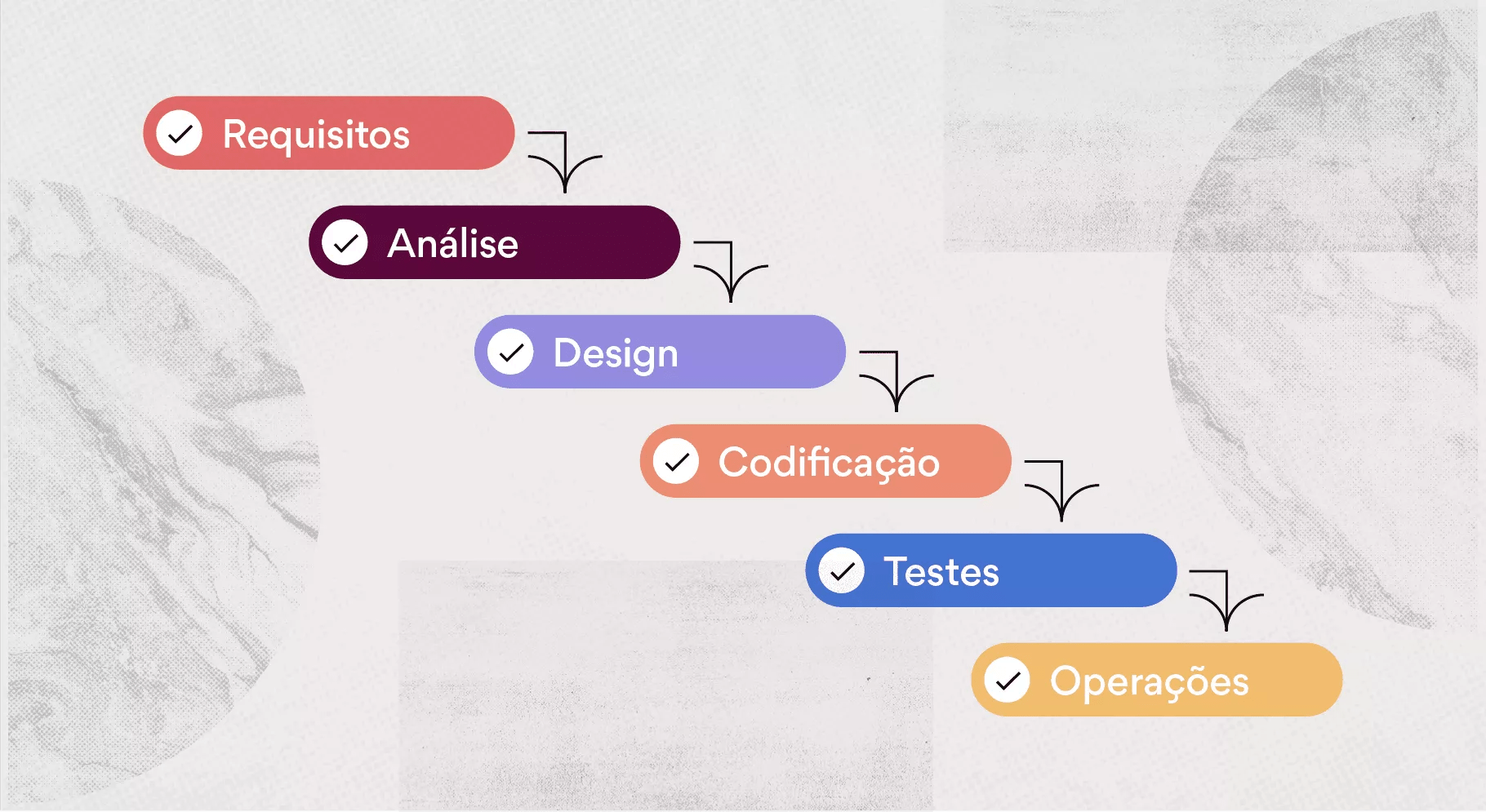
# **Metodologias clássicas de desenvolvimento de software:**

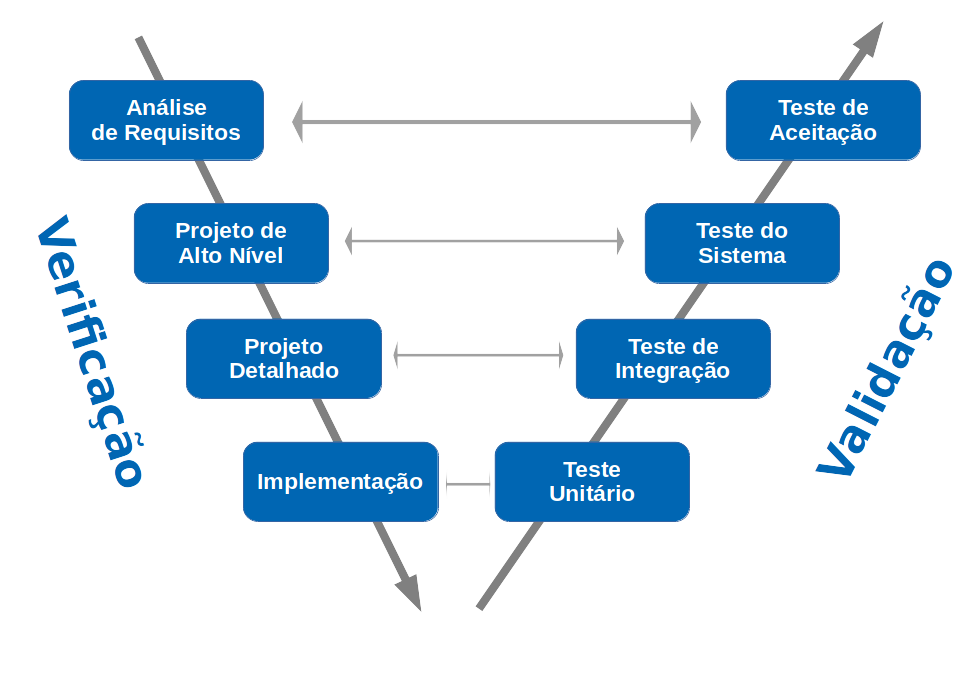
**Cascata**

* **Características**: Linear e sequencial, dividida em fases bem definidas (requisitos, design, implementação, teste, manutenção).
* **Vantagens**: Simples de entender e gerenciar, ideal para projetos com requisitos claros e estáveis.
* **Desvantagens**: Pouca flexibilidade para mudanças, problemas só são detectados tarde no processo.
* **Uso comum**: Projetos pequenos ou com requisitos bem definidos e pouco propensos a mudanças.



### ****V-Model****

* **Características**: Extensão do modelo Cascata, com fases de teste planejadas paralelamente às fases de desenvolvimento.
* **Vantagens**: Ênfase na qualidade e testes desde o início, fácil de acompanhar e gerenciar.
* **Desvantagens**: Assim como o Cascata, não lida bem com mudanças nos requisitos.
* **Uso comum**: Projetos onde a qualidade e a validação são críticas, como em sistemas de segurança ou médicos.



### ****Espiral****

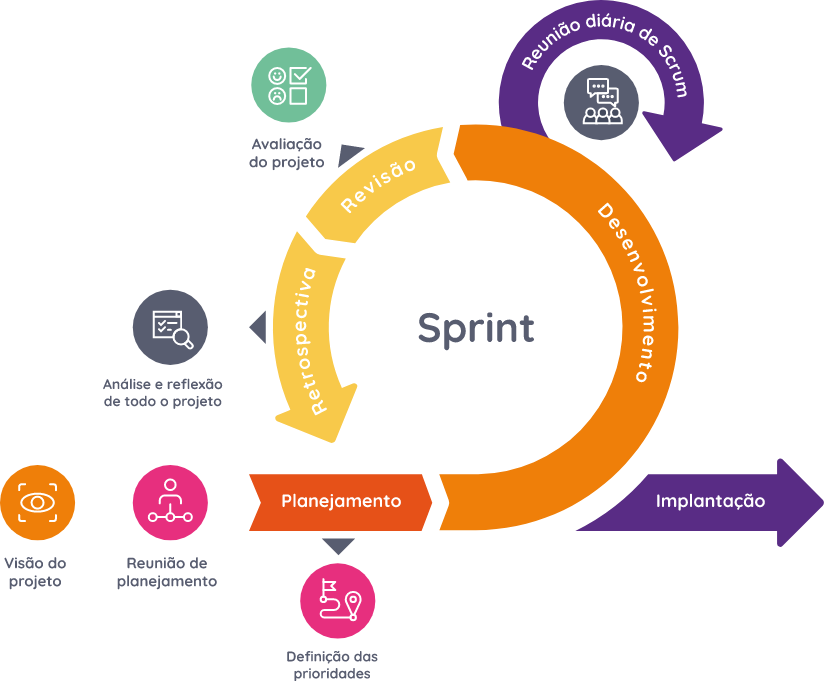
* **Características**: Iterativo e incremental, combina elementos do Cascata com prototipagem. Focado em gerenciamento de riscos.
* **Vantagens**: Flexível, permite mudanças e incorpora feedback contínuo. Ideal para projetos complexos e de alto risco.
* **Desvantagens**: Complexo de gerenciar e pode ser caro devido às iterações contínuas.
* **Uso comum**: Projetos grandes, complexos e com requisitos incertos ou em evolução.



# **Metodologias Ágeis de desenvolvimento de software:**

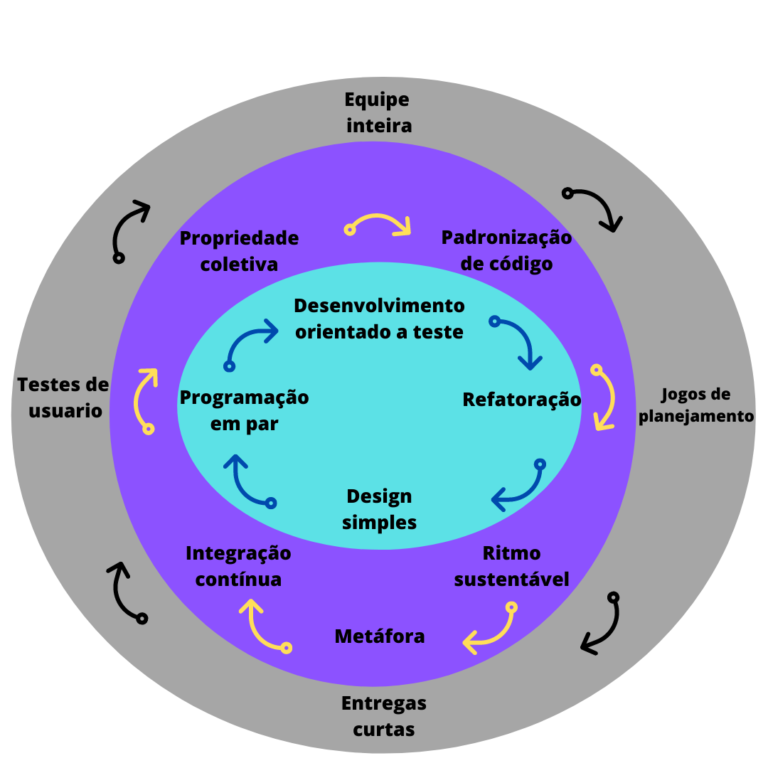
### ****Scrum****

* **Características**: Baseado em iterações curtas chamadas Sprints (2-4 semanas), com papéis definidos (Scrum Master, Product Owner, Time de Desenvolvimento).
* **Práticas**: Reuniões diárias (Daily Scrum), planejamento de Sprint, revisão e retrospectiva.
* **Vantagens**: Flexível, promove colaboração e entrega contínua de valor.
* **Uso comum**: Projetos com requisitos em evolução e equipes multidisciplinares.



### ****XP (Extreme Programming)****

* **Características**: Foco em qualidade de código e entrega rápida, com práticas como programação em pares, integração contínua e testes automatizados.
* **Práticas**: Desenvolvimento orientado a testes (TDD), releases frequentes, simplicidade no design.
* **Vantagens**: Melhoria contínua do código, adaptável a mudanças.
* **Uso comum**: Projetos de software com requisitos voláteis e necessidade de alta qualidade.



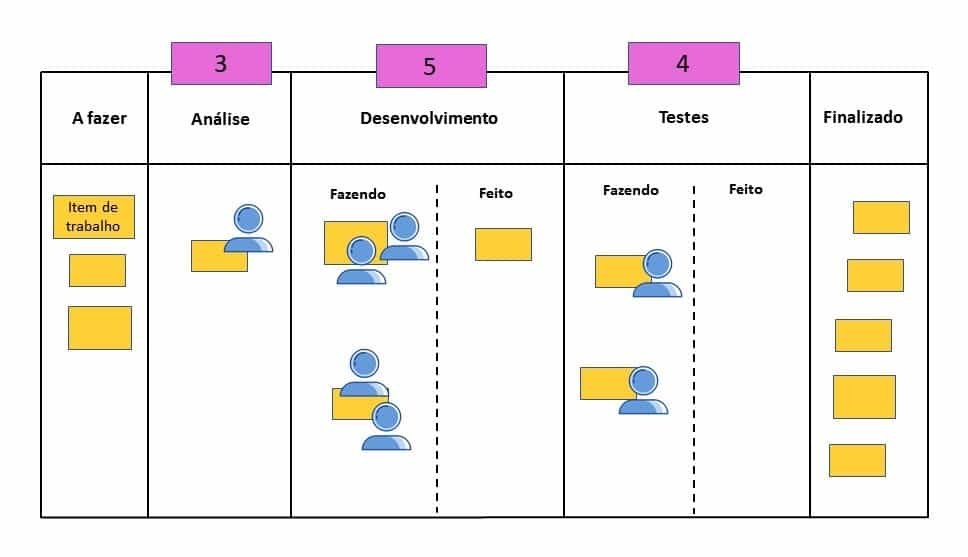
### ****Lean****

* **Características**: Originado na manufatura (Toyota), foca na eliminação de desperdícios e na entrega de valor ao cliente.
* **Práticas**: Mapeamento de fluxo de valor, melhoria contínua (Kaizen), entrega just-in-time.
* **Vantagens**: Redução de custos e desperdícios, aumento da eficiência.
* **Uso comum**: Projetos que buscam otimização de processos e eficiência.



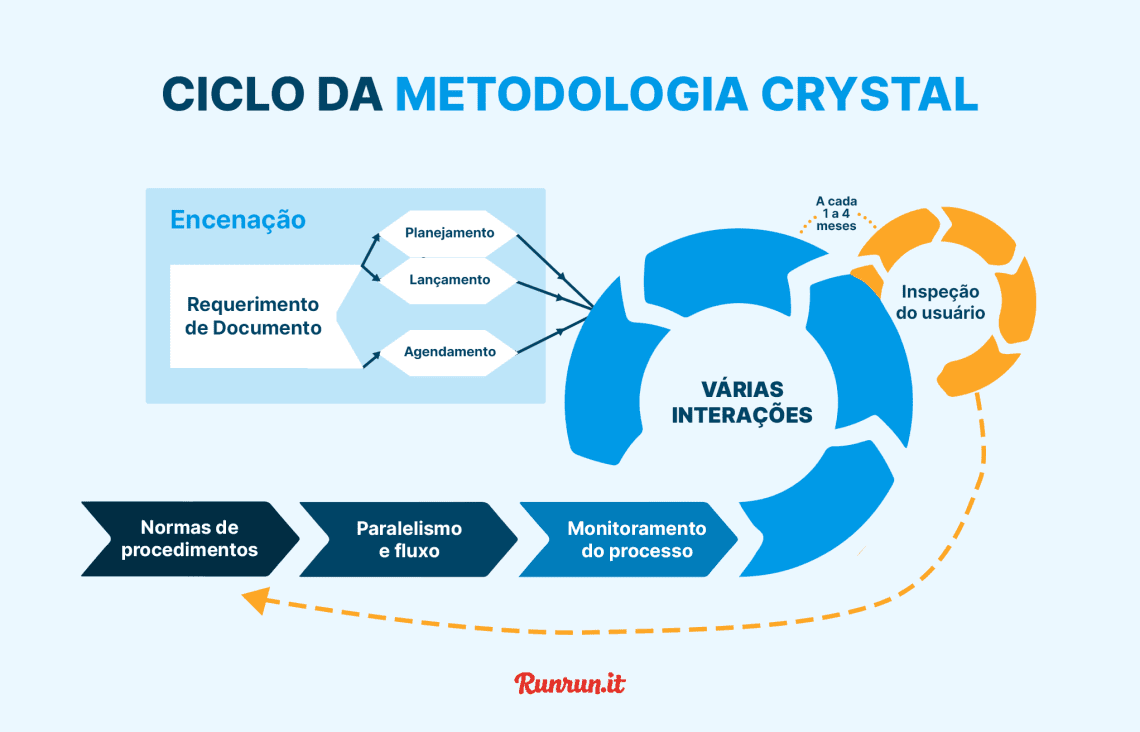
### ****Kanban****

* **Características**: Visualização do fluxo de trabalho em quadros (boards), com foco em limitar o trabalho em progresso (WIP).
* **Práticas**: Uso de colunas (ex: "A fazer", "Fazendo", "Feito") e cartões para representar tarefas.
* **Vantagens**: Flexível, melhora a visibilidade e o fluxo de trabalho.
* **Uso comum**: Equipes que precisam gerenciar demandas contínuas e priorizar tarefas.



### ****Crystal****

* **Características**: Metodologia leve e adaptável, com foco em pessoas e interações. Diferentes "cores" (Crystal Clear, Crystal Yellow, etc.) variam conforme o tamanho da equipe e complexidade do projeto.
* **Práticas**: Comunicação frequente, entrega incremental, ajustes conforme o contexto.
* **Vantagens**: Altamente adaptável, ideal para projetos pequenos ou de baixa criticidade.
* **Uso comum**: Projetos com equipes pequenas e requisitos menos rígidos.



### ****Go Horse****

* **O que é**: Uma piada, criada como crítica às más práticas no desenvolvimento de software.
* **Práticas**: Ignorar planejamento, documentação, testes e boas práticas, e contar com a sorte para resolver problemas.
* **Vantagens** : Nenhuma
* **Desvantagens**: Todas
* **Uso comum**: Ninguém usa

