# Metodologias

## Metodologias Tradicionais

### Em cascata com retorno:

* **Descrição:** Estrutura de desenvolvimento linear onde o projeto é dividido em fases sequenciais (requisitos, design, implementação, testes, manutenção), com a possibilidade de voltar a fases anteriores para correções.
* **Objetivos:** Completar cada fase do desenvovimento antes de passar à próxima.
* **Aplicabilidade:** Adequada para projetos com requisitos bem definidos e estáveis, como software de automação industrial, onde as mudanças são mínimas ou controladas. Pode ser ideal para projetos onde o cliente sabe exatamente o que quer desde o ínicio.

### Modelo em V

* **Descrição:** Este modelo é uma variante da cascata, onde cada fase de desenvolvimento tem uma fase de teste associado (exemplo: design – teste de verificação).
* **Objetivos:** Validar o projeto em cada etapa, garantindo que o produto final esteja em conformidade com os requisitos desde o início.
* **Aplicabilidade:** Útil para projetos críticos onde a qualidade é essencial, como por exemplo sistemas de segurança e automação que exigem verificações rigorosas. Este modelo garante um ciclo de feedback em cada etapa de teste mas permanece rígido em relação ás mudanças em relação ás mudanças de requisitos.

### Modelo Incremental

* **Descrição:** O projeto é desenvolvido e entregue em pequenos incrementos, cada um com um conjunto de funcionalidades novas.
* **Objetivos:** Entregar partes do sistema rapidamentepara testar a viabilidade e realizar ajustes ao longo do tempo.
* **Aplicabilidade:** Adequado para projetos com tempo de entrega apertado ou com funcionalidades que podem ser melhoradas iterativamente, como websites ou outro tipo de aplicações para a Internet. Este modelo permite ver resultados cedo, mas o design inicial precisa de ser suficientemente sólido para suportar incremento

### Modelo Espiral

* **Descrição:** Combina a abordagem incremental com a análise de risco em cada iteração. Este modelo inclui planeamento, análise de riscos, desenvolvimento e avaliação.
* **Objetivos:** Reduzir o risco e o custo, permitindo a revisão e o ajuste em cada ciclo.
* **Aplicabilidade:** Este modelo é usado para projetos com alguma dimensão e complexos, onde os riscos são elevados e há potencial para mudanças. Neste modelo é permitido avaliar constantemente os riscos, mas é dispendioso em recursos e tempo.

## Metodologias modernas (ágeis)

### Scrum

* **Descrição:** Estrutura ágil baseada em ciclos curtos de trabalho chamados *Sprints*, com duração fixa e reuniões diárias.
* **Objetivos:** Entregar um produto funcional em cada sprint promovendo adaptações rápidas e feedback contínuo.
* **Aplicabilidade:** Ideal para projetos em que os requisitos podem mudar como o desenvolvimento de aplicações móveis e jogos. Esta metodologia exige uma equipa organizada e disciplinada para garantir que todos os elementos estejam em sincronia e que o progresso seja visível e ajustável

### Extreme Programming (XP)

* **Descrição:** Focada em ciclos de desenvolvimento curtos e feedback rápido, com práticas como desenvolvimento em par e testes contínuos.
* **Objetivos:** Garantir a qualidade do código e entregar funcionalidades úteis a cada iteração.
* **Aplicabilidade:** Metodologia indicada para projetos que exigem alta qualidade técnica como software de sistemas, jogos, aplicações na cloud. O XP é de elevada ênfase em boas práticas de boas práticas de codificação, tornando o código mais robusto e fácil de manter.

### Lean Development

* **Descrição:** Metodologia que visa reduzir desperdícios e maximizar a eficiência desenvolvendo apenas o necessário e focando-se nas funcionalidades mais valiosas.
* **Objetivos:** Minimizar o esforço e tempo desperdiçado em funcionalidades ou processos desnecessários.
* **Aplicabilidade:** Adequado para start-up’s e projetos onde o tempo e orçamento são limitados. Esta metodologia foca-se na entrega de valor ao cliente de forma rápida e eficiente, mas exige que as equipas saibam bem o que não desenvolver.

### Kanban

* **Descrição:** Metodologia visual que utiliza um quadro de tarefas para monitorizar o fluxo de trabalho. As tarefas são categorizadas em “Por fazer”, “Em desenvolvimento”, “Concluido”.
* **Objetivos:** Melhorar a gestão de tarefas e garantir um fluxo contínuo de desenvolvimento.
* **Aplicabilidade:** Metodologia perfeita para equipas que precisam de flexibilidade e não querem impor ciclos fixos de trabalho. Esta metodologia é muito flexível e adaptável, mas pode ser difícil para equipas menos organizadas.

## Metodologias tradicionais VS Metodologias Modernas

|  |  |  |
| --- | --- | --- |
| **Critério** | **Metodologias Tradicionais** | **Metodologias Modernas** |
| **Flexibilidade** | Rígidas, com dificuldades em adaptar-se a mudanças. | Altamente flexíveis e adaptáveis a alterações frequentes. |
| **Foco na documentação** | Documentação detalhada em cada fase. | Documentação mínima; Foco principal no código funcional. |
| **Tempo de Entrega** | Longo prazo, com entraga única no final. | Entregas rápidas e contínuas, com iterações frequentes. |
| **Adequação ao Projeto** | Projetos com requisitos claros e pouca necessidade de mudança. | Projetos onde os requisitos podem evoluir ao longo do tempo. |
| **Exemplo de Projetos** | Automação industrial, segurança, sistemas de gestão e software de sitema. | Aplicações web, móveis, jogos, software interativo e software de sistemas. |
| **Riscos** | Elevado risco de problemas só serem detetados tardiamente. | Riscos são detetados e corrigidos ao longo do desenvolvimento. |

## Fases de desenvolvimento de um projeto de software

### Fase de definição

**Tarefas Principais:**

* Pesquisa,
* Levantamento de requisitos,
* Análise de necessidades,
* Elaboração de especificações.

### Fase de Desenvolvimento

* Design de arquitetura,
* Programação/Codificação,
* Integração de módulos,
* Testes iniciais.

### Fase de Definição

**Tarefas Principais:**

* Correções,
* Ajustes de funcionalidades,
* Melhorias de funcionalidades,
* Atualizações de segurança,
* Suporte a curto e longo prazo.

***Nota*:** A manutenção é muitas vezes ignorada mas o sucesso de um projeto depende da facilidade com que o software pode ser atualizado e corrigido. Normalmente os projetos de software encontram-se acima dos 60% em fase de manutenção

### Modelo

* **Descrição:**
* **Objetivos:**
* **Aplicabilidade:**