

ENGENHARIA DE SOFTWARE

PROCESSOS DE SOFTWARE

Profº Ms Gustavo Molina
msc.gustavo.unip@gmail.com

Prof. Ms Gustavo Molina



- Graduado em Sistemas de Informação pelo MACKENZIE.
- Licenciado em Matemática pela UNIP.
- Pós – Graduado em Plataforma de Desenvolvimento Web pelo CLARETIANO.
- Pós – Graduado em IA pela faculdade Serra Geral
- Pós – Graduado em Gestão e Governança de Tecnologia da Informação pela UNIP
- Mestre em Engenharia Elétrica pela FEI
- Doutorando em Ciências da Educação pela Ivy Enber Christian University

Prof. Ms Gustavo Molina



<https://www.linkedin.com/in/gustavo-molina-a2798418/>



<http://lattes.cnpq.br/8512452850609937>



msc.gustavo.unip@gmail.com



<https://github.com/gustavomolina17>

Processo de Software

De modo geral, o que é um PROCESSO?

Processo de Software

- Uma receita de bolo é um processo?
- A abertura de uma conta em um banco é um processo?
- Realizar uma compra é um processo?
- Programar é um processo?



Processo de Software

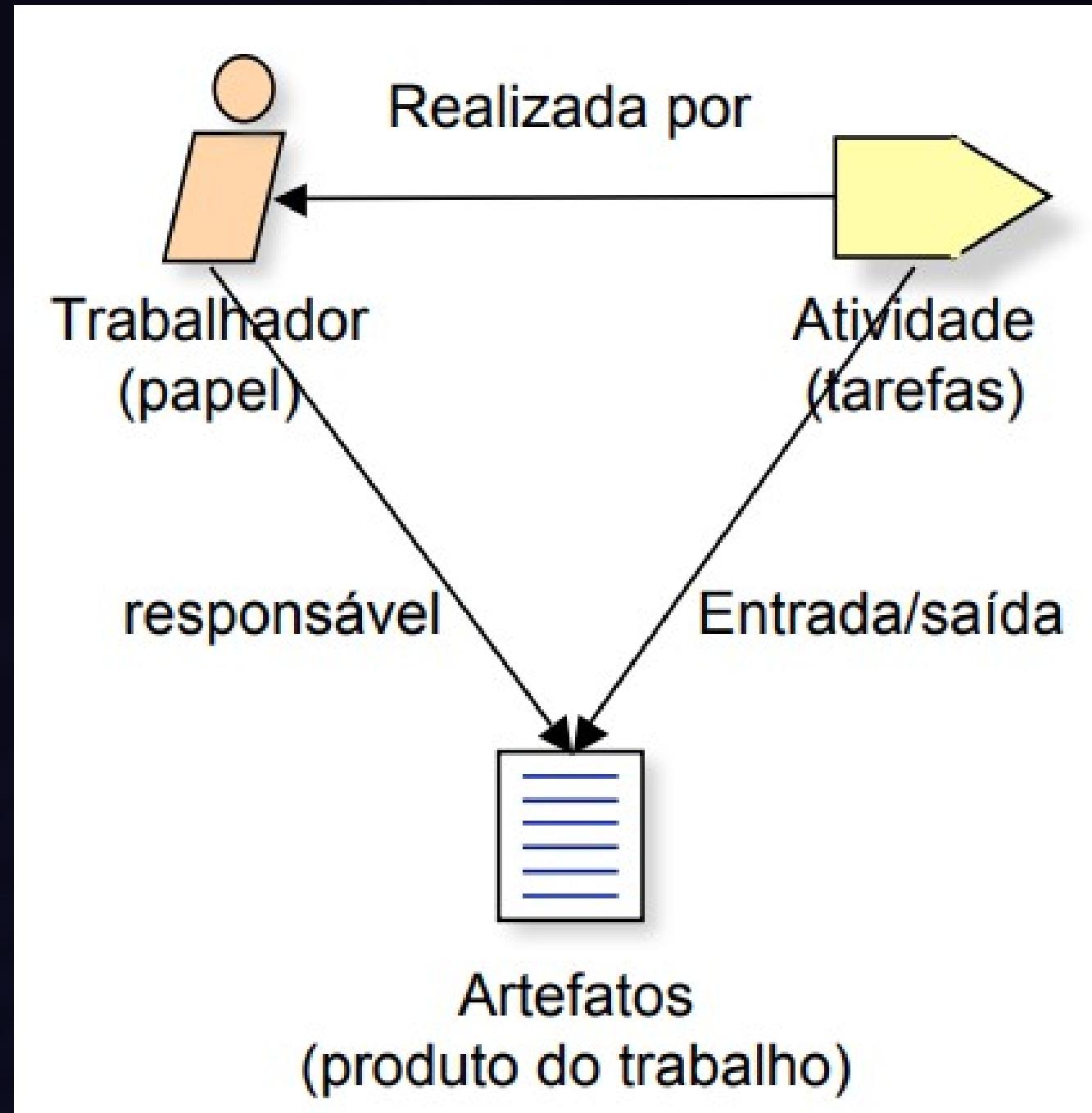
- Um processo de software pode ser visto como o conjunto de atividades, métodos, práticas e transformações que guiam pessoas na produção de software.
- Um processo eficaz deve, claramente, considerar as relações entre as atividades, os artefatos produzidos no desenvolvimento, as ferramentas e os procedimentos necessários e a habilidade, o treinamento e a motivação do pessoal envolvido.

Elementos que compõem um processo de software:

- Processo de Software Processos Atividades
 - Atv1. Realizar levantamento de requisitos.
 - Pré-Atividade: Atv0
 - Insumo: Documento Planejamento de Entrevista Produto: Documento Registro de Entrevista.
 - Recurso Humano: Analista de Sistemas Procedimento: Técnica para Realização de Entrevistas
- Pré-atividades
- Subatividades Artefatos
- Insumos Produtos
- Recursos
- Recursos Humanos Ferramentas de Software Hardware
 - Atv2. Documentar requisitos.
 - Pré-Atividade: Atv1
 - Insumo: Documento Registro de Entrevista. Produto: Especificação Textual de Requisitos. Recurso Humano: Analista de Sistemas
 - Procedimento: Roteiro para Elaboração da Especificação Textual de Requisitos.
- Procedimentos
- Métodos Técnicas Roteiros

Processo de Software

- Um processo é organizado em atividades.
- Atividades são de responsabilidade de um papel (membro da equipe).
- Um artefato (produto do trabalho) é um modelo, documento ou código produzido por uma atividade.
- Atividades devem gerar um artefato de saída, que possa ser verificado, e podem requisitar um artefato de entrada.



Processos de Software

- Na definição de um processo, deve-se definir quais são os papéis.
- Exemplos:
 - Analista
 - Arquiteto
 - Desenvolvedor
 - Testador
 - Gerente

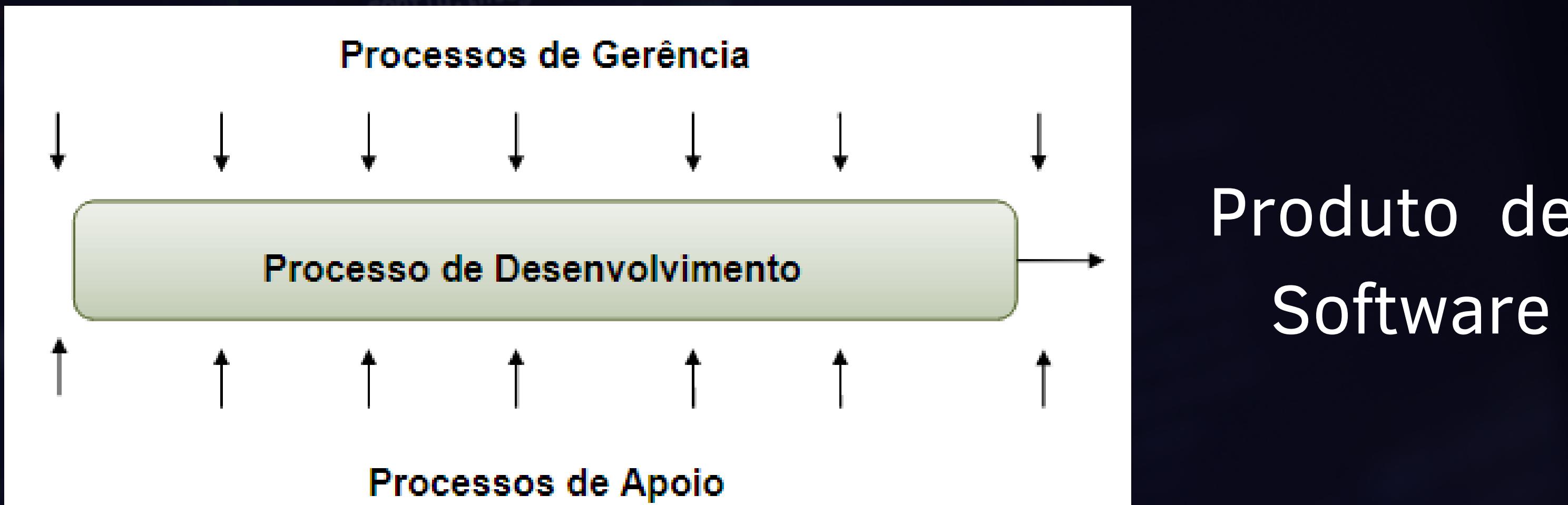


- Processos de software, em uma abordagem de Engenharia de Software, envolvem diversas atividades que podem ser classificadas quanto ao seu propósito em:
 - Atividades de Desenvolvimento (ou Técnicas ou de Construção): são as atividades diretamente relacionadas ao processo de desenvolvimento do software, ou seja, que contribuem diretamente para o desenvolvimento do produto de software a ser entregue ao cliente. Ex.: especificação e análise de requisitos, projeto e implementação.

- Atividades de Gerência de Projeto: são aquelas relacionadas ao planejamento e acompanhamento gerencial do projeto, tais como realização de estimativas, elaboração de cronogramas, análise dos riscos do projeto etc.
- Atividades de Apoio: são aquelas relacionadas principalmente com a garantia da qualidade do produto em desenvolvimento e do processo de software utilizado, tais como revisões e inspeções de produtos (intermediários ou finais) do desenvolvimento.

Atividades do processo de software:

As Atividades dos Processos de Gerência e de Apoio podem ser entendidas como “atividades guarda-chuva”, pois “cobrem” todo o processo de desenvolvimento.



Definição de Processos de Software



- Há várias maneiras diferentes de fazer um bolo de chocolate.
- Pessoas alérgicas a leite precisam de receitas que não contenham lactose.
- Pessoas diabéticas precisam de receitas que não incluem açúcar. Há várias formas diferentes de abrir uma conta bancária.
- Algumas pessoas podem abrir contas apenas para receberem seus salários. Algumas pessoas podem abrir contas com vantagens especiais, como limite de crédito.

Processos de software podem ter diversas definições distintas.

O que deve ser considerado para definir um processo de software?

- Características da aplicação (domínio do problema, tamanho, complexidade etc)
 - Tecnologia a ser adotada na sua construção (paradigma de desenvolvimento,
 - linguagem de programação, mecanismo de persistência etc)
 - Organização onde o produto será desenvolvido
 - Características da equipe
 - Estabilidade dos requisitos
 - Outros
-
- Modelos de Ciclo de Vida fornecem o arcabouço para a definição do processo.

Modelo de Ciclo de Vida de Software ou Modelo de Processo de Software

- Representação abstrata de um esqueleto de processo, incluindo tipicamente algumas atividades principais, a ordem de precedência entre elas e, opcionalmente, artefatos requeridos e produzidos.
- É um importante ponto de partida para definir como o projeto deve ser conduzido, mas a sua adoção não é o suficiente para guiar e controlar um projeto de software na prática.

Modelo de Ciclo de Vida de Software ou Modelo de Processo de Software

- Geralmente envolve as seguintes fases: Análise e Especificação de Requisitos, Projeto,
- Implementação, Testes, Entrega e Implantação, Operação, Manutenção.
- Os principais modelos de ciclo de vida podem ser agrupados em três categorias
- principais: modelos sequenciais, modelos incrementais e modelos evolutivos.

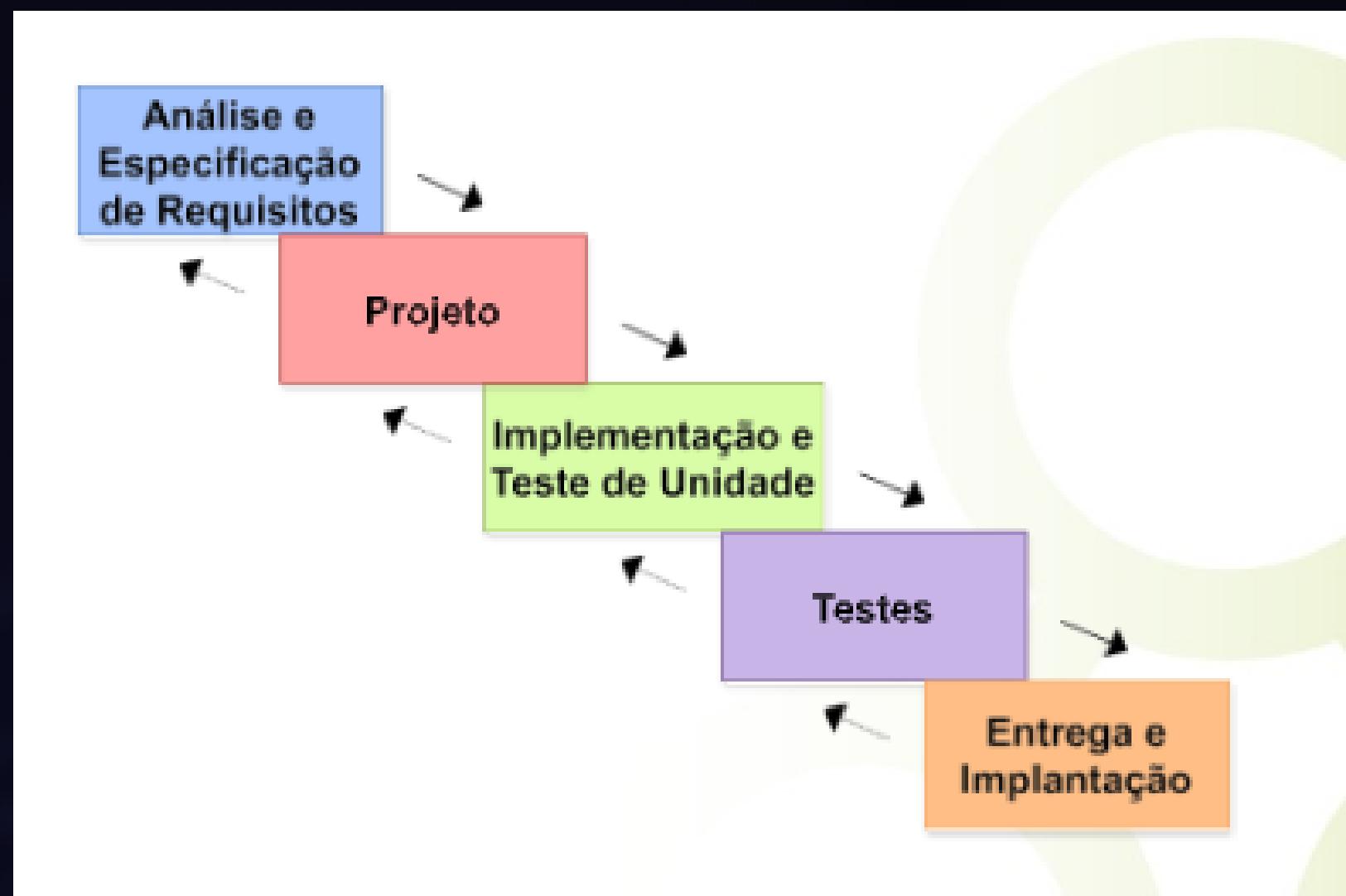
Modelos Sequenciais - Modelo Cascata (Modelo de Ciclo de Vida Clássico)

- É o modelo de ciclo de vida mais antigo e mais amplamente usado.
- Indicado para problemas bastante pequenos e bem definidos, onde os desenvolvedores conhecem bem o domínio do problema e os requisitos podem ser claramente estabelecidos



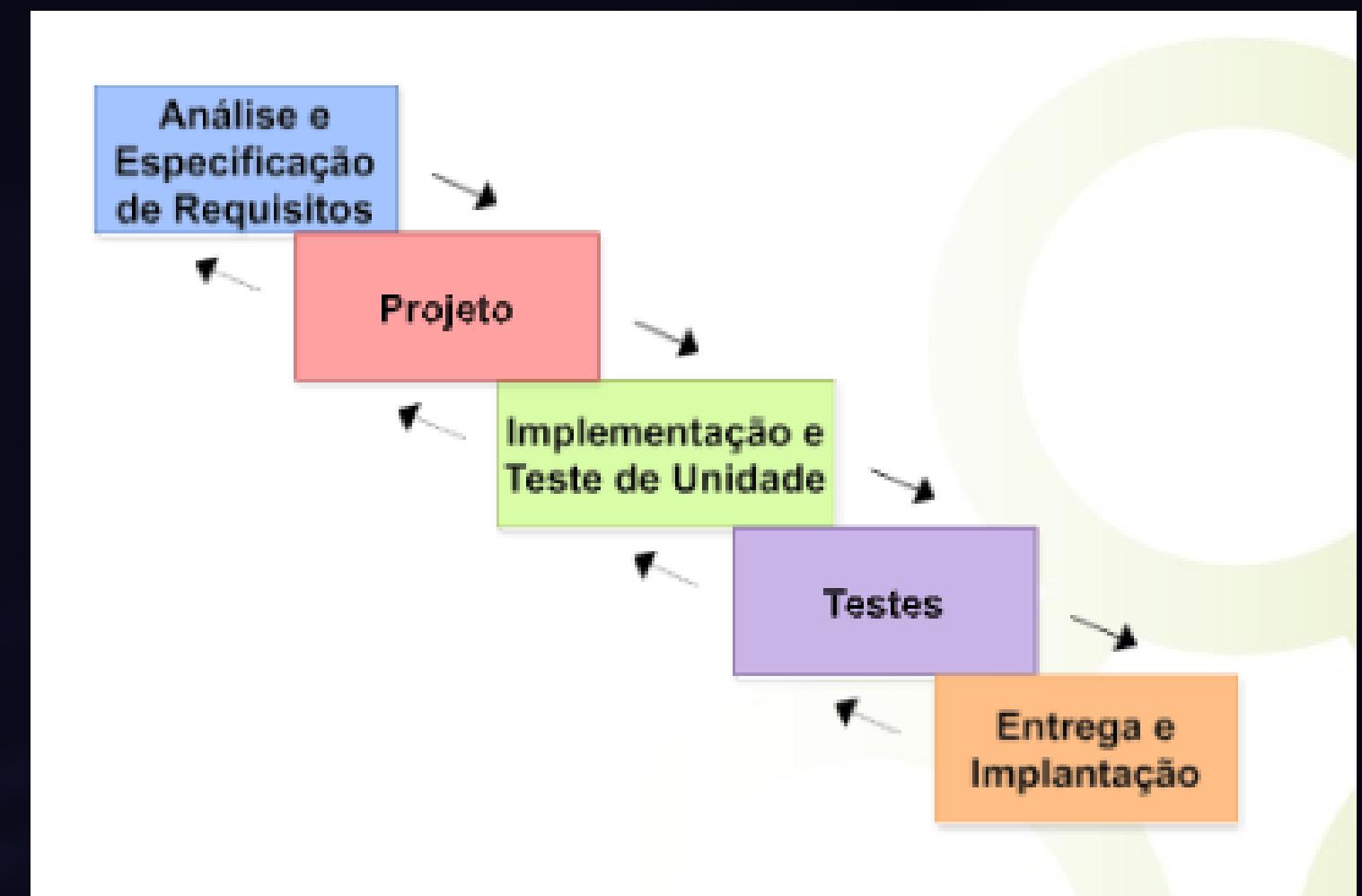
Modelos Sequenciais - Modelo Cascata (Modelo de Ciclo de Vida Clássico)

- Uma fase só deve ser iniciada após a conclusão daquela que a precede.
- Uma vez que, na prática, essas fases se sobrepõem de alguma forma, geralmente, permite-se um retorno à fase anterior para a correção de erros encontrados.



Modelos Sequenciais - Modelo Cascata (Modelo de Ciclo de Vida Clássico)

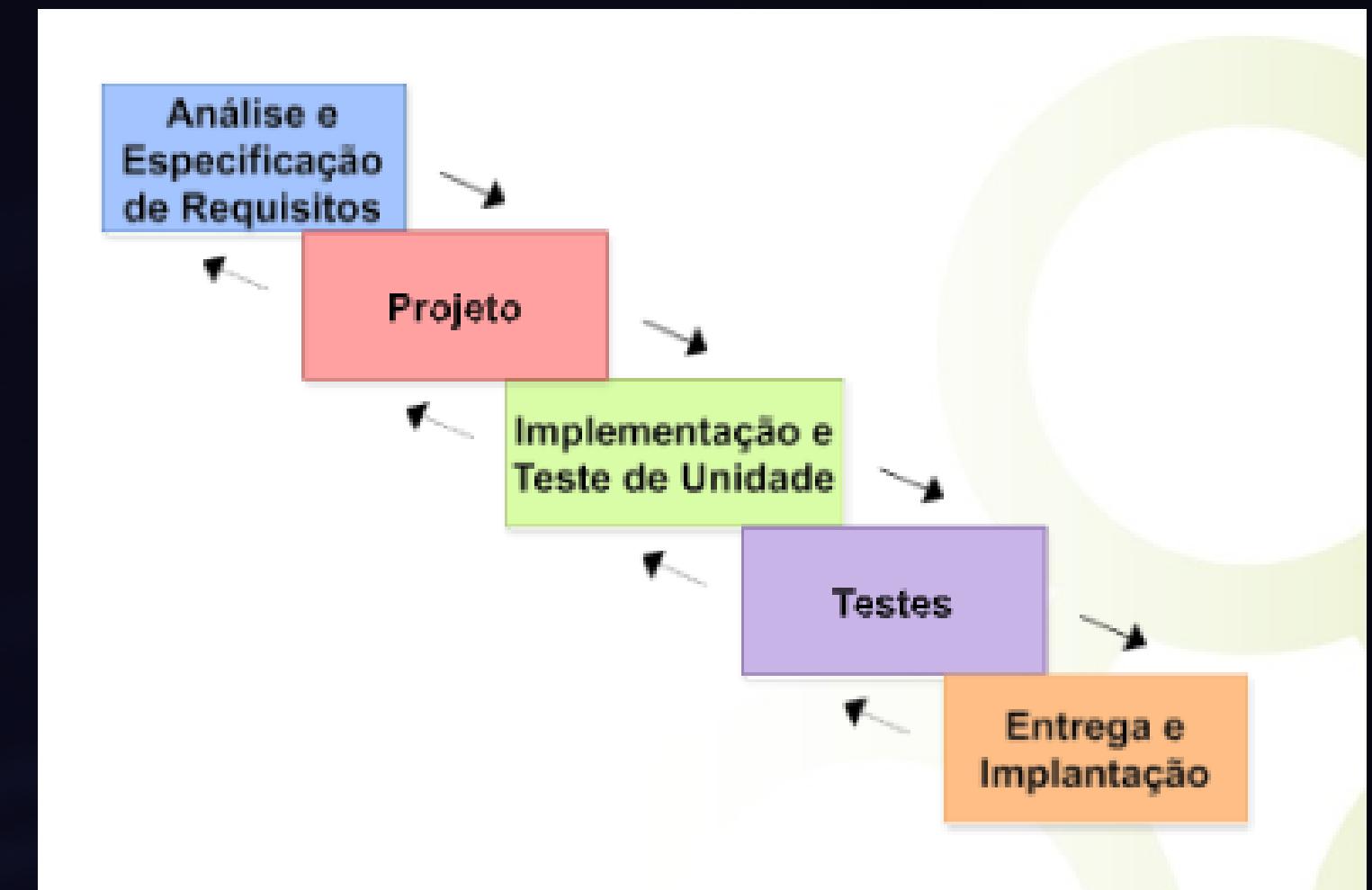
- A entrega do sistema completo ocorre somente ao final da fase de Entrega e Implantação.
- O uso de revisões ao fim de cada fase permite o envolvimento do usuário.
- Cada fase serve como uma base aprovada e documentada para o passo seguinte, facilitando bastante a gerência de configuração.



Modelos Sequenciais - Modelo Cascata (Modelo de Ciclo de Vida Clássico)

Problemas:

- Projetos reais muitas vezes não seguem o fluxo sequencial que o modelo propõe.
- Os requisitos devem ser estabelecidos de maneira completa, correta e clara logo no início de um projeto. A aplicação deve, portanto, ser entendida pelo desenvolvedor desde o início do projeto.



Modelos Sequenciais - Modelo Cascata (Modelo de Ciclo de Vida Clássico)

Problemas:

- *O usuário precisa ser paciente. Uma versão operacional do software não estará disponível até o final do projeto.*
- *A introdução de certos membros da equipe, tais como projetistas e programadores, é frequentemente adiada desnecessariamente.*
- *A natureza linear do ciclo de vida clássico leva a “estados de bloqueio” nos quais alguns membros da equipe do projeto precisam esperar que outros membros da equipe completem tarefas dependentes.*

Modelos Sequenciais - Modelo em V

- *Variação do Cascata que procura enfatizar a estreita relação entre as atividades de teste (teste de unidade, teste de integração, teste de sistema e teste de aceitação) e as demais fases do processo.*
- *A conexão entre os lados direito e esquerdo do modelo em V implica que, caso sejam encontrados problemas em uma atividade de teste, a correspondente fase do lado esquerdo e suas fases subsequentes podem ter de ser executadas novamente para corrigir ou melhorar esses problemas.*

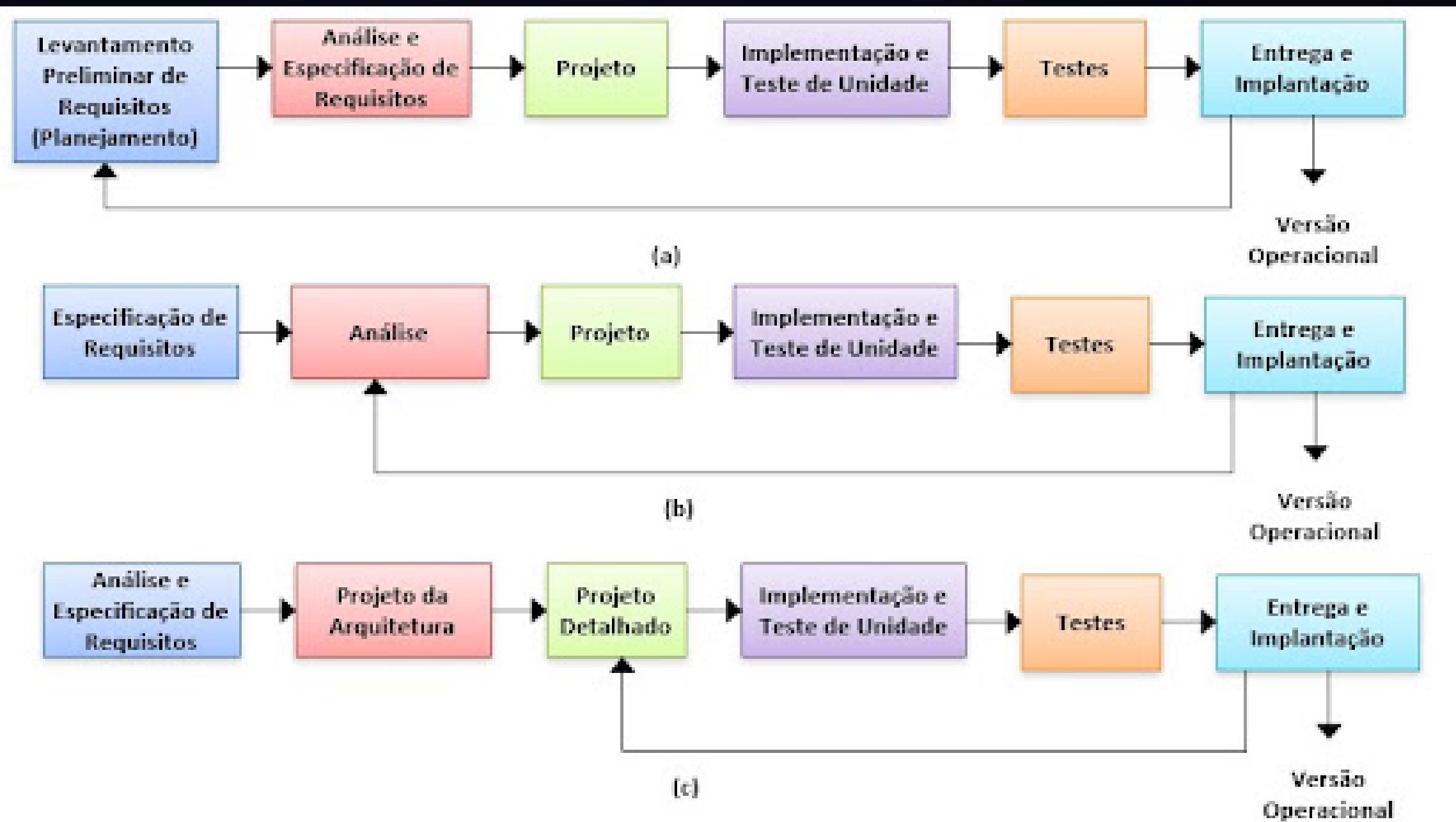
Modelos Sequenciais - Modelo em V

- *Testes de unidade e integração devem garantir que todos os aspectos do projeto do sistema foram implementados corretamente no código.*
- *Teste de sistema busca verificar se o sistema atende aos requisitos definidos na especificação.*
- *Testes de aceitação, conduzidos tipicamente pelos usuários e clientes, validam os requisitos, confirmando que os requisitos corretos foram*



Modelos Incrementais - Modelo Incremental Clássico

Seu princípio fundamental é que, a cada ciclo ou iteração, uma versão operacional do sistema será produzida e entregue para uso ou avaliação detalhada do cliente.



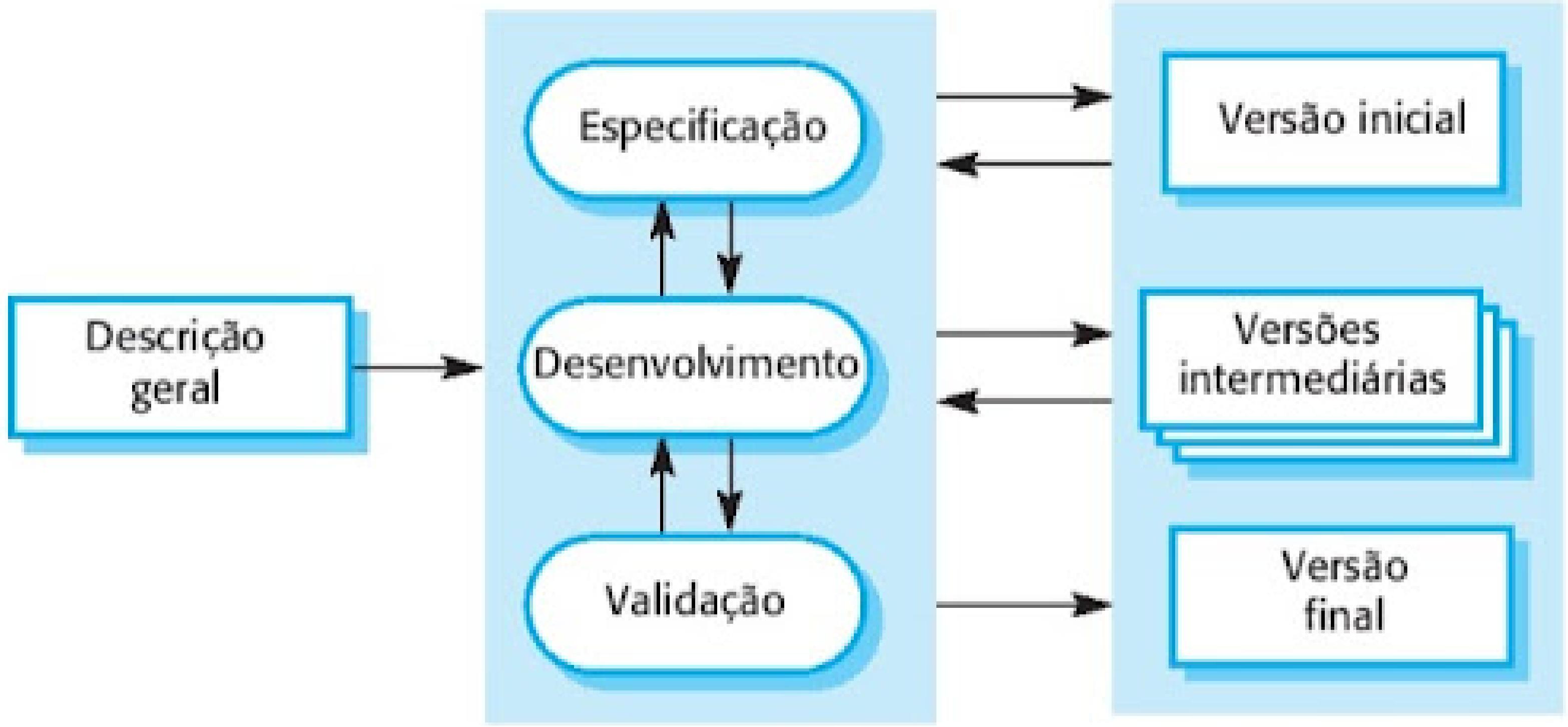
Modelos Incrementais - Modelo Incremental Clássico

- *Vantagens:*
- *Menor custo e menos tempo são necessários para se entregar a primeira versão;*
- *Os riscos associados ao desenvolvimento de um incremento são menores, devido ao seu tamanho reduzido;*
- *O número de mudanças nos requisitos pode diminuir devido ao curto tempo de desenvolvimento de um incremento.*
- *Desvantagens:*
- *Se os requisitos não são tão estáveis ou completos quanto se esperava, alguns incrementos podem ter de ser bastante alterados;*
- *A gerência do projeto é mais complexa, sobretudo quando a divisão em subsistemas inicialmente feita não se mostrar boa.*

Modelos Incrementais - Modelo Incremental Clássico

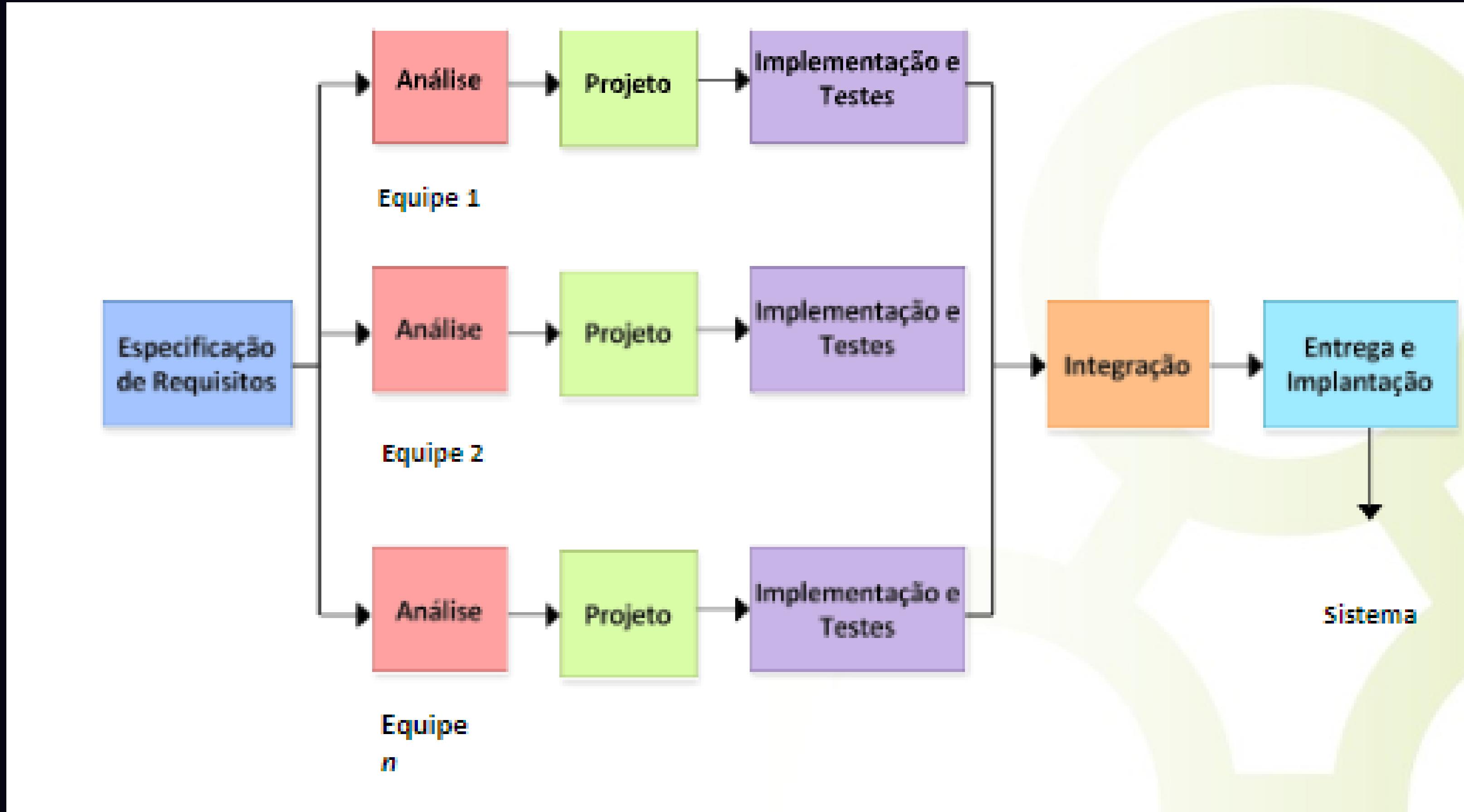
- *Os problemas de desenvolvimento incremental tornam-se particularmente críticos para sistemas grandes, complexos e de longa vida útil, onde equipes diferentes desenvolvem partes diferentes do sistema.*
- *Os sistemas grandes precisam de um framework ou arquitetura estáveis e as responsabilidades das diferentes equipes que trabalham em partes do sistema precisam ser claramente definidas com relação a essa arquitetura. Isso deve ser planejado com antecedência, em vez de desenvolvido de forma incremental.*
- *(SOMMERVILLE, 2019)*

Atividades simultâneas



Modelos Incrementais - RAD (Rapid Application Development)

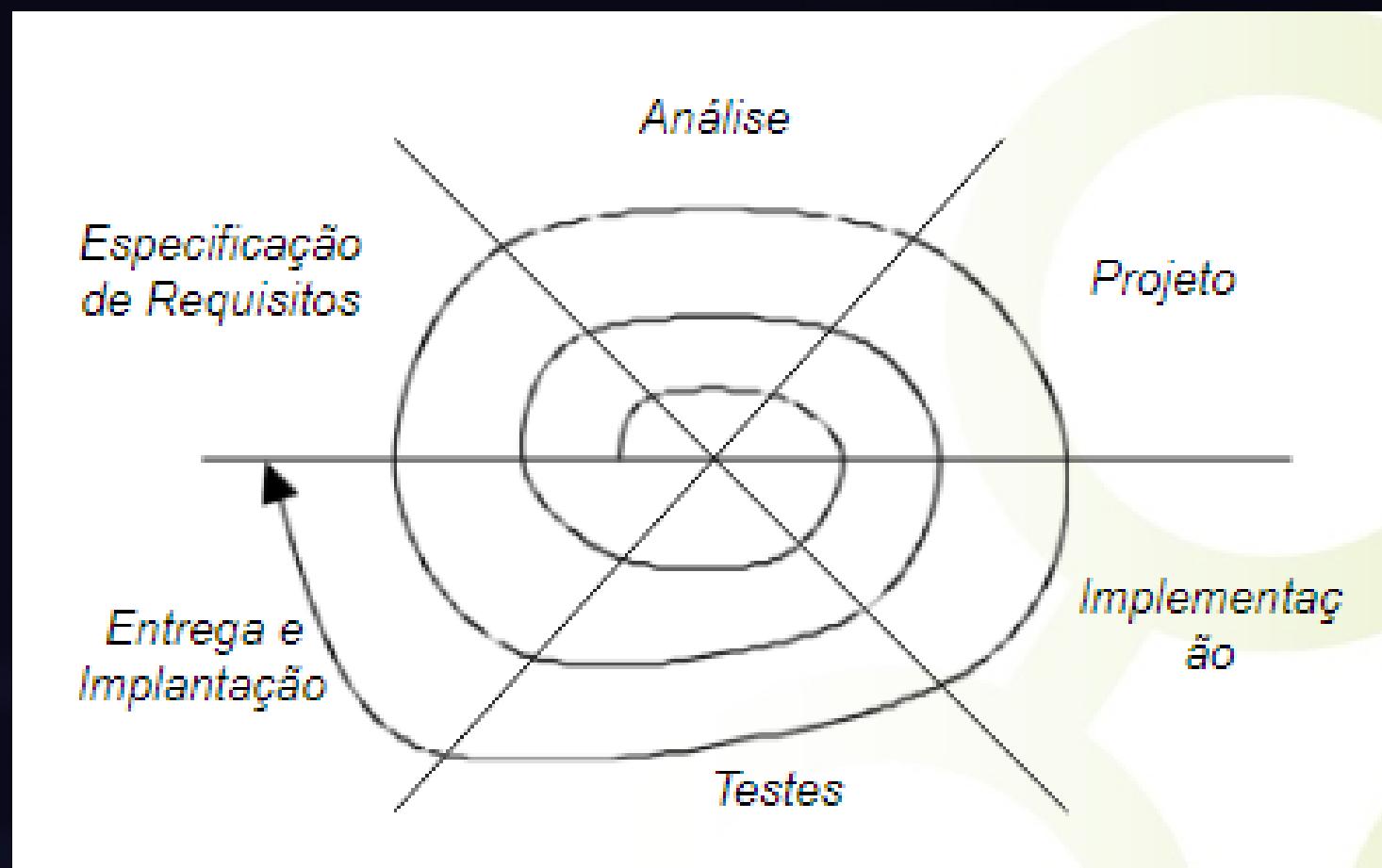
- *Prima por um ciclo de desenvolvimento curto (tipicamente de até 90 dias).*
- *Os incrementos são desenvolvidos em paralelo por equipes distintas e apenas uma única entrega é feita.*
- *Exige disponibilidade de equipes.*
- *Os requisitos têm de ser bem definidos, o escopo do projeto tem de ser restrito e o sistema modular.*



Modelos Evolutivos - Modelo Espiral

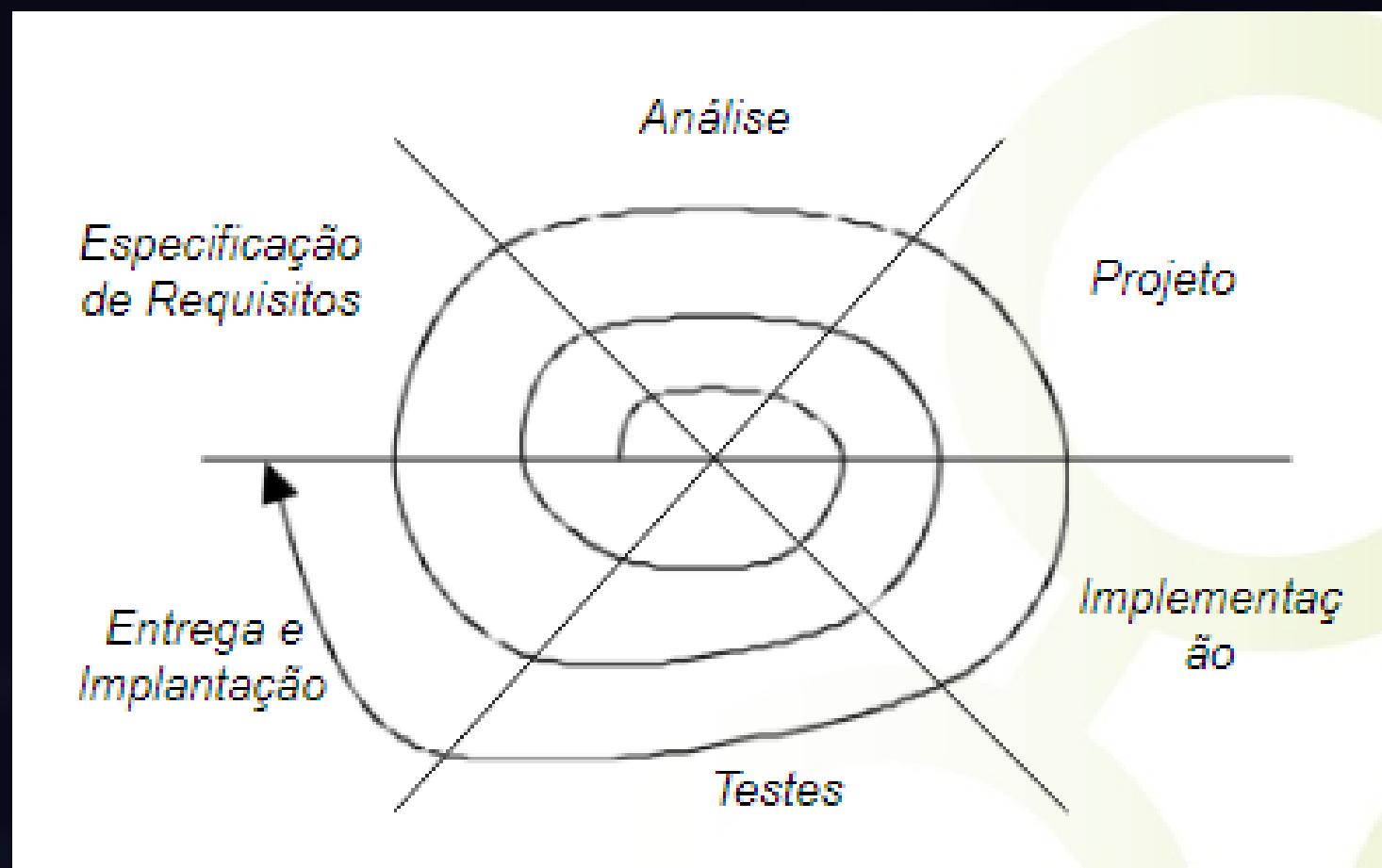
O sistema é desenvolvido em ciclos, sendo que nos primeiros ciclos nem sempre todas as atividades são resultante do primeiro ciclo pode ser uma especificação do produto ou um estudo de viabilidade).

- As passadas subsequentes ao longo da espiral podem ser usadas para desenvolver protótipos, chegando progressivamente a versões operacionais do software, até se obter o produto completo.



Modelos Evolutivos - Modelo Espiral

- A cada ciclo, o planejamento deve ser revisto com base no feedback do cliente, ajustando, inclusive, o número de iterações planejadas.
- Pode ser difícil convencer clientes, especialmente em situações envolvendo contrato, que a abordagem evolutiva é gerenciável.



O Processo Unificado - RUP (Rational Unified Process)

- Um modelo de processo de software genérico que apresenta o desenvolvimento de software como uma atividade interativa de quatro fases. (SOMMERVILLE, 2011)
- É mais que um modelo de processo de desenvolvimento. É uma abordagem completa para o desenvolvimento de software, incluindo, além de um modelo de processo de software, a definição detalhada de responsabilidades (papéis), atividades, artefatos e fluxos de trabalho, dentre outros.

O Processo Unificado - RUP (Rational Unified Process)

- É um bom exemplo de modelo de processo híbrido. Ele reúne elementos de todos os modelos de processo genéricos, ilustra as boas práticas de especificação e projeto e oferece suporte à prototipagem e entrega incremental.

O RUP é normalmente descrito de três perspectivas:

- 1. Uma perspectiva dinâmica, que mostra as fases do modelo ao longo do tempo.
- 2. Uma perspectiva estática, que mostra as atividades do processo que são executadas.
- 3. Uma perspectiva prática, que sugere boas práticas a serem utilizadas durante o processo.(SOMMERVILLE, 2011)

O Processo Unificado - RUP (Rational Unified Process)

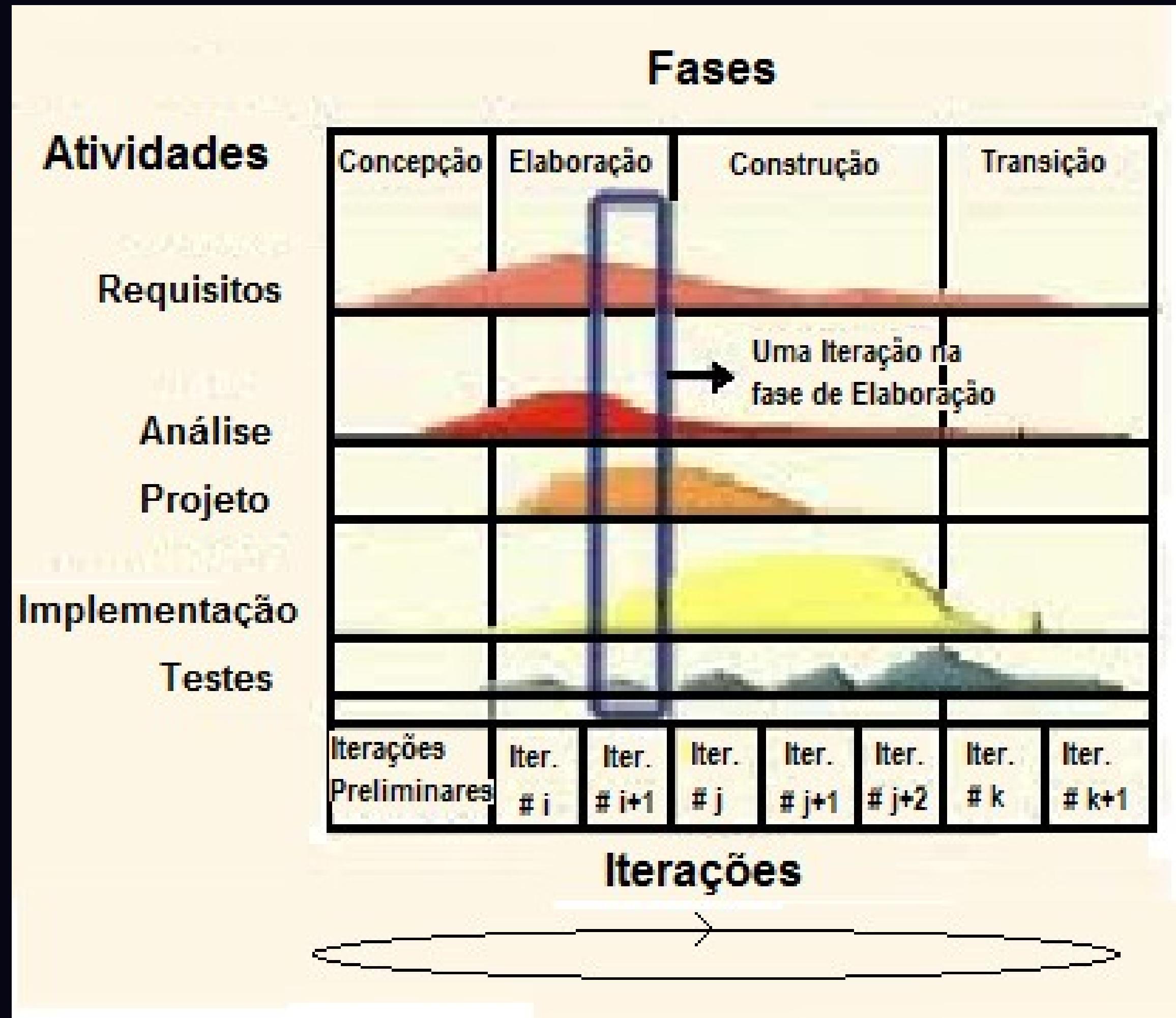
- No que se refere às fases, o RUP considera quatro fases no desenvolvimento de software:
 - Concepção;
 - Elaboração;
 - Construção;
 - Transição;
- A fase de Concepção visa estabelecer um bom entendimento do escopo do projeto, obtendo um entendimento de alto nível dos requisitos a serem tratados.
- Nesta fase o foco está na comunicação com o cliente para a identificação de requisitos e nas atividades de planejamento.

O Processo Unificado - RUP (Rational Unified Process)

- No que se refere às atividades do processo de desenvolvimento, o foco é o levantamento de requisitos, ainda que atividades de modelagem conceitual (análise) e para a elaboração de um esboço bastante inicial da arquitetura do sistema (projeto) possam ser realizadas. Protótipos podem ser construídos para apoiar a comunicação com o cliente.
- Elaboração: os objetivos desta fase são analisar o domínio do problema, estabelecer a arquitetura do sistema, desenvolver o plano do projeto e identificar seus maiores riscos.
- Assim, em termos do processo de desenvolvimento, o foco são as atividades de análise e projeto.

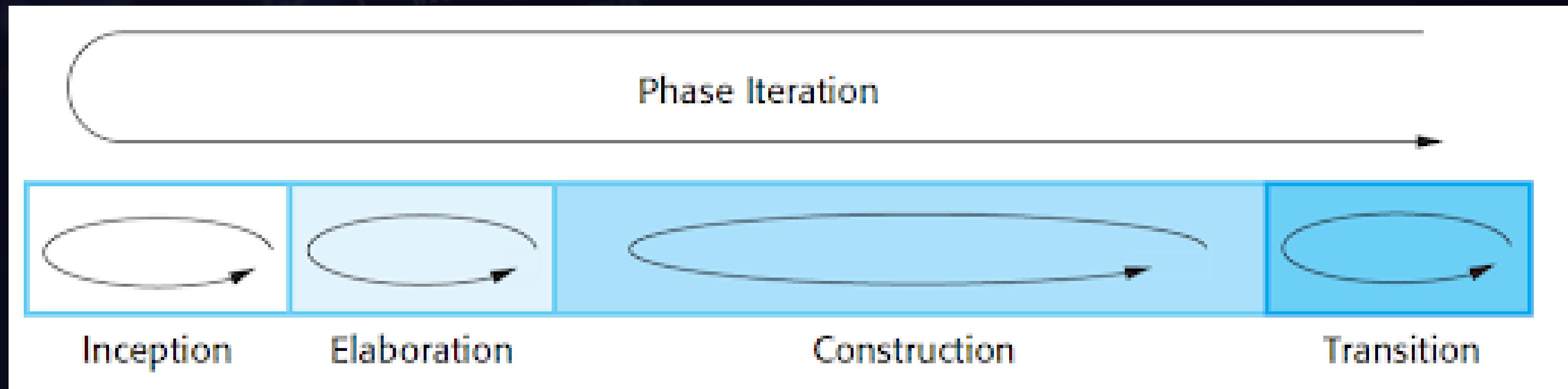
O Processo Unificado - RUP (Rational Unified Process)

- Construção: envolve o projeto detalhado de componentes, sua implementação e testes.
- Nesta fase, os componentes do sistema são desenvolvidos, integrados e testados.
- Ao concluir esta fase, deve-se ter um sistema de software em funcionamento e a documentação associada que está pronta para ser entregue aos usuários.
(SOMMERVILLE, 2011)
- Transição: o propósito desta fase é fazer a transição do sistema do ambiente de desenvolvimento para o ambiente de produção.
- São feitos testes de sistema e de aceitação e a entrega do sistema aos seus usuários.
- Ao concluir esta fase, deve-se ter um sistema de software documentado que esteja funcionando corretamente em seu ambiente operacional. (SOMMERVILLE, 2011)



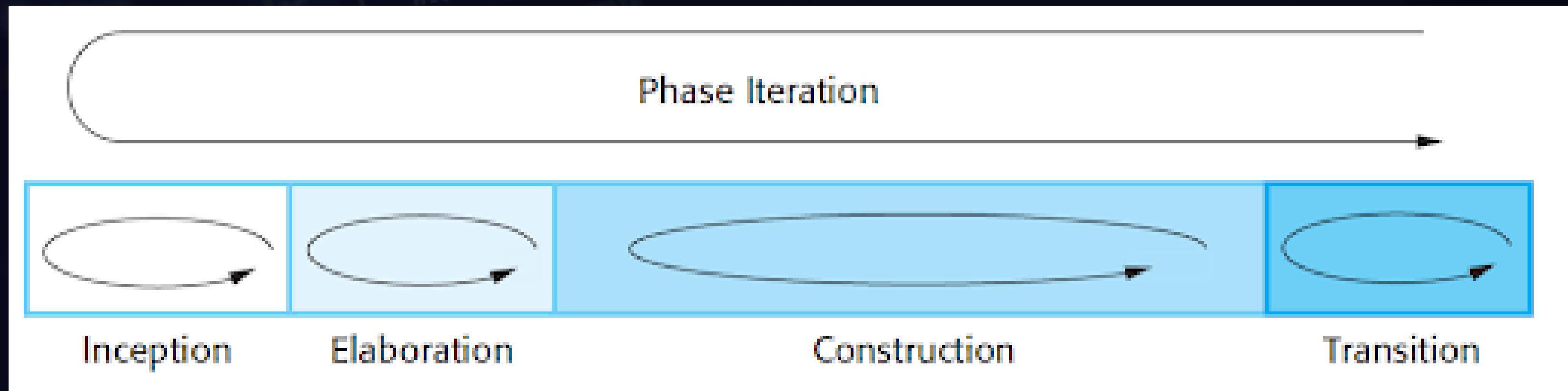
O Processo Unificado - RUP (Rational Unified Process)

- A iteração no RUP é suportada de duas maneiras: Cada fase pode ser executada de forma iterativa com os resultados desenvolvidos de forma incremental. Além disso, todo o conjunto de fases também pode ser executado de forma incremental. (SOMMERVILLE, 2011)



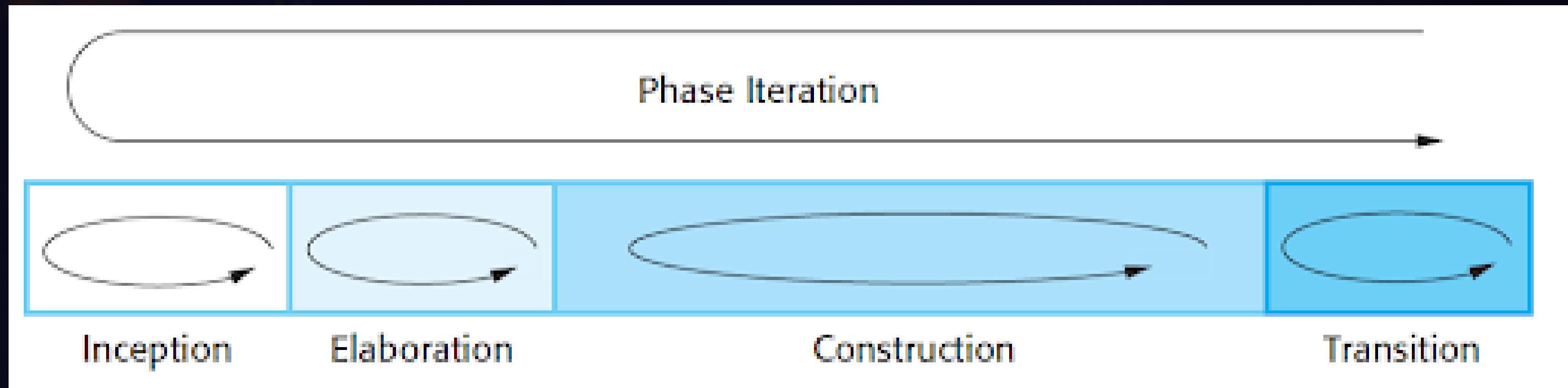
O Processo Unificado - RUP (Rational Unified Process)

- As atividades do processo de desenvolvimento são distribuídas ao longo de uma iteração, em função do foco da fase correspondente. (SOMMERVILLE, 2011)



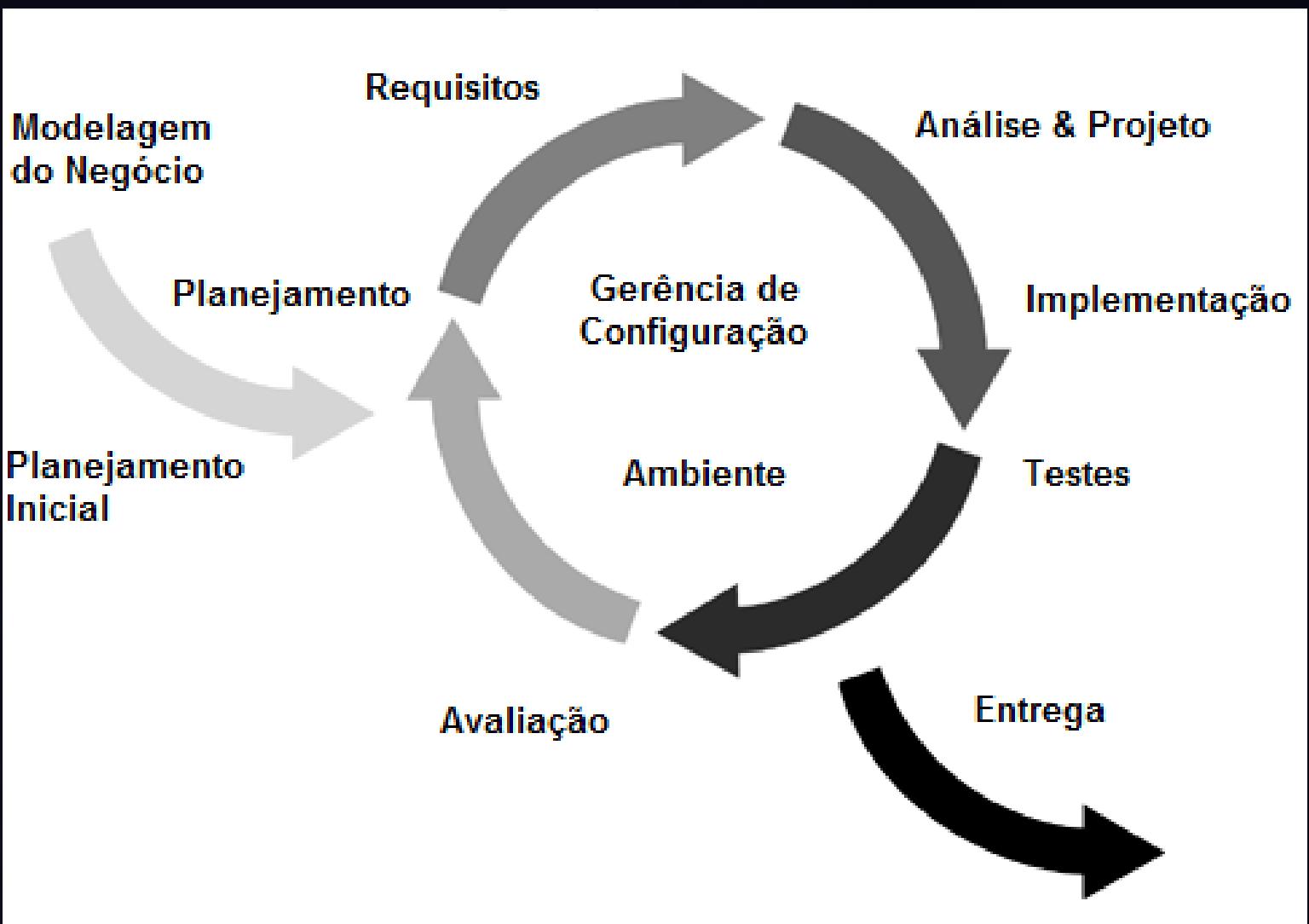
O Processo Unificado - RUP (Rational Unified Process)

- Por exemplo, em uma iteração que ocorra no final da fase de elaboração, tipicamente são realizadas atividades de especificação de requisitos, análise, projeto, implementação e testes. Já em uma iteração típica da fase de concepção, essencialmente são realizadas atividades de levantamento de requisitos, com algum trabalho de modelagem (análise). Eventualmente, se um protótipo for desenvolvido, pode haver algum trabalho de implementação e testes. (SOMMERVILLE, 2011)



O Processo Unificado - RUP (Rational Unified Process)

- A vantagem de apresentar visões dinâmica e estática é que as fases do processo de desenvolvimento não estão associadas a fluxos de trabalho específicos.
- Em princípio, pelo menos, todos os fluxos de trabalho do RUP podem estar ativos em todos os estágios do processo. Nas fases iniciais do processo, a maior parte do esforço provavelmente será gasta em fluxos de trabalho, como modelagem de negócios e requisitos e, nas fases posteriores, em testes e implantação.



O Processo Unificado - RUP (Rational Unified Process)

- A perspectiva prática no RUP descreve boas práticas de engenharia de software que são recomendadas para uso no desenvolvimento de sistemas. Seis práticas recomendadas fundamentais são recomendadas:
- 1. Desenvolver software iterativamente: Planeje incrementos do sistema com base nas prioridades do cliente e desenvolva os recursos do sistema de mais alta prioridade no início do processo de desenvolvimento.
- 2. Gerenciar os requisitos: Documente explicitamente os requisitos do cliente e acompanhe as mudanças nesses requisitos. Analise o impacto das mudanças no sistema antes de aceitá-las.
- 3. Usar arquiteturas baseadas em componentes: Estruture a arquitetura do sistema em componentes.
- Fonte: (SOMMERVILLE, 2011)

O Processo Unificado - RUP (Rational Unified Process)

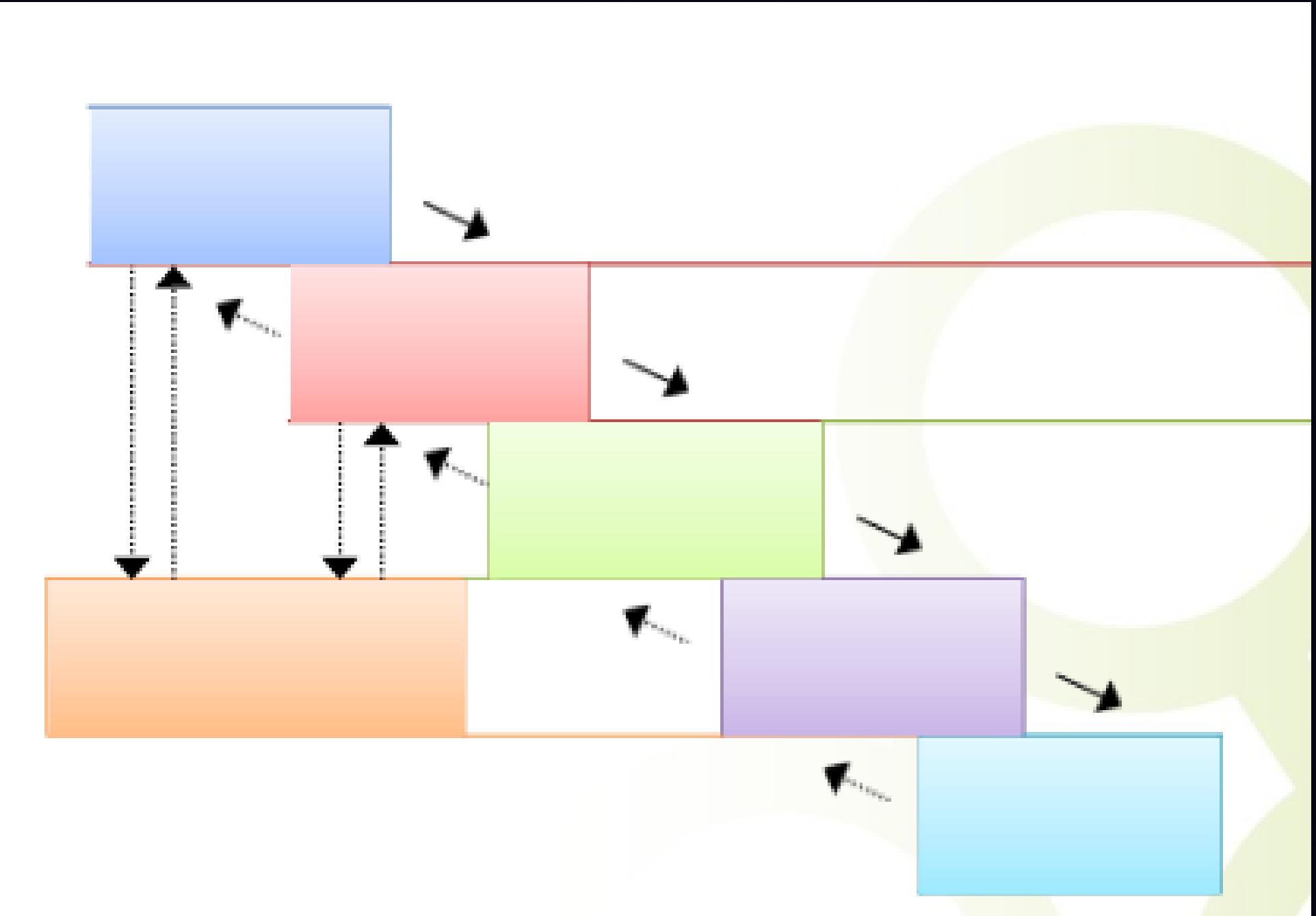
- A perspectiva prática no RUP descreve boas práticas de engenharia de software que são recomendadas para uso no desenvolvimento de sistemas. Seis práticas recomendadas fundamentais são recomendadas:
- 4. Modelar software visualmente: Use modelos UML gráficos para apresentar visualizações estáticas e dinâmicas do software.
- 5. Verificar a qualidade do software: Certifique-se de que o software atenda aos padrões de qualidade da organização.
- 6. Controlar as alterações no software: Gerencie as alterações no software usando um sistema de gerenciamento de alterações e procedimentos e ferramentas de gerenciamento de configuração.
- Fonte: (SOMMERVILLE, 2011)

O Processo Unificado - RUP (Rational Unified Process)

- O RUP não é um processo adequado para todos os tipos de desenvolvimento, por exemplo, desenvolvimento de software integrado. No entanto, ele representa uma abordagem que potencialmente combina três modelos de processo genéricos.
- As inovações mais importantes no RUP são a separação de fases e fluxos de trabalho, e o reconhecimento de que a implantação de software no ambiente de um usuário faz parte do processo.
- As fases são dinâmicas e têm objetivos.
- Os fluxos de trabalho são estáticos e são atividades técnicas que não estão associadas a uma única fase, mas podem ser usadas ao longo do desenvolvimento para atingir os objetivos de cada fase.
- Fonte: (SOMMERVILLE, 2011)

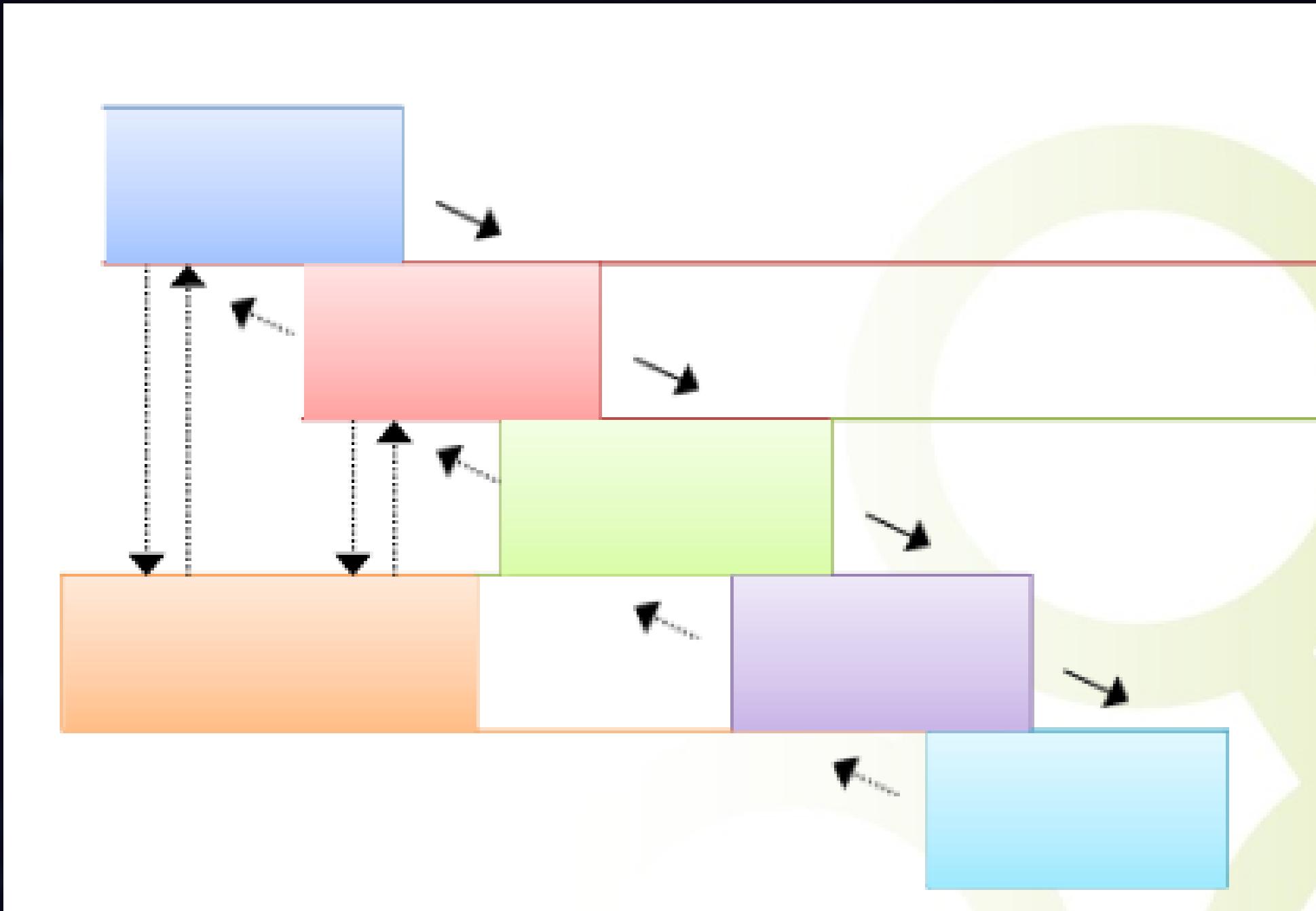
Prototipação

- Muitas vezes, clientes têm em mente um conjunto geral de objetivos para um sistema de software, mas não são capazes de identificar claramente as funcionalidades (requisitos) ou informações o sistema terá de prover.



Prototipação

- A prototipação é uma técnica para ajudar engenheiros de software e clientes a entender o que está sendo construído quando os requisitos não estão claros.
- Pode ser aplicada no contexto de qualquer modelo de processo (na figura: modelo Cascata com Prototipação)



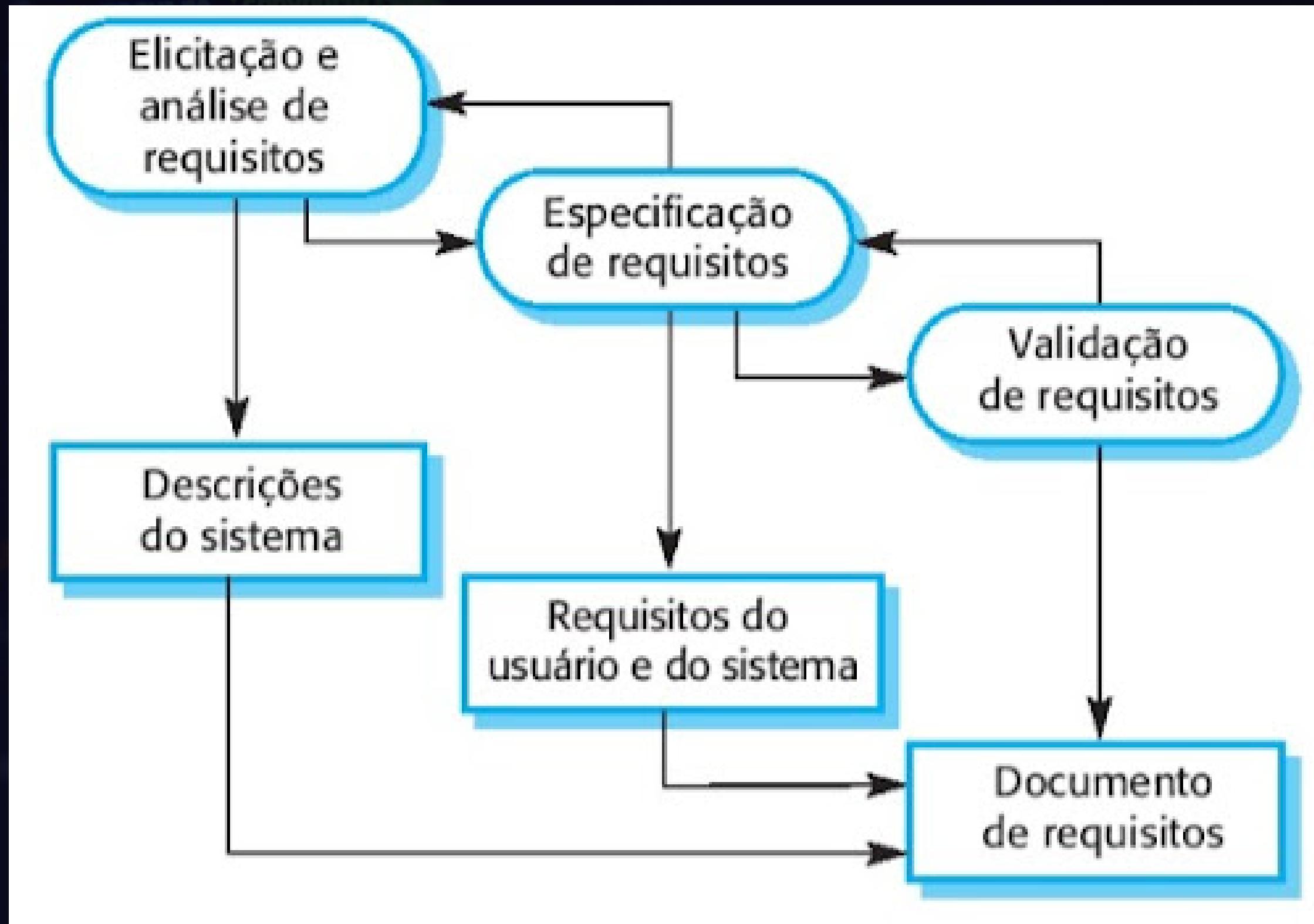
Prototipação



Visão Geral do Processo de Desenvolvimento

- O processo de desenvolvimento de software engloba as atividades que contribuem diretamente para o desenvolvimento do produto de software a ser entregue ao cliente, incluindo a sua documentação. De maneira geral, o processo de desenvolvimento de software envolve as seguintes atividades: Análise e Especificação de Requisitos, Projeto, Implementação, Testes, Entrega e Implantação do Sistema.
- Mas o que é feito em cada etapa do processo?

Análise e Especificação de Requisitos

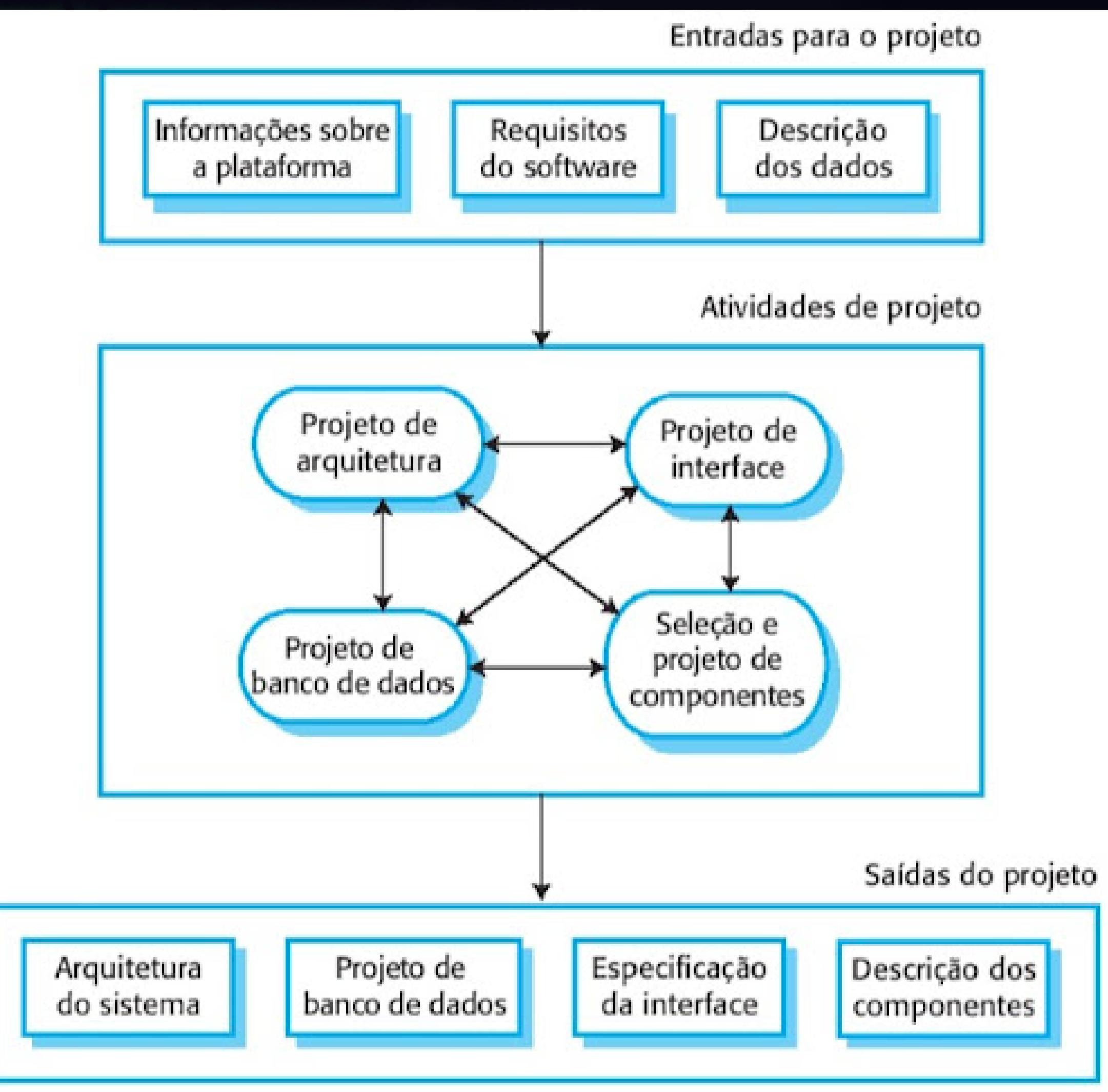


Projeto

- A fase de Projeto é responsável por incorporar requisitos tecnológicos aos requisitos essenciais do sistema e, portanto, requer que a plataforma de implementação seja conhecida.
- Basicamente, envolve duas grandes etapas: projeto da arquitetura do sistema e o projeto detalhado. O objetivo da primeira etapa é definir a arquitetura geral do software, tendo por base o modelo construído na fase de análise de requisitos. Essa arquitetura deve descrever a estrutura de nível mais alto da aplicação e identificar seus principais componentes.
- O propósito do projeto detalhado é detalhar o projeto do software para cada componente identificado na etapa anterior. Os componentes de software devem ser sucessivamente refinados em níveis maiores de detalhamento, até que possam ser codificados e testados.

Projeto

```
5      abort("The Rails en  
6      require 'spec_help  
7      require 'rspec/rail  
8  
9      require 'capybara'  
10     require 'capybara  
11  
12     Copybara.javasc  
13     Category.delete  
14     Shoulda::Match  
15     config.inte  
16     with.test  
17     with.lib  
18     end  
19     end  
20  
21     # Add addi  
22  
23     # Requir  
24     # spec/s  
25     # run o  
26     # in
```

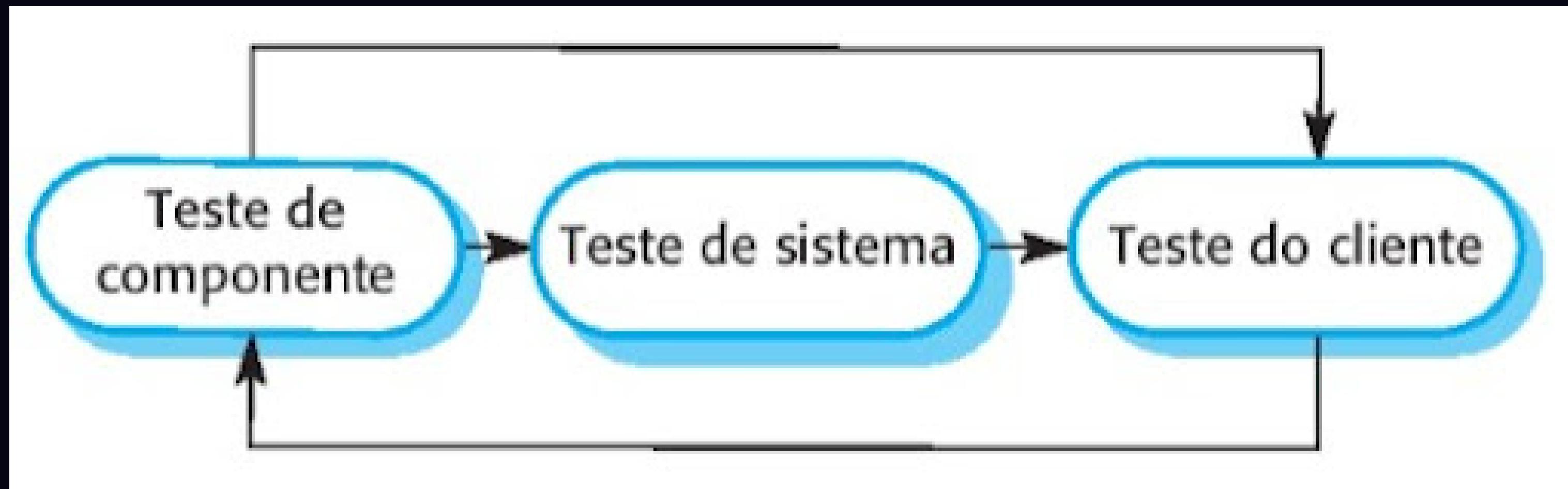


Implementação e teste de Unidade

- O projeto deve ser traduzido para uma forma passível de execução pela máquina. A fase de implementação realiza esta tarefa, isto é, cada unidade de software do projeto detalhado é implementada.

Teste

- A fase de Testes inclui diversos níveis de testes, a saber, teste de unidade, teste de integração e teste de sistema. Inicialmente, cada unidade de software implementada deve ser testada. A seguir, os diversos componentes devem ser integrados sucessivamente até se obter o sistema. Finalmente, o sistema como um todo deve ser testado.



Teste

- O teste de cliente ou teste de aceitação às vezes é chamado de 'teste alfa'. Os sistemas personalizados são desenvolvidos para um único cliente. O processo de teste alfa continua até que o desenvolvedor do sistema e o cliente concordem que o sistema entregue é uma implementação aceitável dos requisitos.
- Quando um sistema deve ser comercializado como um produto de software, um processo de teste denominado 'teste beta' é frequentemente usado. O teste beta envolve a entrega de um sistema a vários clientes em potencial que concordam em usar aquele sistema. Eles relatam problemas aos desenvolvedores do sistema. Isso expõe o produto para uso real e detecta erros que podem não ter sido previstos pelos integradores do sistema. Após esse feedback, o sistema é modificado e liberado para outros testes beta ou para venda geral.

Entrega e Implantação

- Uma vez testado e aprovado, o software deve ser colocado em produção. Para tal, contudo, é necessário treinar os usuários, configurar o ambiente de produção e, muitas vezes, converter bases de dados.
- O propósito da fase de Implantação e Entrega é disponibilizar o software para o cliente, garantindo que o mesmo satisfaz os requisitos estabelecidos. Isto requer a instalação do software e a condução de testes de aceitação. Quando o software tiver demonstrado prover as capacidades requeridas, ele pode ser aceito e a operação iniciada.

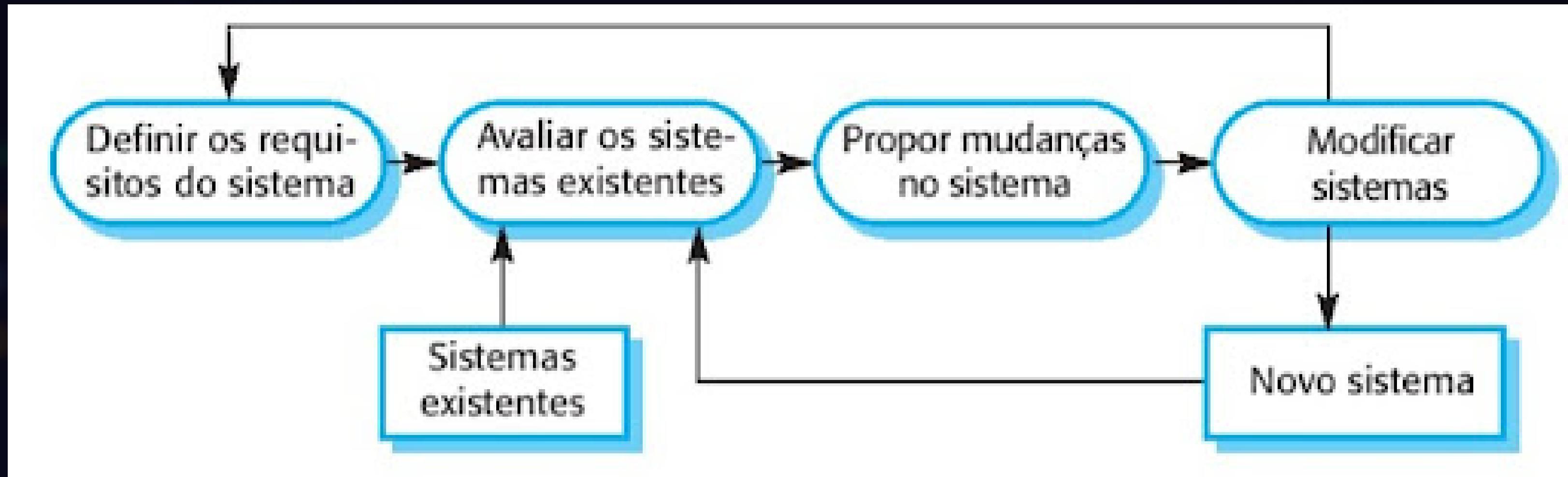
Evolução de Software

- Encerrado o processo de desenvolvimento, com a entrega do sistema ao cliente, diz-se que o sistema está em operação, quando o software é utilizado pelos usuários no ambiente de produção.
- Contudo, indubitavelmente, o software sofrerá mudanças após ter sido entregue para o cliente.

Evolução de Software

- Alterações ocorreram porque erros foram encontrados, porque o software precisa ser adaptado para acomodar mudanças em seu ambiente externo, ou porque o cliente necessita de funcionalidade adicional ou aumento de desempenho. Muitas vezes, dependendo do tipo e porte da manutenção necessária, a manutenção pode requerer a definição de um novo processo, onde cada uma das fases do processo de desenvolvimento é reaplicada no contexto de um software existente ao invés de um novo.

Evolução de Software



Referências

- SOMMERVILLE, I. Engenharia de Software. 9. ed. São Paulo: Pearson Prentice Hall, 2011
- SOMMERVILLE, I. Engenharia de Software. 10. ed. São Paulo: Pearson Prentice Hall, 2019