### 1.2 Ciclo de Vida de Desenvolvimento de Software

O Ciclo de Vida do Desenvolvimento de Software descreve todo o percurso, desde a concepção inicial até a disponibilização final ao usuário. Para exemplificar esse conceito para aqueles que iniciam como estagiários na área de sistemas, considere o desenvolvimento de um aplicativo para controle de tarefas. Este exemplo será abordado sob duas perspectivas metodológicas distintas: o modelo cascata e o Scrum.

Exemplo Aplicado: Sistema de Gestão de Reservas em Biblioteca Acadêmica com o Processo Unificado (RUP)

O Processo Unificado da Rational (RUP - Rational Unified Process), uma abordagem iterativa destacada, estrutura o desenvolvimento em quatro fases principais: Concepção, Elaboração, Construção e Transição. Cada fase é associada a disciplinas específicas, trabalhadas com intensidade variável, proporcionando equilíbrio entre rigor técnico e flexibilidade adaptativa.

Fases do RUP Aplicadas ao Projeto

* Concepção: Nesta fase inicial, define-se o escopo do projeto e valida-se o caso de negócio por meio de entrevistas e análises detalhadas dos processos atuais, identificando falhas no controle manual das reservas e definindo casos de uso iniciais, como "Realizar Reserva".
* Elaboração: Aqui, requisitos são refinados e a arquitetura do sistema é estabelecida. Diagramas UML são utilizados para formalizar o entendimento das funcionalidades e a arquitetura MVC (Model-View-Controller) é selecionada para organizar a lógica, dados e interface.
* Construção: Durante esta fase ocorre a implementação completa das funcionalidades planejadas. Módulos específicos são desenvolvidos e submetidos a rigorosos testes unitários e de integração, assegurando qualidade e robustez.
* Transição: Nesta etapa, o sistema é implantado em ambiente real, incluindo atividades de instalação, treinamento dos usuários e elaboração de documentação detalhada. Ajustes finos são realizados com base no feedback dos usuários durante a fase inicial de monitoramento.

Disciplinas Complementares no RUP

O RUP integra também disciplinas complementares essenciais:

* Gestão de Configuração e Mudança: controla versões e alterações para garantir rastreabilidade.
* Gestão de Projetos: assegura planejamento, acompanhamento das atividades e monitoramento contínuo dos resultados.
* Ambiente: fornece infraestrutura necessária com ferramentas padronizadas e recursos técnicos.

A metodologia RUP demonstra grande eficácia ao proporcionar clareza organizacional e divisão coerente entre fases e disciplinas. A aplicação prática no sistema de gestão de reservas para a biblioteca acadêmica confirma sua capacidade em entregar soluções robustas, confiáveis e aderentes às reais necessidades dos usuários, contribuindo para um aprendizado significativo especialmente para profissionais em formação.

1.2.1 Implantação do Sistema de Gestão de Reservas Utilizando o Modelo Cascata

A implantação do sistema de gestão de reservas de livros em biblioteca acadêmica utilizando o modelo cascata segue um método rigorosamente sequencial e estruturado. Primeiramente, na fase de documentação dos requisitos, a equipe realiza entrevistas detalhadas com usuários para identificar todas as funcionalidades necessárias, tais como "realizar reserva", "consultar disponibilidade" e "cancelar reserva". Nesta etapa inicial, define-se claramente cada requisito, assegurando que não ocorram desvios durante as fases posteriores do desenvolvimento.

Em seguida, ocorre a fase de design, onde a equipe técnica projeta a arquitetura geral do sistema, incluindo interfaces de usuário, telas específicas e fluxos de interação detalhados. Os desenvolvedores criam diagramas representativos da interface gráfica e do banco de dados para garantir a clareza visual e funcional do sistema. Ao final desta etapa, todas as decisões relacionadas ao layout e funcionalidades já devem estar precisamente estabelecidas e documentadas.

A terceira fase consiste na implementação, na qual a equipe de desenvolvimento codifica o sistema estritamente conforme o design previamente aprovado. Durante esse período, os desenvolvedores constroem cada módulo e integram as funcionalidades conforme planejado. Esta etapa requer estrita aderência às especificações anteriores, com revisões periódicas para assegurar a conformidade com os requisitos iniciais.

Na fase seguinte, realiza-se a etapa de testes, onde o sistema completo passa por rigorosos procedimentos de validação para identificar e corrigir quaisquer erros ou falhas operacionais. Equipes especializadas realizam testes unitários, testes integrados e testes sistêmicos para garantir o perfeito funcionamento do software. Após aprovação dos testes, o sistema torna-se apto para ser entregue aos usuários finais.

Finalmente, na fase de entrega, o sistema completo é instalado e implantado no ambiente real da biblioteca acadêmica. Nesta etapa conclusiva, são realizados treinamentos específicos com usuários finais para assegurar que estes dominem completamente o funcionamento do novo sistema. Além disso, é fornecida documentação detalhada e procedimentos claros para uso e manutenção, garantindo que o sistema possa ser operado e sustentado eficazmente a longo prazo.

1.2.2 Implantação do Sistema de Gestão de Reservas Utilizando Scrum e XP

A implantação do sistema de gestão de reservas para uma biblioteca acadêmica utilizando Scrum combinado com Extreme Programming (XP) adota uma abordagem iterativa e incremental. Primeiramente, a equipe desenvolve o backlog do produto, listando todas as funcionalidades requeridas pelos usuários, como "realizar reserva", "renovar empréstimo" e "verificar histórico". Essa lista é continuamente revisada e priorizada pelo Product Owner para assegurar valor constante ao usuário final.

Em seguida, no planejamento dos sprints, o projeto é dividido em ciclos curtos, geralmente com duração entre uma a quatro semanas. A equipe determina claramente quais funcionalidades serão entregues em cada sprint, promovendo transparência e alinhamento contínuo com as expectativas dos usuários. O objetivo desta fase é garantir entregas incrementais e rápidas, permitindo validações frequentes do progresso do sistema.

Durante o desenvolvimento incremental, os membros da equipe adotam práticas do XP como programação em pares, integração contínua e testes automatizados. Cada funcionalidade é desenvolvida, testada e integrada ao sistema, assegurando robustez e qualidade contínua do software. Esta metodologia permite detectar problemas e realizar correções de forma ágil, minimizando retrabalho e aumentando a confiabilidade do sistema.

Na fase de revisão e feedback, ao final de cada sprint, as funcionalidades são apresentadas aos usuários finais, que fornecem avaliações imediatas e sugestões para aprimoramentos futuros. Este momento é fundamental para garantir que o sistema esteja continuamente alinhado com as expectativas dos usuários e necessidades da biblioteca. O feedback recebido é essencial para guiar as decisões futuras e melhorar constantemente o produto entregue.

Finalmente, com base nos retornos dos usuários, a equipe realiza ajustes contínuos no sistema, adaptando rapidamente funcionalidades existentes ou revisando as próximas funcionalidades a serem desenvolvidas. Essa abordagem iterativa promove alta adaptabilidade às mudanças, permitindo entregas constantes e alinhamento dinâmico com as necessidades reais da biblioteca. O uso combinado de Scrum e XP favorece uma implantação ágil e eficiente, mantendo sempre a qualidade e a satisfação dos usuários como prioridades centrais.

**1.3 Marcos Históricos das Metodologias de Desenvolvimento**

A evolução das metodologias de desenvolvimento de software ocorreu em resposta aos desafios enfrentados pelas organizações ao executar projetos complexos. Na década de 1960, a crise do software impulsionou o surgimento da engenharia de software, resultado da alta frequência de falhas em projetos que extrapolavam prazos e orçamentos previstos (Dennis, Wixom & Tegarden, 2021). Esse cenário estimulou o desenvolvimento de métodos mais rigorosos e estruturados para lidar com a complexidade crescente dos sistemas.

Nos anos 1970, houve a consolidação do modelo cascata, oferecendo uma abordagem estruturada e sequencial para o desenvolvimento de software. Esse modelo claramente definiu etapas sequenciais obrigatórias como requisitos, projeto, implementação, testes e manutenção. Contudo, apesar de sua clareza estrutural, o modelo cascata mostrou limitações na flexibilidade para mudanças durante a execução dos projetos.

Na década de 1980, o desenvolvimento estruturado emergiu com maior ênfase em técnicas rigorosas de análise e projeto. Essas técnicas proporcionaram melhorias significativas na modularidade, documentação e manutenção dos softwares desenvolvidos. Este período marcou a importância do planejamento detalhado e a divisão clara de responsabilidades dentro das equipes.

Nos anos 1990, metodologias ágeis começaram a surgir como resposta à necessidade crescente por flexibilidade, comunicação constante e capacidade de adaptação às mudanças frequentes. Metodologias como Scrum e XP ganharam destaque ao priorizar entregas incrementais e rápidas, fortalecendo a interação contínua com os usuários finais. Essas abordagens tornaram-se especialmente relevantes para ambientes dinâmicos, com demandas variáveis e urgentes.

A partir dos anos 2000, houve a consolidação das práticas de integração contínua e DevOps. Essas metodologias promovem entregas rápidas, incrementais e de alta qualidade, através da automação extensiva de testes e integração frequente entre desenvolvimento e operações (Rocha, 2024). Atualmente, essas práticas são amplamente adotadas para garantir eficiência operacional e excelência contínua na entrega de valor aos usuários.

**1.4 Função da Análise e do Projeto no Contexto do Desenvolvimento entre abordagens ágil e tradicional**

Existem duas abordagens principais no desenvolvimento de sistemas: a tradicional, representada pelo modelo cascata, e a ágil, exemplificada pelo Scrum e XP. Cada abordagem oferece metodologias específicas e diferentes estratégias na gestão do projeto de software. A escolha entre elas depende das características do projeto, tais como complexidade, estabilidade dos requisitos e necessidade de flexibilidade.

1.4.1 Metodologias Tradicionais (Cascata)

As metodologias tradicionais, como o modelo cascata, propõem uma abordagem sequencial e estruturada. Nesse modelo, cada fase do desenvolvimento deve ser totalmente concluída antes de iniciar a seguinte, garantindo uma documentação detalhada e rigorosa. Embora promova clareza e controle, essa abordagem limita significativamente a capacidade de adaptação às mudanças durante o projeto.

Uma característica importante do modelo cascata é sua baixa flexibilidade em alterações de requisitos. Mudanças realizadas após o início das etapas podem causar grandes retrabalhos e custos adicionais. Por exemplo, um sistema de gestão de biblioteca que precise incorporar novas funcionalidades após a fase de implementação pode enfrentar dificuldades técnicas e aumento expressivo de custos.

Outra desvantagem dessa metodologia é o tempo prolongado até a entrega final do produto. Como todas as etapas devem ser completas sequencialmente, os usuários só podem acessar o sistema após a conclusão de todo o processo. Um sistema complexo, como um ERP, pode levar meses ou anos até ser completamente implantado e operacionalizado, dificultando respostas rápidas a necessidades emergentes.

1.4.2 Metodologias Ágeis (Scrum e XP)

As metodologias ágeis, como Scrum e XP, seguem uma abordagem iterativa e incremental, permitindo entregas rápidas e constantes. O desenvolvimento ocorre em ciclos curtos chamados sprints, proporcionando entregas frequentes de funcionalidades incrementais. Isso permite maior interação com os usuários e ajustes contínuos conforme o feedback recebido.

Uma vantagem significativa dos métodos ágeis é o engajamento constante dos usuários e clientes. A comunicação frequente permite adaptações rápidas às necessidades identificadas pelos usuários durante o desenvolvimento. Por exemplo, em um sistema de gestão de biblioteca, módulos como empréstimo e devolução podem ser aprimorados continuamente conforme a necessidade identificada pela equipe e pelos usuários.

Outra vantagem fundamental das metodologias ágeis é a flexibilidade para alterações frequentes. A possibilidade de ajustar ou acrescentar novos requisitos ao longo do projeto minimiza retrabalhos extensivos. Assim, um sistema como o de gestão de biblioteca pode rapidamente incorporar novas funcionalidades, como reservas online ou notificações automáticas, adaptando-se às demandas dos usuários.

**1.5 Comparação detalhada entre abordagens Ágil e Tradicional**

Ambas as metodologias apresentam vantagens e limitações específicas, devendo a escolha considerar o contexto do projeto. As metodologias tradicionais oferecem documentação extensa e previsibilidade, adequadas para projetos com requisitos estáveis. Já as metodologias ágeis são apropriadas para projetos que demandam adaptação constante e entregas rápidas.

1.5.1 Metodologias Tradicionais

As metodologias tradicionais oferecem fases claramente definidas, facilitando o planejamento detalhado e controle rigoroso do projeto. Essa abordagem gera documentação extensa e detalhada, útil para manutenção futura e treinamento de equipes. Contudo, sua rigidez gera desafios significativos para incorporar mudanças, resultando frequentemente em retrabalhos dispendiosos.

1.5.2 Metodologias Ágeis

As metodologias ágeis oferecem flexibilidade e entregas frequentes, permitindo rápida adaptação às mudanças. Os projetos ágeis têm ciclos curtos e comunicação constante, garantindo maior alinhamento às expectativas dos usuários. Entretanto, exigem engajamento contínuo dos clientes e apresentam menor previsibilidade a longo prazo, o que pode dificultar planejamentos detalhados.

Dessa forma, ambas as abordagens têm aplicação específica, e sua seleção deve basear-se em uma análise cuidadosa das características do projeto, da maturidade da equipe e das expectativas dos stakeholders. Assim, é crucial que gestores e analistas considerem cuidadosamente esses fatores ao decidir sobre a metodologia ideal. Com esse entendimento, os projetos terão maior probabilidade de sucesso ao atender efetivamente às necessidades dos usuários e alcançar os objetivos organizacionais.

1.5.3 Metodologias Ágeis (Scrum e XP)

As metodologias ágeis, como Scrum e XP, oferecem um desenvolvimento flexível, iterativo e incremental, adaptando-se rapidamente às mudanças. Isso garante ciclos curtos de entrega (sprints), permitindo entregas frequentes e gerando valor imediato aos usuários. Por exemplo, startups conseguem rapidamente lançar e ajustar funcionalidades de aplicativos conforme o feedback dos usuários.

Uma vantagem adicional das metodologias ágeis é sua alta flexibilidade e capacidade de adaptação contínua aos requisitos emergentes. Mudanças frequentes são facilmente incorporadas ao longo do desenvolvimento sem impacto negativo significativo. Equipes desenvolvendo aplicativos podem rapidamente implementar novas funcionalidades solicitadas pelos usuários, mantendo a relevância e satisfação contínua.

Contudo, metodologias ágeis também têm desafios, como menor previsibilidade no planejamento de longo prazo. A flexibilidade pode dificultar estimativas precisas de prazos e custos, especialmente com frequentes alterações de requisitos. Além disso, exigem alto envolvimento contínuo do cliente, o que pode ser um desafio se o cliente não estiver regularmente disponível para fornecer feedback necessário.