scrums que não são resolvidos rapidamente podem atrasar o progresso.
* **Entendimento Inadequado do Scrum:** A implementação incorreta devido a um entendimento superficial ou incorreto das práticas e princípios Scrum.

Essas respostas resumem as principais questões sobre o framework Scrum de forma concisa.