Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - IFSP
Câmpus Jacareí

Tecnologia em Análise e Desenvolvimento de Sistemas - ADS

2º Semestre de 2023

Engenharia de Software – JCRESW1

Prof. Lineu Mialaret

Aula 6: Processos de Desenvolvimento de Software (1)

Histórico de Desenvolvimento de Software

- 1950s: Primeiros compiladores e interpretadores.
- 1960s: Primeiro grande software relatado na literatura, o sistema OS/360.
  - Mais de 1000 desenvolvedores.
  - Custo estimado de US$ 50.000.000,00 por ano.
- 1968: Crise do software, quando surge a Engenharia de Software.
- 1970s:
  - *Lower-CASE tools* (programação, depuração, colaboração).
  - Ciclo de Vida Cascata (*Waterfall*).
  - Desenvolvimento Estruturado.

Histórico de Desenvolvimento de Software (cont.)

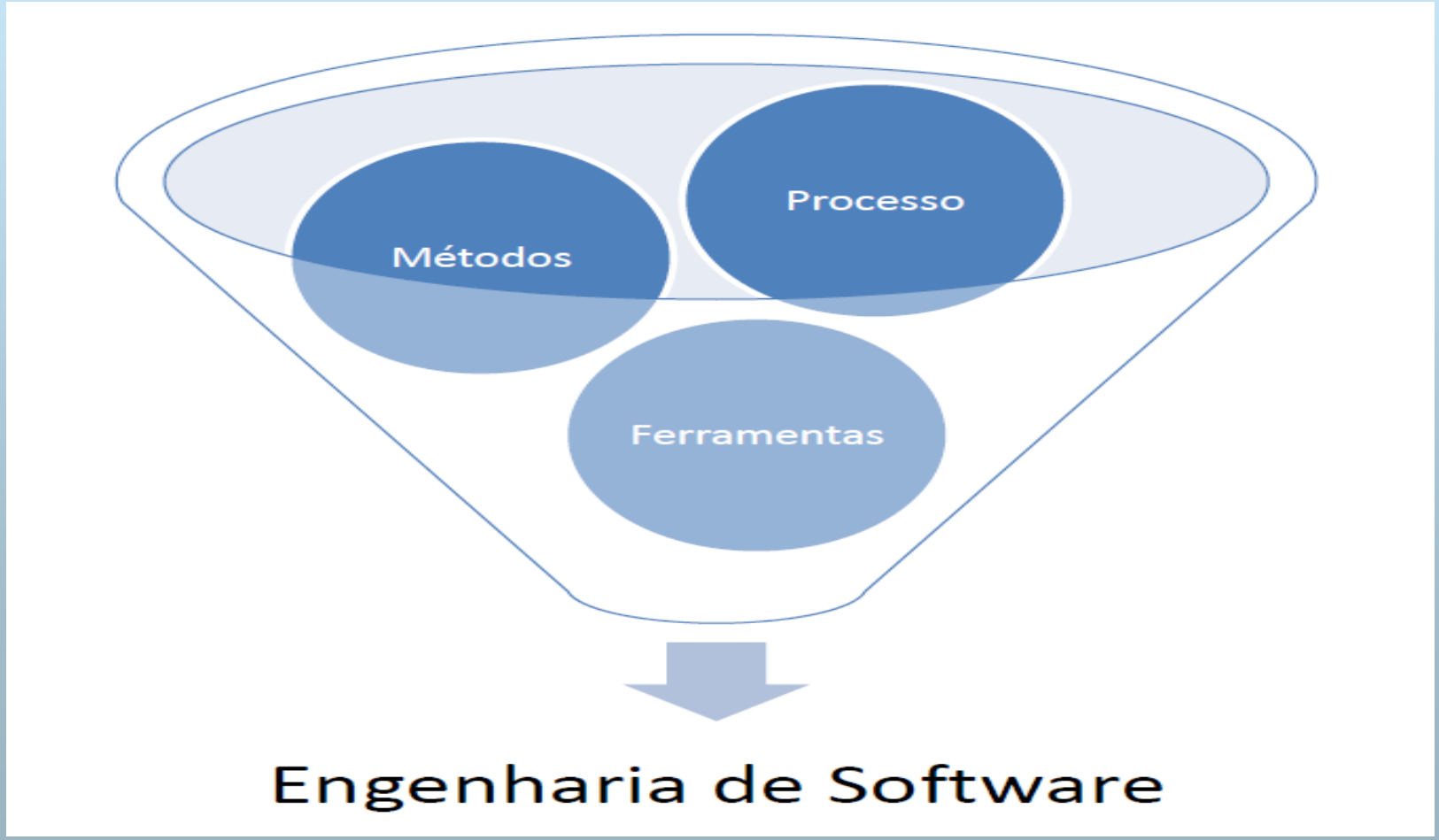
- 1980s:
  - Ciclo de Vida Espiral.
  - Desenvolvimento Orientado a Objetos.
  - Controle de Versões.
  - Testes.
- 1990s: *Upper-CASE tools*
  - Processos.
  - Modelagem.
- 2000s:
  - Métodos Ágeis.
  - Desenvolvimento Dirigido por Modelos (*Model Driven Development* - MDD).

Histórico de Desenvolvimento de Software (cont.)

- Atualmente:
  - DevOPs
  - Continuous (CI/CD)
  - …

Desenvolvimento de Software

- Pode-se abstrair a Engenharia de Software quanto ao desenvolvimento de software nos seguintes elementos.



Processo de Software

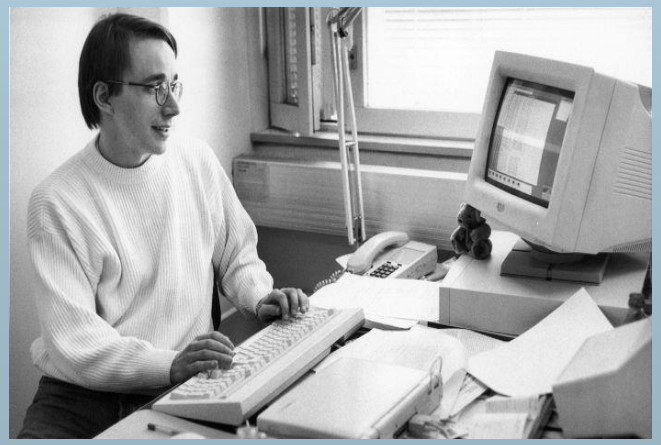
- A produção de um carro em uma fábrica de automóveis segue um processo bem definido.
  - No início as chapas de aço são cortadas e prensadas, para ganhar a forma de portas, tetos e capôs.
  - Depois, o carro é pintado e instalam-se painel, bancos, cintos de segurança e a fiação.
  - Por fim, instala-se a parte mecânica, incluindo motor, suspensão e freios.

Processo de Software (cont.)

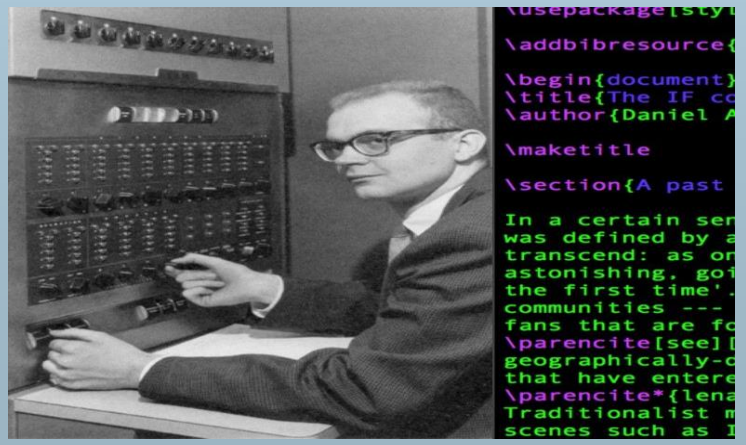
- Assim como carros, software também é produzido de acordo com um processo, embora certamente menos mecânico e mais dependente de esforço intelectual.
  - Um processo de desenvolvimento de software define um conjunto de passos, tarefas, eventos e práticas que devem ser seguidos por desenvolvedores de software, na produção de um sistema.
  - Ele define os passos gerais para o desenvolvimento e manutenção do software.
  - Serve como uma estrutura de encadeamento de métodos e ferramentas.

Processo de Software (cont.)

- Alguns críticos de processos de software costumam fazer a seguinte pergunta:
  - Por que preciso seguir um processo?
- E complementam, perguntando sobre qual processo:
  - Linus Torvalds usou na implementação do sistema operacional Linux?
  - Donald Knuth usou na implementação do formatador de textos TeX?



Linus Torvalds



Donald Knuth

Processo de Software (cont.)

- Tanto o Linux (no seu início) e o TeX são projetos individuais, liderados por um único desenvolvedor.
  - Nesses casos, a adoção de um processo é menos importante.
  - Ou, dizendo de outra forma, o processo em tais projetos é pessoal, composto pelos princípios, práticas e decisões tomadas pelo seu único desenvolvedor; e que terão impacto apenas sobre ele mesmo.
  - Estes projetos representam exceções individuais!

Processo de Software (cont.)

- Porém, os sistemas de software atuais são por demais complexos para serem desenvolvidos por uma única pessoa.
  - Por isso, casos de sistemas desenvolvidos por heróis solitários serão cada vez mais raros.
  - Na prática, os sistemas modernos (interesse deste curso) são desenvolvidos em equipes (ainda que reduzidas).
- E essas equipes, para produzir software com qualidade, produtividade e no prazo, precisam de um ordenamento, mesmo que mínimo.

Processo de Software (cont.)

- Por isso, empresas dão tanto valor a processos de software.
  - Eles são o instrumento de que as empresas dispõem para coordenar, motivar, organizar e avaliar o trabalho de seus desenvolvedores, de forma a garantir que eles trabalhem com produtividade e produzam sistemas alinhados com os objetivos da organização.
  - Sem um processo, mesmo que simplificado e leve, como os processos ágeis (que serão estudados mais tarde) há o risco de que os times de desenvolvimento passem a trabalhar de forma descoordenada, gerando produtos sem valor para o negócio da empresa.

Processo de Software (cont.)

- Por fim, processos são importantes não apenas para a empresa, mas também para os desenvolvedores, pois permitem que eles tomem consciência das tarefas e resultados que se esperam deles.
  - Sem um processo, os desenvolvedores podem se sentir perdidos, trabalhando de forma errática e sem alinhamento com os demais membros do time.

Processo de Software (cont.)

- Um Processo de Software é um conjunto de atividades técnicas, colaborativas e gerenciais relacionadas que levam à produção de um sistema de software.
  - Define os passos gerais para o desenvolvimento e manutenção do software.
  - Serve como uma estrutura de encadeamento de métodos e ferramentas

Processo de Software (cont.)

- Quanto ao Processo de Software.
  - Existem muitos tipos diferentes de sistemas de software e não há um método universal de engenharia de software que seja aplicável a todos eles.
  - Consequentemente, não existem processos de software universalmente aplicáveis.
  - Há alguns processos pré-fabricados disponibilizados.
  - O processo utilizado nas diferentes empresas depende do tipo de software que está sendo desenvolvido, dos requisitos do cliente e das habilidades das pessoas que o desenvolvem.

Métodos e Ferramentas (cont.)

- Métodos.
  - São os "*how to's*" de como fazer um passo específico do processo.
- Ferramentas.
  - Automatizam o processo e os métodos.

Métodos e Ferramentas (cont.)

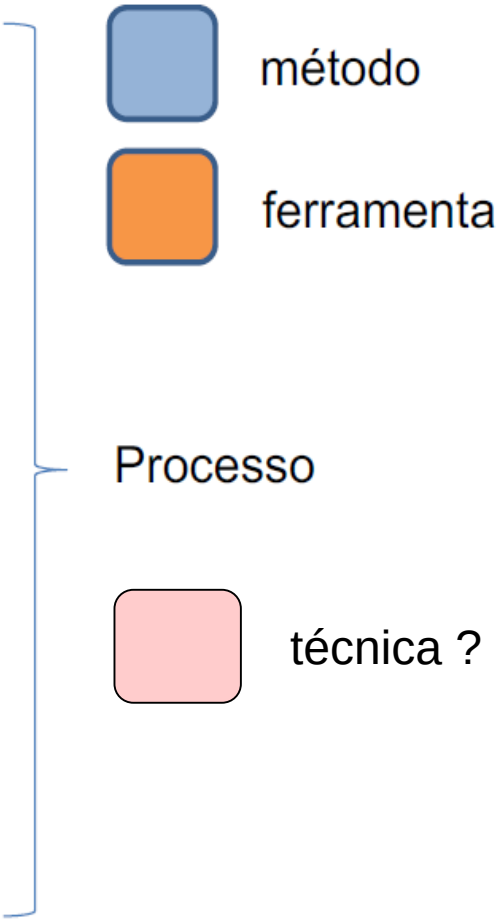
1. Coloque em uma panela funda o leite condensado, a margarina e o chocolate em pó.
2. Cozinhe [no fogão] em fogo médio e mexa sem parar com uma colher de pau.
3. Cozinhe até que o brigadeiro comece a desgrudar da panela.
4. Deixe esfriar bem, então unte as mãos com margarina, faça as bolinhas e envolva-as em chocolate granulado.

O que é processo, método ou ferramenta?

Construção de um doce de brigadeiro (1).

Métodos e Ferramentas (cont.)

1. **Coloque** em uma **panela** funda o leite condensado, a margarina e o chocolate em pó.
2. **Cozinhe** [no **fogão**] em fogo médio e **mexa** sem parar com uma **colher de pau**.
3. **Cozinhe** até que o brigadeiro comece a desgrudar da **panela**.
4. Deixe esfriar bem, então **unte** as mãos com margarina, **faça as bolinhas** e **envolva**-as em chocolate granulado.



Construção de um doce de brigadeiro (2).

Métodos e Ferramentas (cont.)

Cuidado com o "desenvolvimento guiado por ferramentas"

- É importante usar a ferramenta certa para o problema
- O problema não deve ser adaptado para a ferramenta disponível

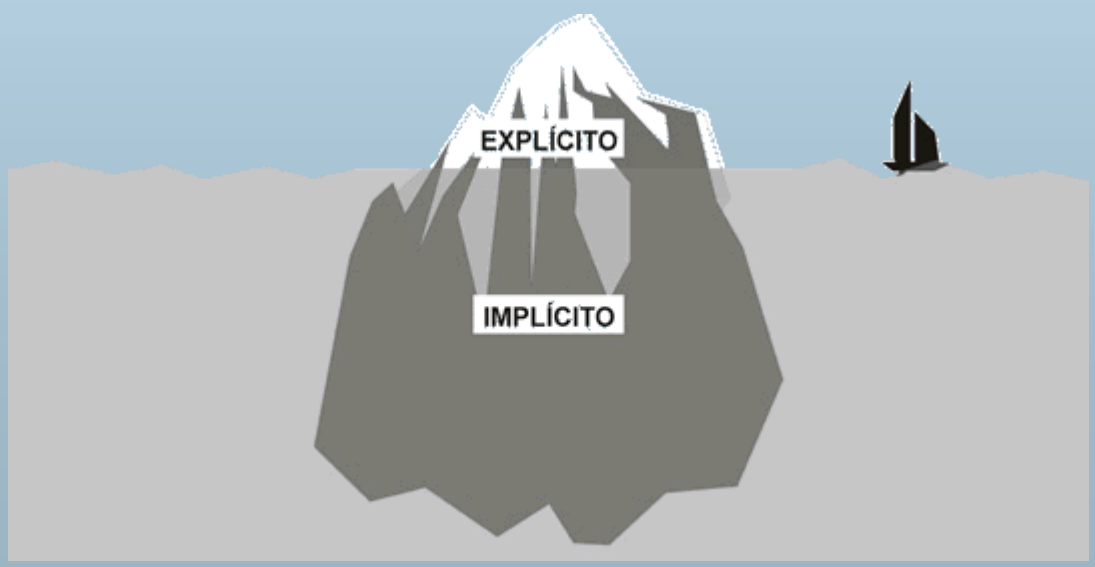
"Para quem tem um martelo, tudo parece prego"

Observações Importantes (cont.)

- A Engenharia de Software disponibiliza um conjunto de processos, métodos e ferramentas para produzir software de qualidade.
- Pode-se pensar como em um supermercado.
  - Em função do problema, se escolhe o processo, os métodos e as ferramentas.
- Entretanto.
  - Menos do que o necessário pode levar a desordem.
  - Mais do que o necessário pode emperrar o projeto.

Observações Importantes (cont.)

- Processos sempre existem, seja de forma implícita ou explícita.
  - Processos implícitos são difíceis de serem seguidos, em especial por desenvolvedores novatos e inexperientes.
  - Processos explícitos estabelecem as regras de forma clara.



Observações Importantes (cont.)

- Último indicador para medir a qualidade de um processo.
  - Satisfação do Cliente.
- Outros indicadores importantes.
  - Qualidade dos produtos gerados (adequação aos requisitos).
  - Custo real do projeto (orçamento).
  - Duração real do projeto (prazo).

Observações Importantes (cont.)

- Embora existam muitos processos de software diferentes, todos eles devem incluir, de alguma forma, quatro atividades fundamentais da Engenharia de Software.
  - Especificação - A funcionalidade do software e as restrições sobre sua operação devem ser definidas.
  - Desenvolvimento - O software deve ser produzido para atender à especificação.
  - Validação - O software deve ser validado para garantir que atenda ao que o cliente deseja.
  - Evolução - O software deve evoluir para atender às mudanças nas necessidades dos clientes.

Observações Importantes (cont.)

- É importante descrever também quem está envolvido, o que está sendo produzido e quais condições influenciam a sequência dessas atividades.
  - Uma atividade de processo resulta em produtos ou em entregas, que são artefatos de desenvolvimento.
    - Por exemplo, o resultado da atividade de projeto da arquitetura pode ser um modelo da arquitetura do software.
  - Os papéis refletem as responsabilidades das pessoas envolvidas no processo.
    - Entre os exemplos de papéis, tem-se os do gerente de projeto, do gerente de configuração e do programador.

Observações Importantes (cont.)

- Há condições que devem ser mantidas antes ou depois de uma atividade do processo ter sido aprovada ou um produto ter sido produzido.
  - Antes de começar o projeto da arquitetura, por exemplo, uma precondição poderia ser a de que o cliente tenha aprovado todos os requisitos.
  - Depois que essa atividade for concluída, uma pós-condição poderia ser a de que os modelos em UML que descrevem a arquitetura fossem revisados.

Modelos de Processos de Software

- Um modelo de processo de software, às vezes chamado de Ciclo de Vida do Desenvolvimento de Software (ou modelo SDLC, do inglês *Software Development Life Cycle)* constitui-se numa representação simplificada de um processo de software.
  - Cada modelo representa um processo a partir de uma perspectiva particular e desse modo, fornece apenas informações parciais sobre esse processo.

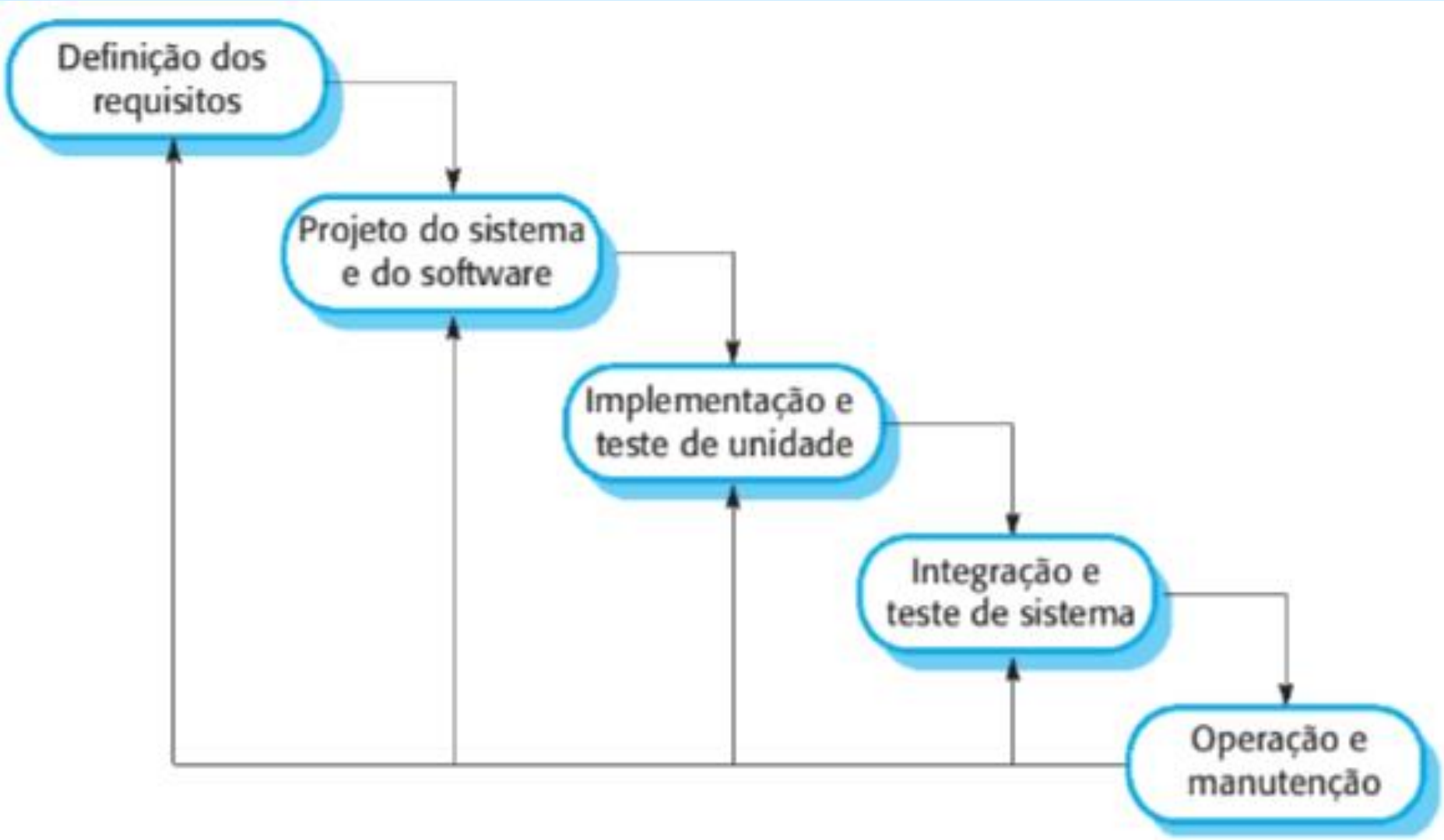
Modelos de Processos de Software

- Neste contexto, serão apresentados alguns modelos de ciclo de vida.
  - Cascata
  - Modelo em V
  - Incremental
  - Evolutivo
  - Reutilização de Software
  - RAD
  - Prototipação
  - Espiral
  - Ágeis

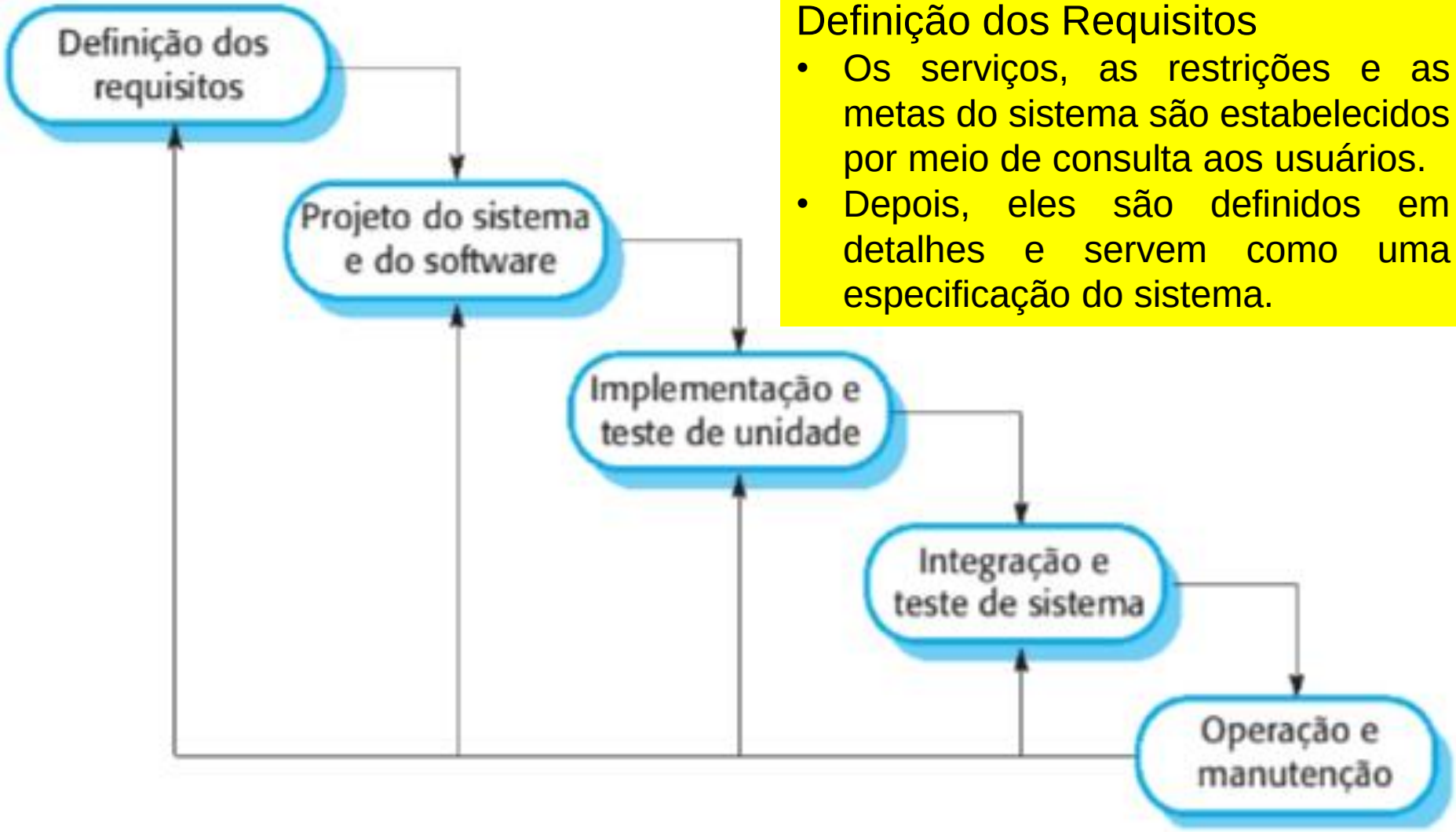
Modelo em Cascata (*Waterfall Model*)

- O primeiro modelo de processo de desenvolvimento de software a ser publicado foi derivado dos modelos utilizados na engenharia de grandes sistemas militares.
- Ele apresenta o processo de desenvolvimento de software como uma série de estágios ou etapas.
  - Devido à cascata de uma fase para outra, esse modelo é conhecido como Modelo em Cascata (*Waterfall Model*).
  - O modelo em cascata é um exemplo de processo dirigido por plano.
  - A princípio, pelo menos, é necessário planejar e criar um cronograma de todas as atividades de processo antes de começar o desenvolvimento do software.

Modelo em Cascata (*Waterfall Model*) (cont.)



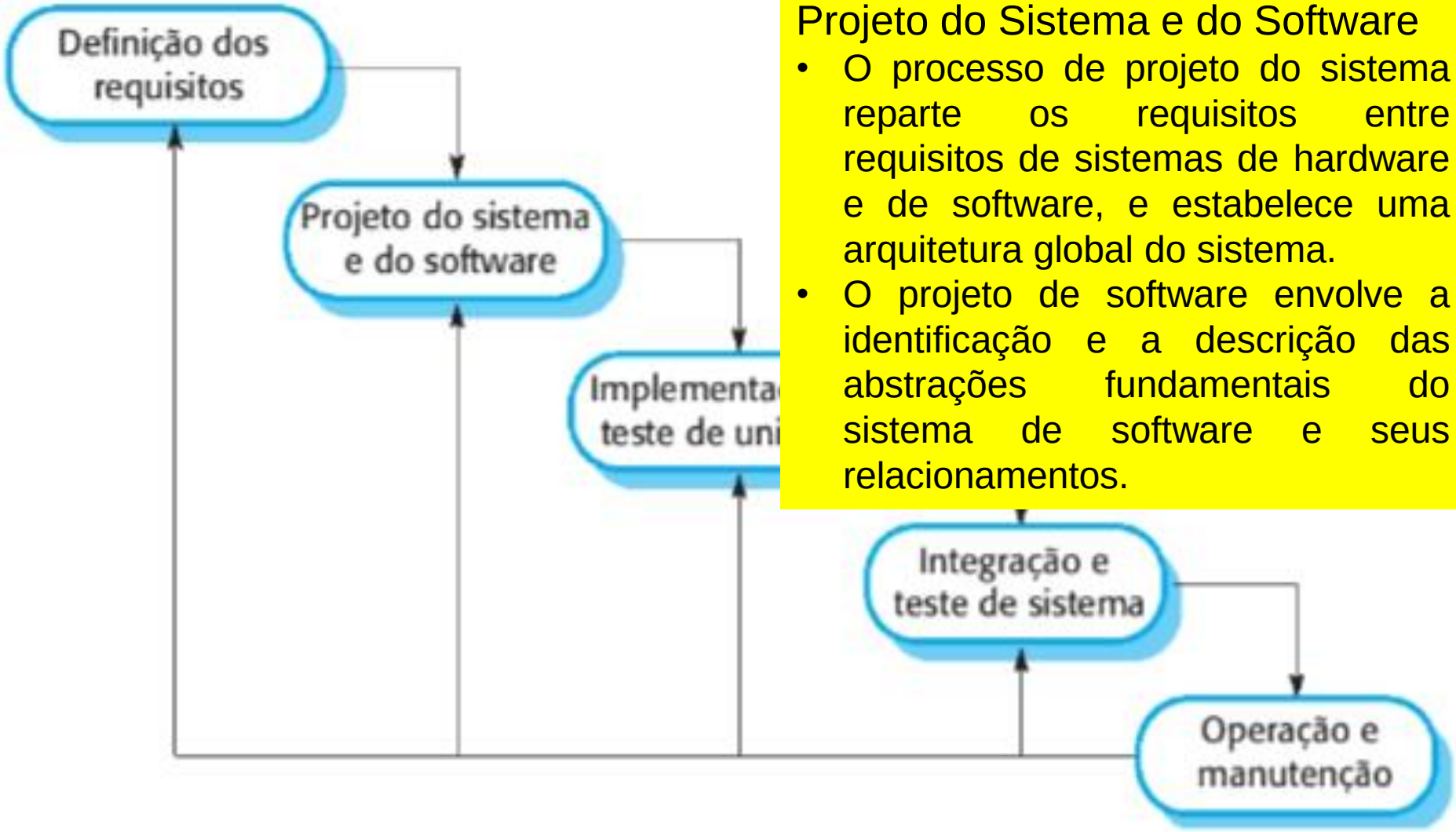
Modelo em Cascata (*Waterfall Model*) (cont.)



Definição dos Requisitos

- Os serviços, as restrições e as metas do sistema são estabelecidos por meio de consulta aos usuários.
- Depois, eles são definidos em detalhes e servem como uma especificação do sistema.

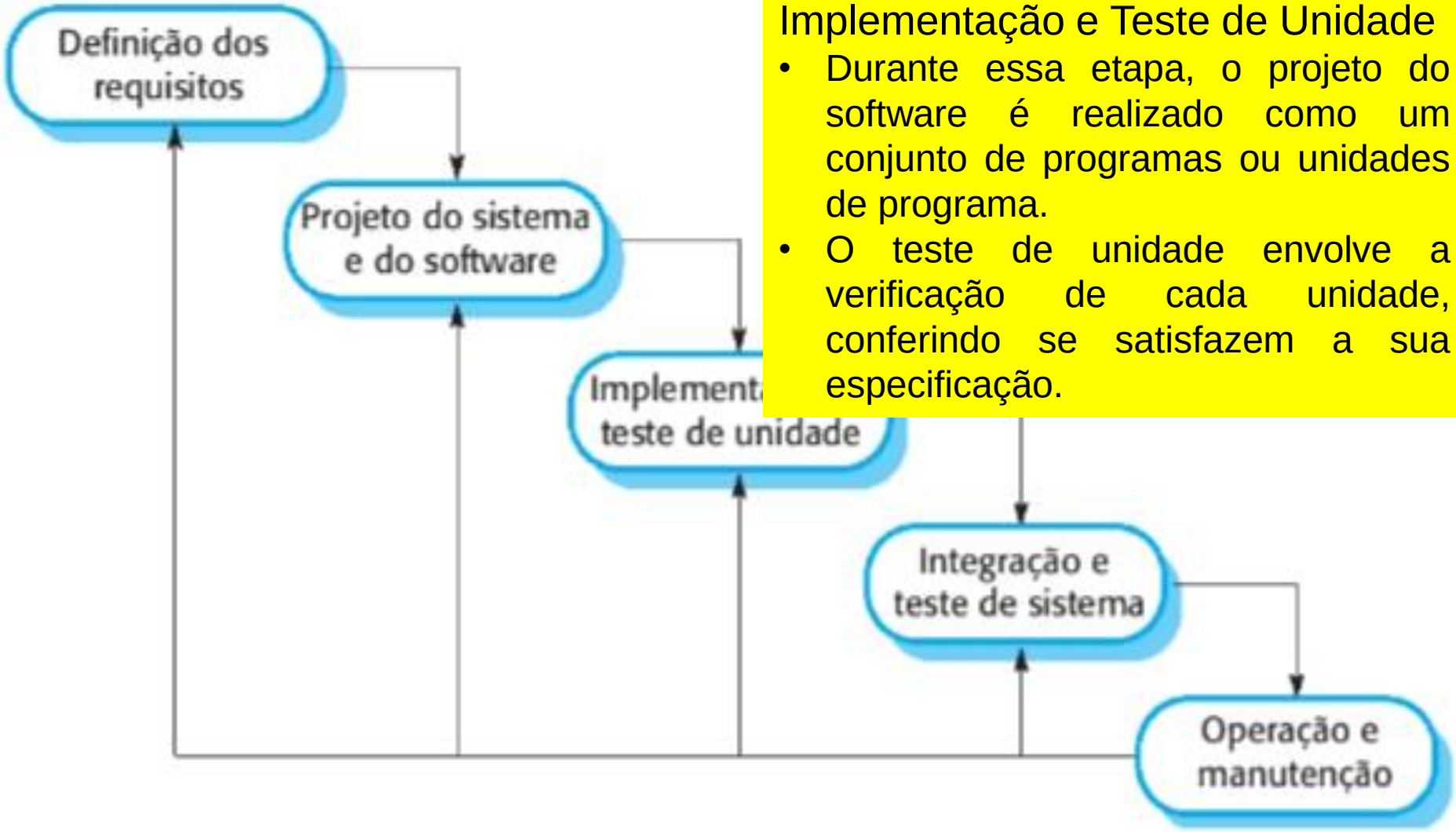
Modelo em Cascata (*Waterfall Model*) (cont.)



Projeto do Sistema e do Software

- O processo de projeto do sistema reparte os requisitos entre requisitos de sistemas de hardware e de software, e estabelece uma arquitetura global do sistema.
- O projeto de software envolve a identificação e a descrição das abstrações fundamentais do sistema de software e seus relacionamentos.

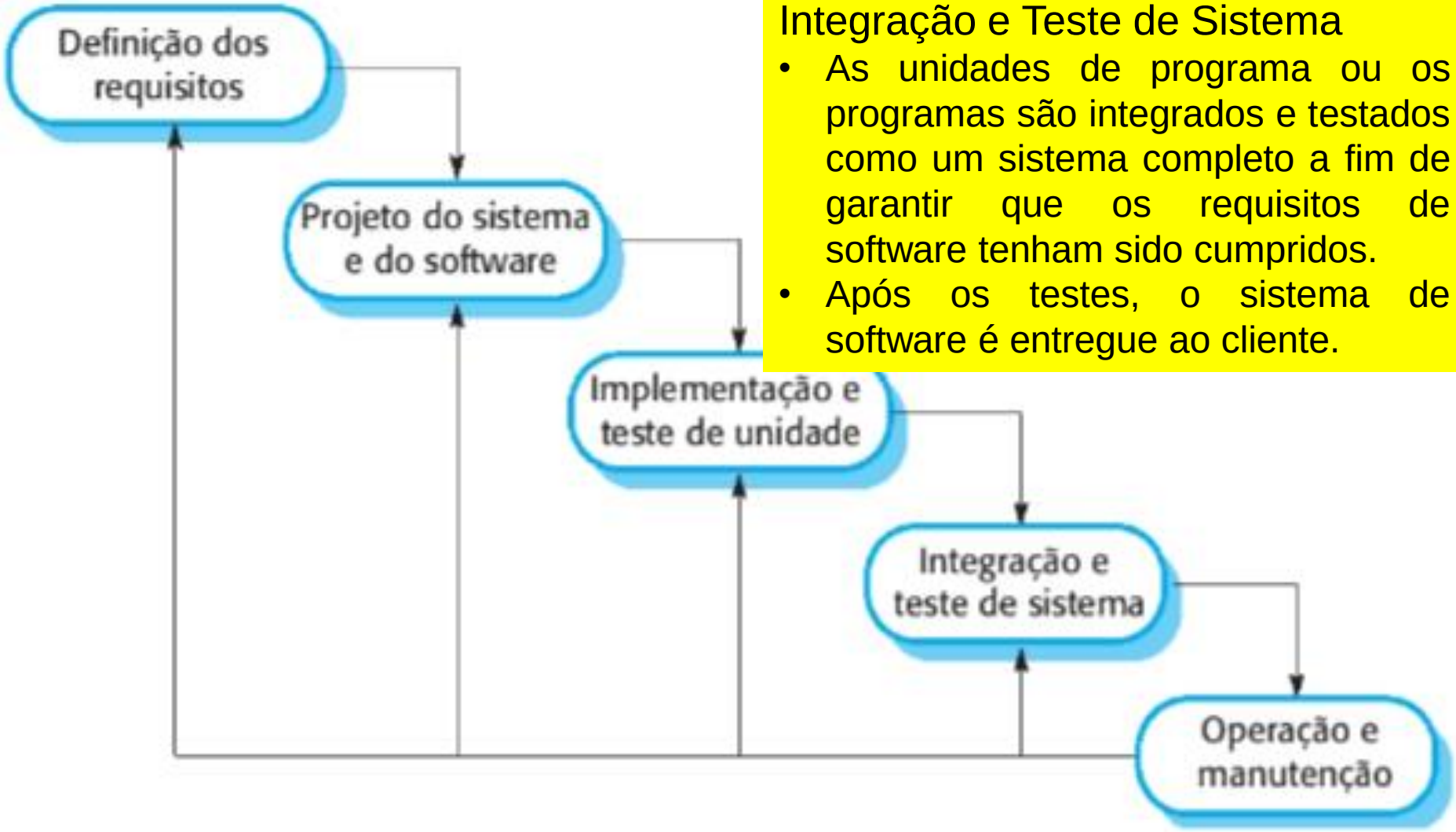
Modelo em Cascata (*Waterfall Model*) (cont.)



Implementação e Teste de Unidade

- Durante essa etapa, o projeto do software é realizado como um conjunto de programas ou unidades de programa.
- O teste de unidade envolve a verificação de cada unidade, conferindo se satisfazem a sua especificação.

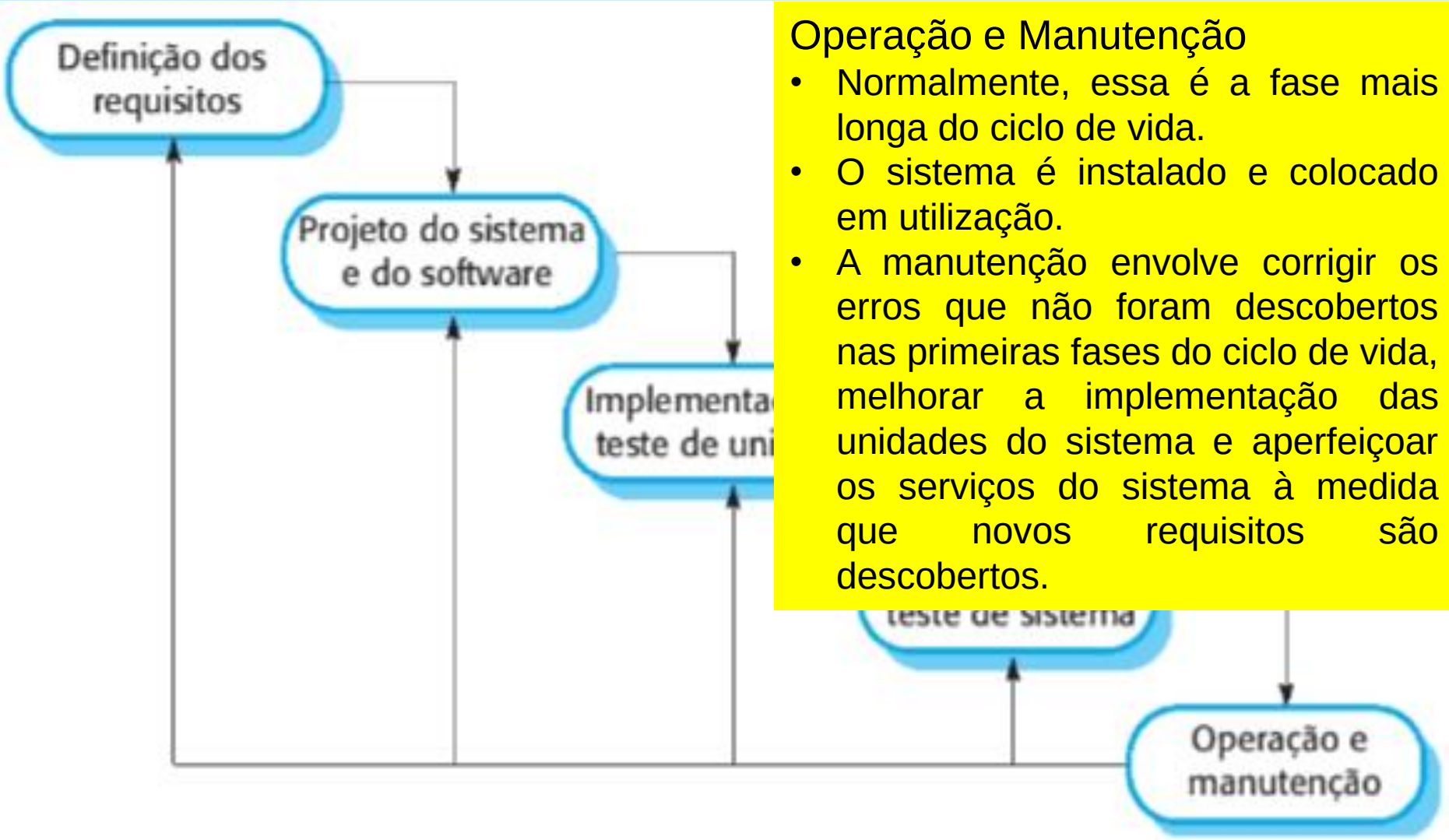
Modelo em Cascata (*Waterfall Model*) (cont.)



Integração e Teste de Sistema

- As unidades de programa ou os programas são integrados e testados como um sistema completo a fim de garantir que os requisitos de software tenham sido cumpridos.
- Após os testes, o sistema de software é entregue ao cliente.

Modelo em Cascata (*Waterfall Model*) (cont.)

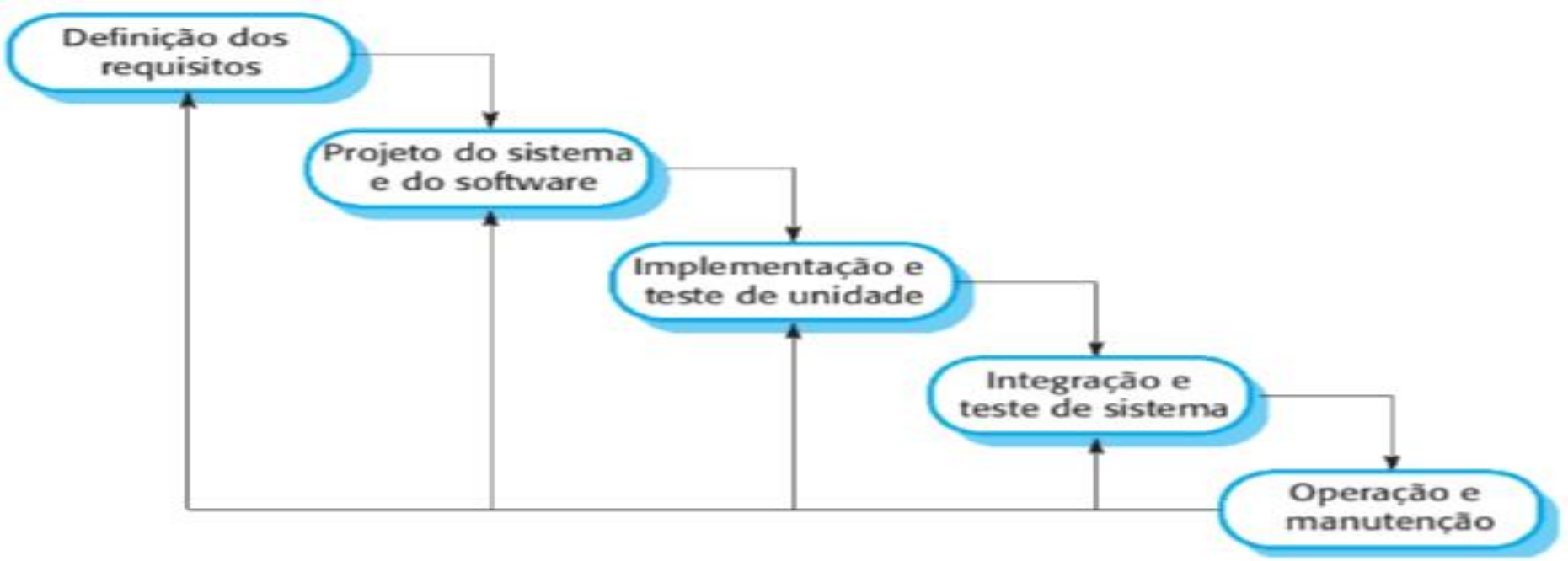


Operação e Manutenção

- Normalmente, essa é a fase mais longa do ciclo de vida.
- O sistema é instalado e colocado em utilização.
- A manutenção envolve corrigir os erros que não foram descobertos nas primeiras fases do ciclo de vida, melhorar a implementação das unidades do sistema e aperfeiçoar os serviços do sistema à medida que novos requisitos são descobertos.

Características do Modelo em Cascata

- Em princípio, todas as atividades de uma fase devem ser completadas para então se iniciar a próxima fase.
  - Assim, a principal desvantagem do Modelo em Cascata é a dificuldade de aceitar mudanças após o processo estar em execução.



Características do Modelo em Cascata (cont.)

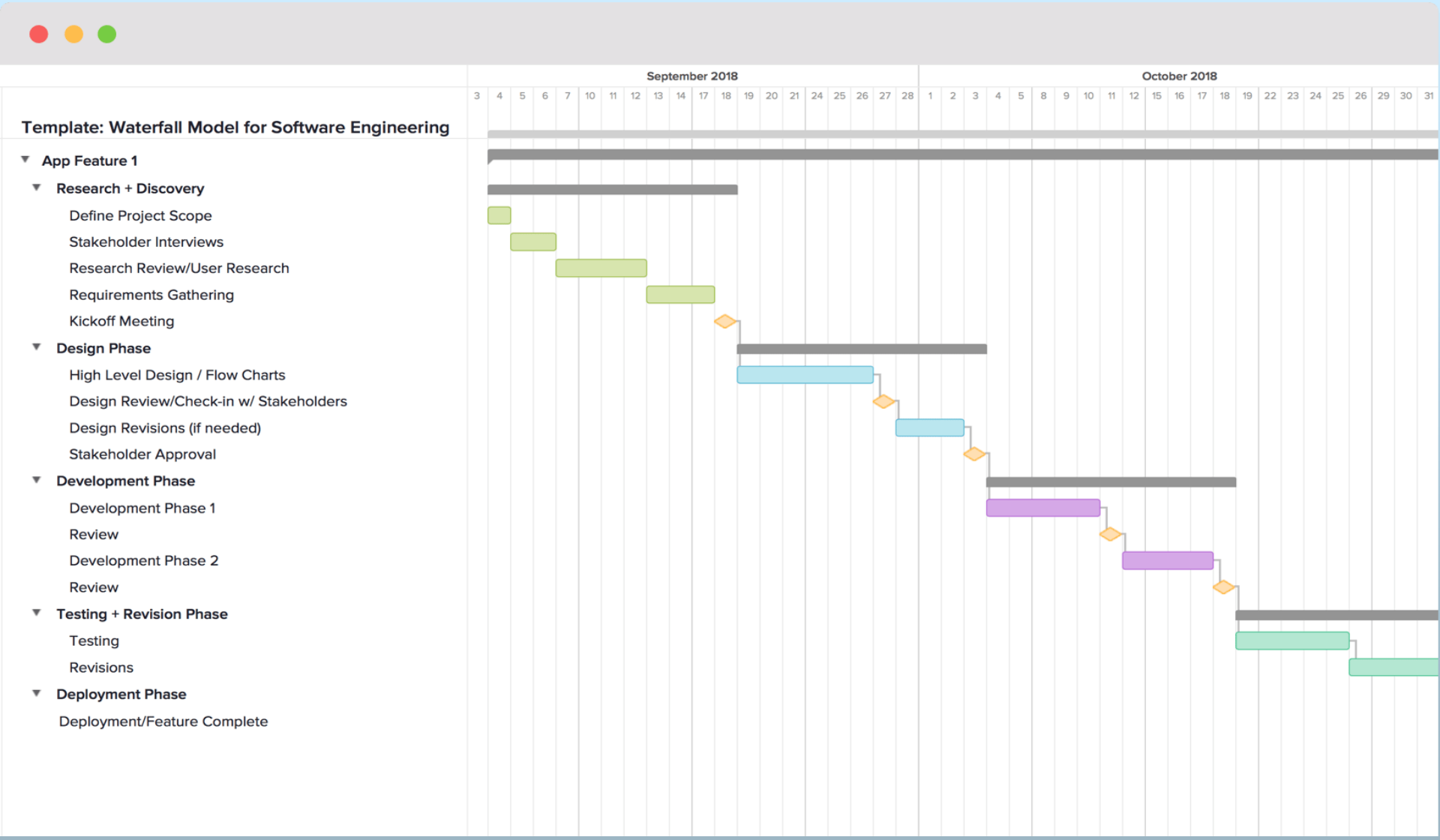


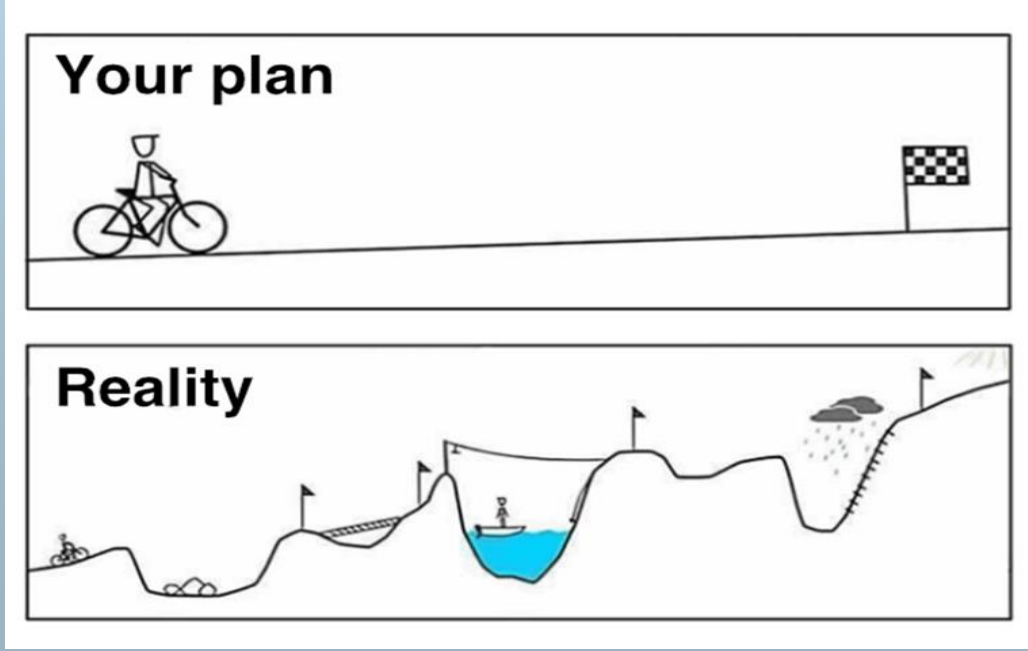
Gráfico de Gantt.

Características do Modelo em Cascata (cont.)

Advantages	Dis-Advantages
Easy to understand, easy to use. Good for management control (plan, staff, track).	
Before the next phase of development, each phase must be completed	Error can be fixed only during the phase
Suited for smaller projects where requirements are well defined	It is not desirable for complex project where requirement changes frequently
Team should perform quality assurance test (Verification and Validation) before completing each stage	Testing period comes quite late in the developmental process
Detailed documentation is done at every phase	Documentation occupies a lot of time of developers and testers
Project is completely dependent on project team with minimum customer involvement	Customers' feedbacks cannot be included during ongoing development phase
Any changes needed are made in development	Small changes or errors that arise in the completed software may cause a lot of problems
Milestones are well understood (in plan)	It is difficult to measure progress in stages. Can give a false impression about progress
Provides structure to inexperienced staff	

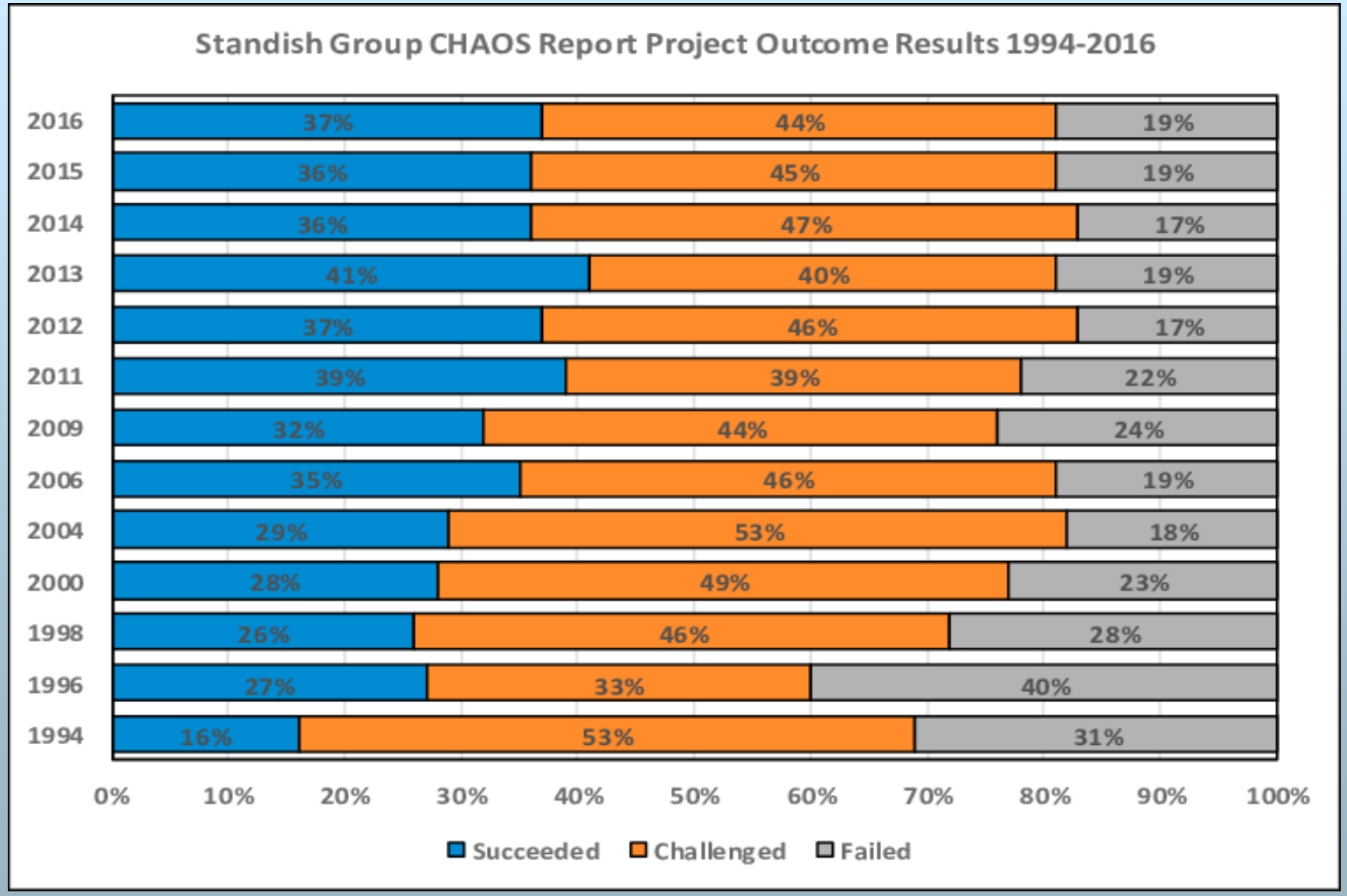
Características do Modelo em Cascata (cont.)

- A integração é feita como um "*big-bang*".
- Há pouca oportunidade para os clientes verem seu sistema antes de concluí-lo (pode ser tarde demais para perceber erros).
- Todos os requisitos devem ser conhecidos claramente no início.

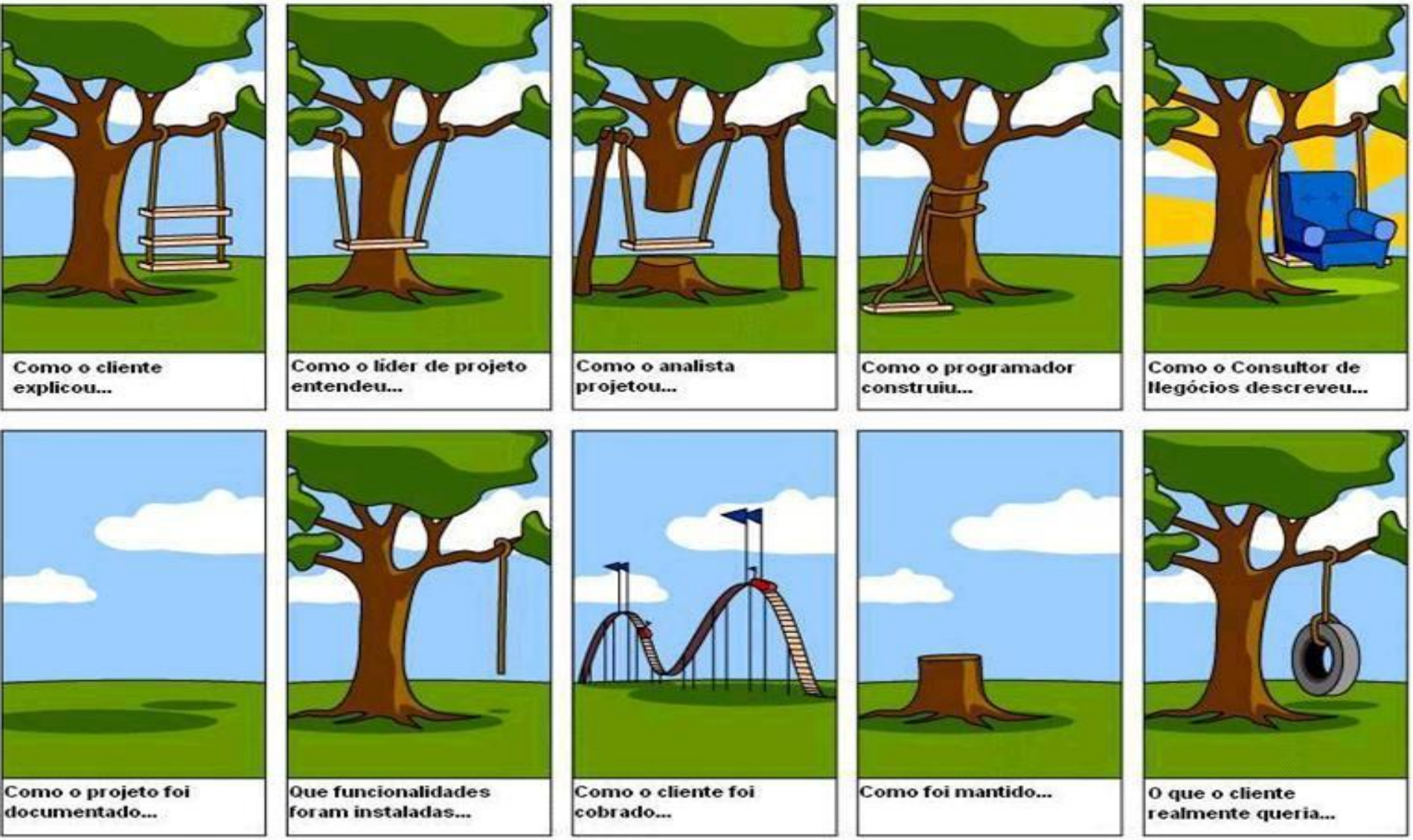


Características do Modelo em Cascata (cont.)

Características do Modelo em Cascata (cont.)



Características do Modelo em Cascata (cont.)



Quando Utilizar o Modelo em Cascata

- Se os requisitos forem claros e fixos
  - Isso pode não mudar...
  - Não mudar com muita frequência…
- Se não houver requisitos pouco claros (sem confusão ou ambiguidade).
- Se a aplicação não for complexa e não for extensa.
- Se a tecnologia preferida/necessária for bem compreendida.
  - A tecnologia usada não é dinâmica e é estável.
- Se o projeto for curto e se houver uma equipe pequena.
- Se o risco for zero ou mínimo.

FBI e o Caso Sentinel

Entendendo o caso:

Em 2003, o FBI decidiu digitalizar seus arquivos de casos, substituir buscas físicas por buscas integradas e online em uma plataforma, beneficiaria os usuários e o trabalho do FBI em achar com mais rapidez e eficiência conexões entre os casos já trabalhados.

Em 2006, após a contratação de uma empresa de segurança para dar início ao projeto que tinha uma projeção de base de 30 mil usuários finais e o orçamento de US$ 451 milhões, optou-se pela metodologia cascata. Foram estimadas 4 fases para o projeto e foi acertado o prazo de dezembro de 2009 para entrega do software.

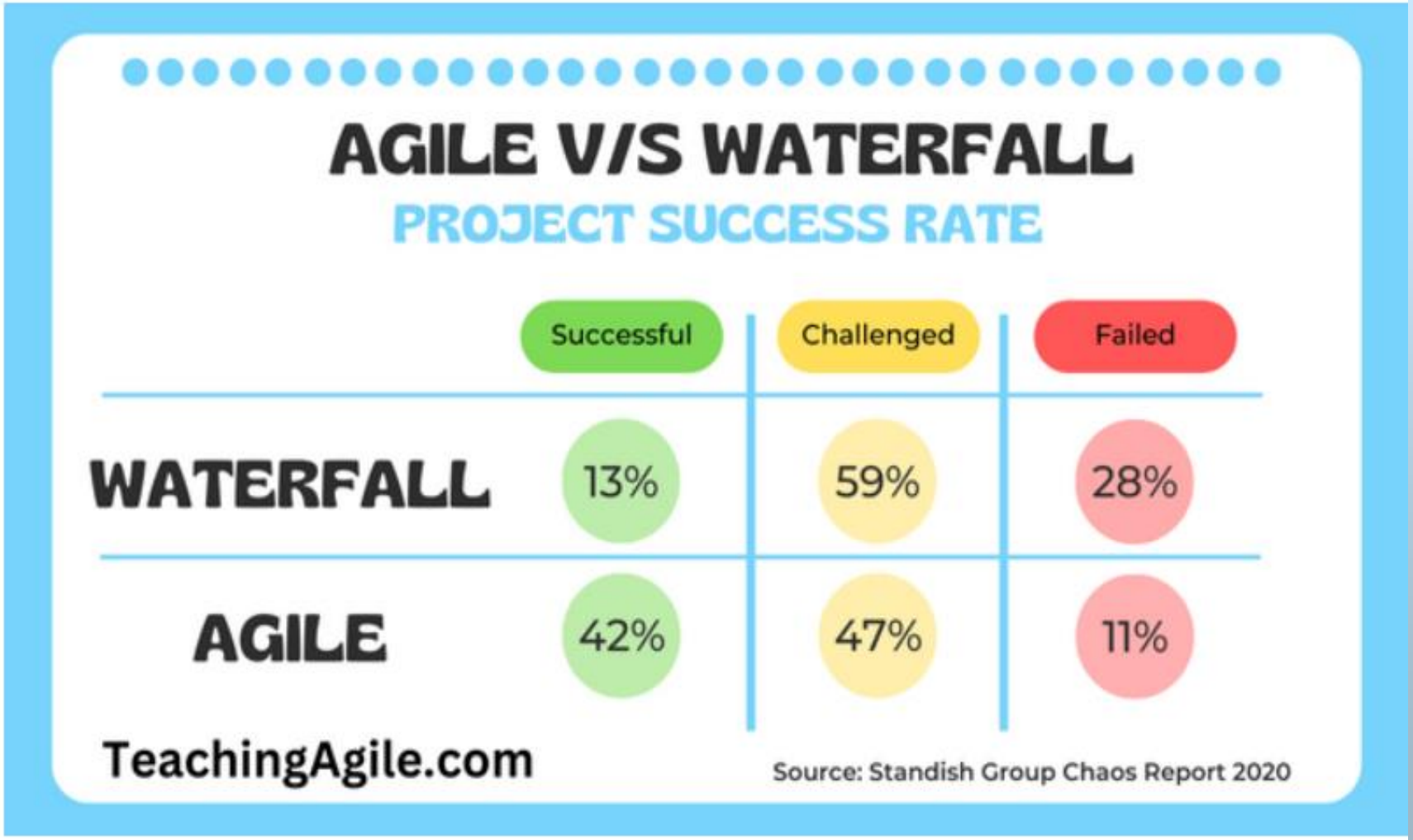
Em agosto de 2010, o FBI já havia gastado US$ 405 milhões do orçamento anteriormente previsto, no entanto recebeu apenas duas fases das quatro estimadas a princípio e decidiu cancelar o projeto. Um auditor, contratado pelo FBI concluiu que para termina o projeto da forma como estava sendo feito, precisaria de mais 6 (seis) anos e US$ 35 milhões, além do orçamento.

FBI e o Caso Sentinel (cont.)

Ainda em 2010, com um novo CTO decidiu retomar o projeto utilizando o framework Scrum. De acordo com a projeção do CTO, seria possível, com o saldo remanescente entregar o projeto em um período de 12 meses. Houve uma redução da equipe de 400 para 45 pessoas, das quais 15 eram desenvolvedores. As Sprints para a entrega de novas funcionalidades eram de um mês e as releases e implementação de recursos construídos eram feitas a cada 3 sprints em um piloto de campo.

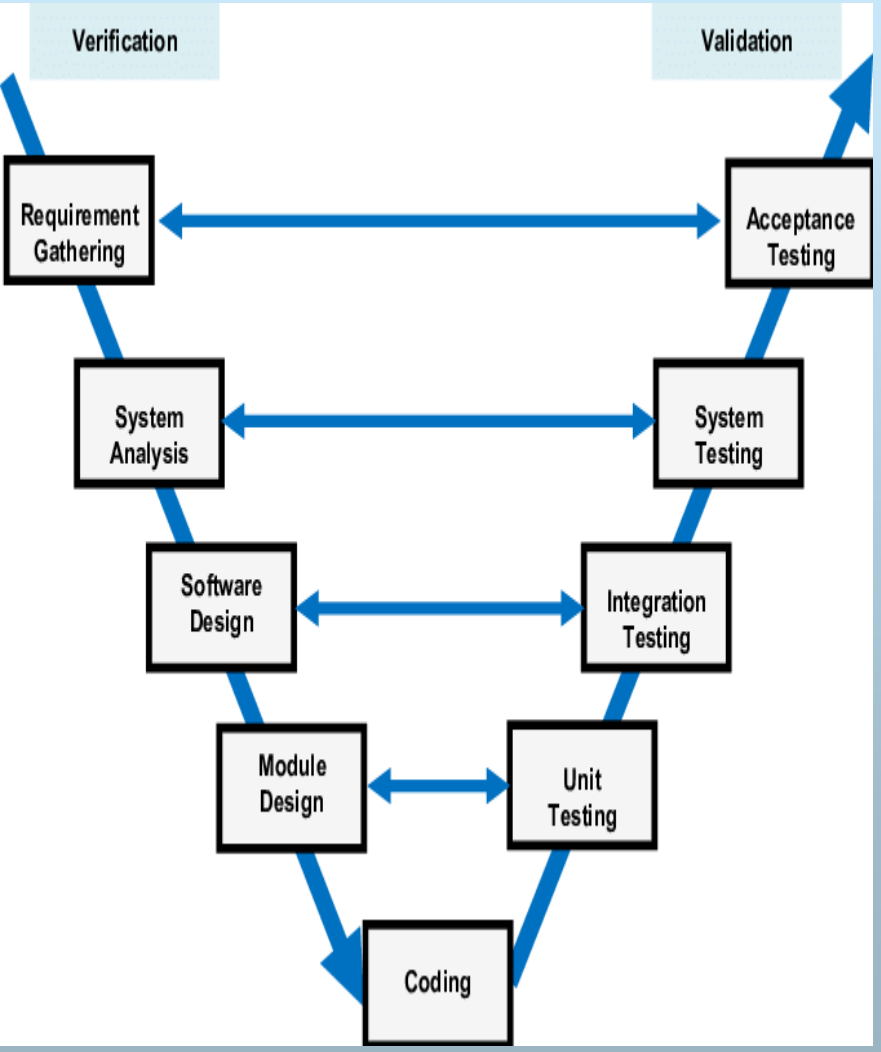
Em novembro de 2011, um ano após a retomada do projeto Sentinel, todas as fases foram concluídas.

FBI e o Caso Sentinel (cont.)



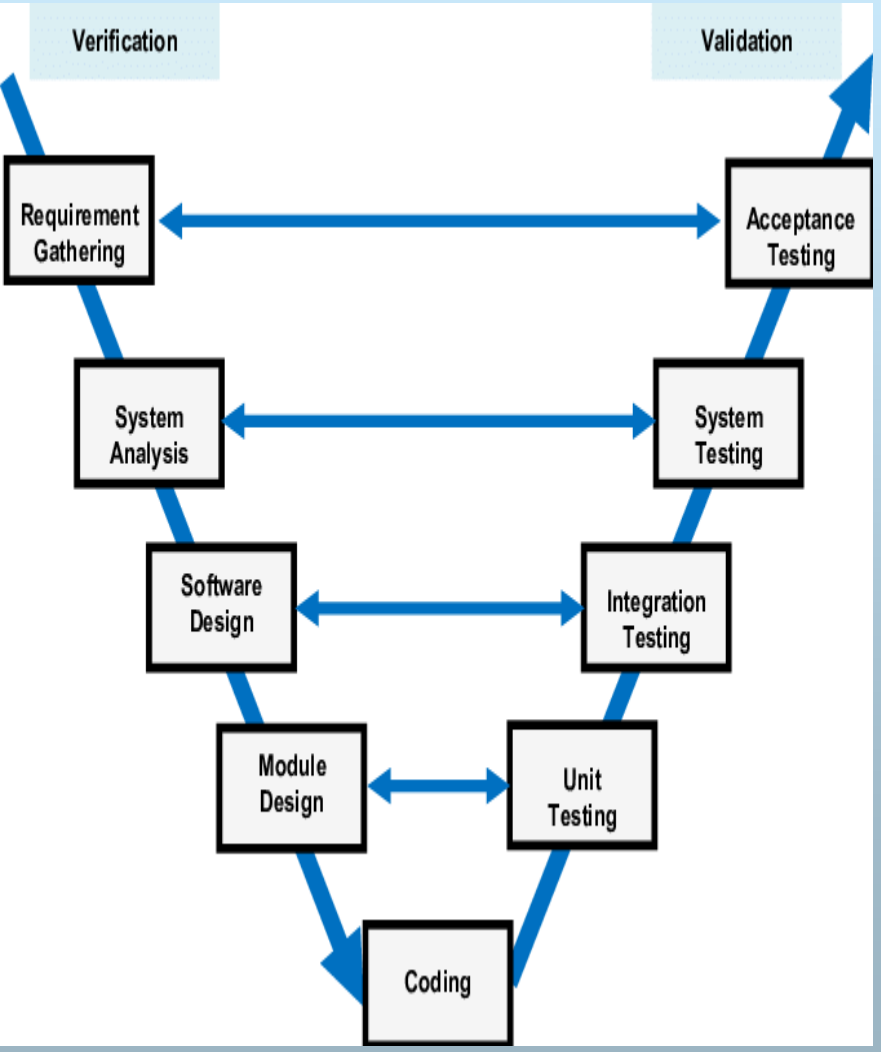
Agile v/s Waterfall based on 2020 Standish Group Chaos Study

Modelo de Desenvolvimento em V



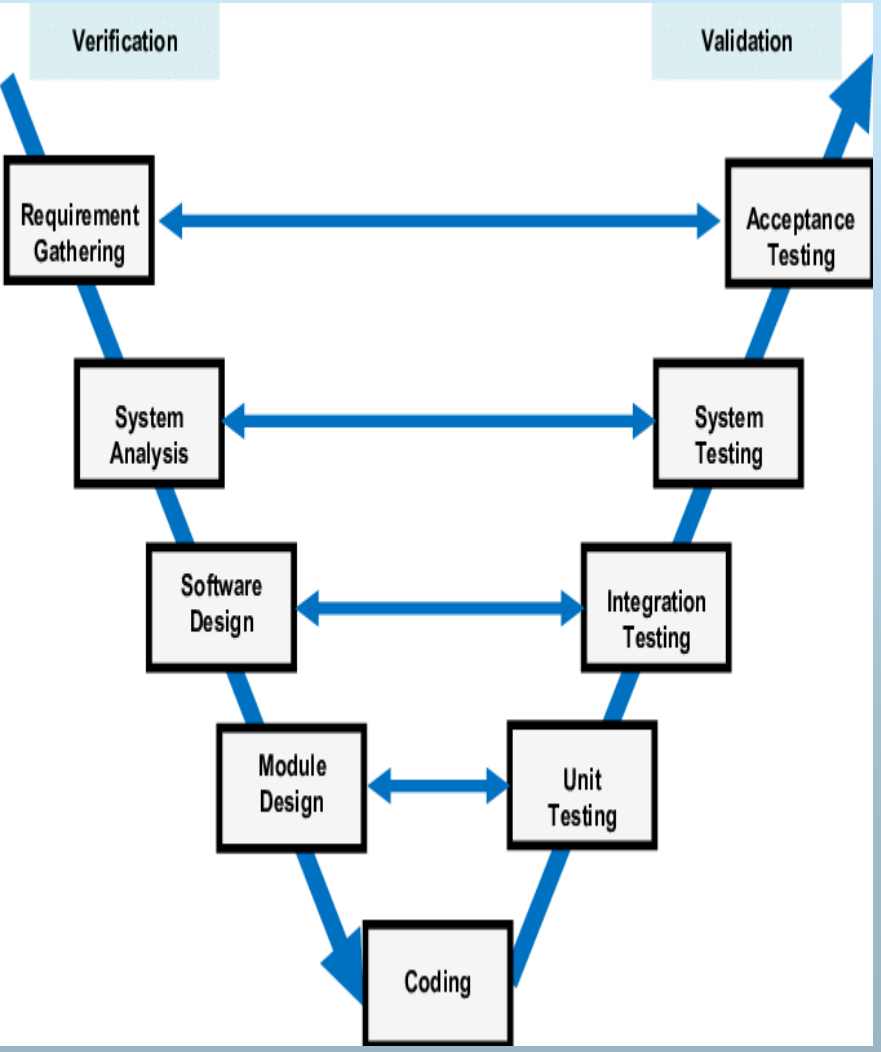
- Ao contrário do Modelo Cascata, o Modelo de Desenvolvimento de Software em V (Modelo de Desenvolvimento em V ou Modelo V) integra o processo de teste ao longo do processo de desenvolvimento, implementando o princípio de testar do início.
  - O Modelo V inclui níveis de teste associados a cada fase de desenvolvimento correspondente, ainda suportando o princípio de testar do início, e a execução dos testes associados a cada nível de teste ocorre sequencialmente, mas em alguns casos pode ocorrer a sobreposição.

Modelo de Desenvolvimento em V (cont.)



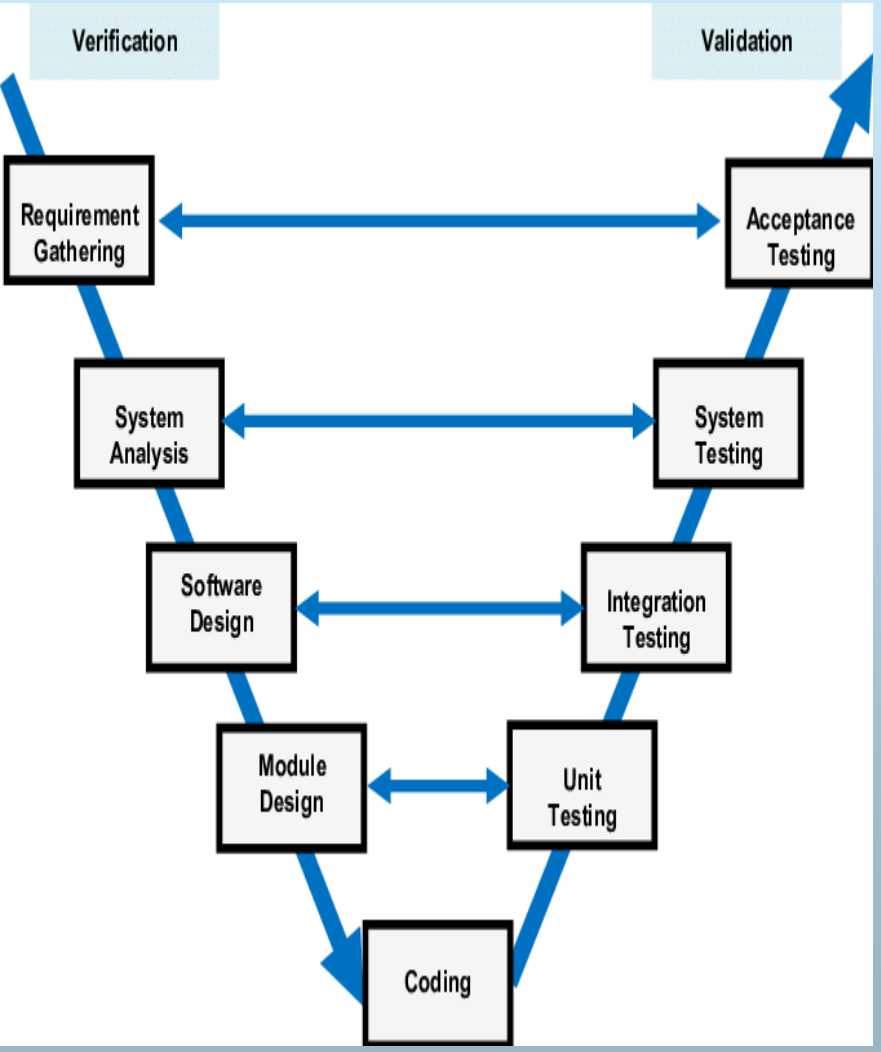
- Features.
  - It is plan-driven.
  - A modified of the Waterfall model.
  - Highly disciplined SDLC model.
  - Highlights the verification and validation of the product.
  - Testing of the product is planned in parallel with a corresponding phase of development.

Modelo de Desenvolvimento em V (cont.)



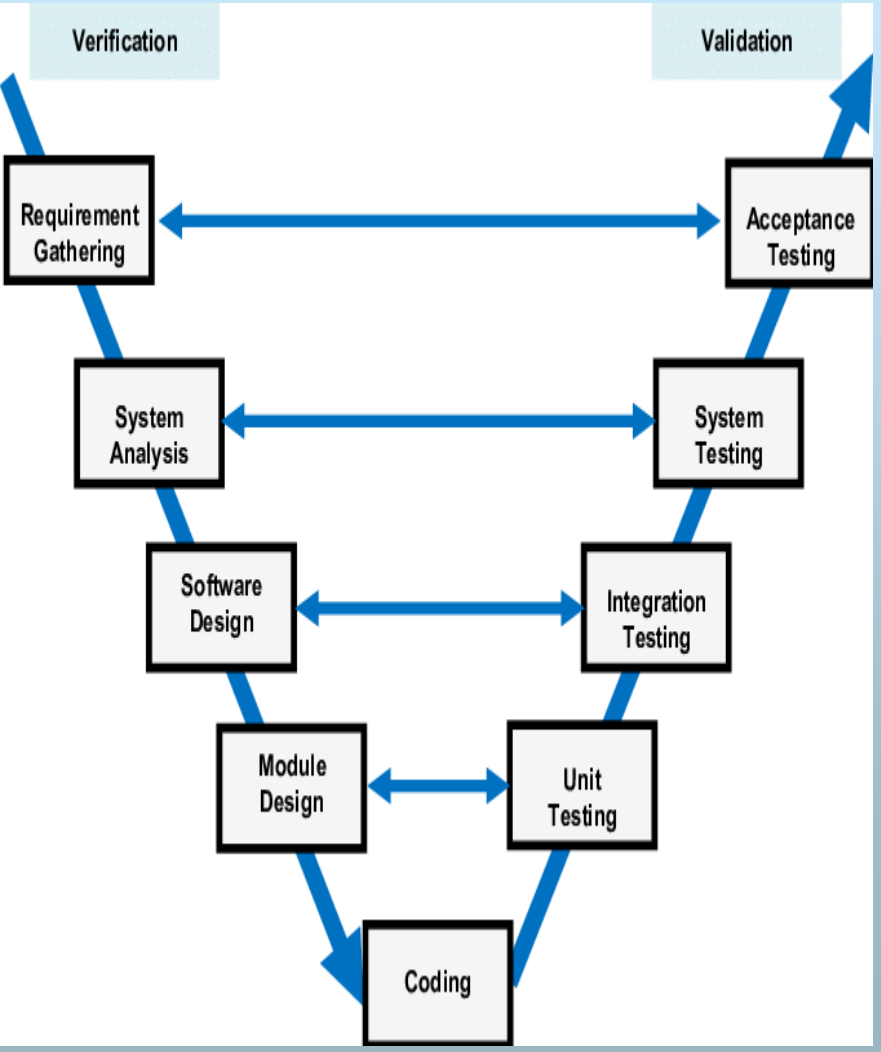
- Strengths.
  - Test planning for verification and validation of the product is done in early stages of product development (for quality control steps).
  - Why testing is important?
    - Each deliverable must be testable to increase SW quality.
  - Project manager can monitor progress by milestones.
  - Easy to use.
  - If high reliability products are needed and expected, this model is good.
    - Because many tests are performed.

Modelo de Desenvolvimento em V (cont.)



- Weakness.
  - Does not easily handle concurrent events/tasks, because all tasks are created plan-driven for this model.
  - Does not good for iterations or phases.
  - Does not easily apply when requirement changes needed (difficult to change plans in any time!!!).
  - Does not contain risk analysis activities.

Modelo de Desenvolvimento em V (cont.)



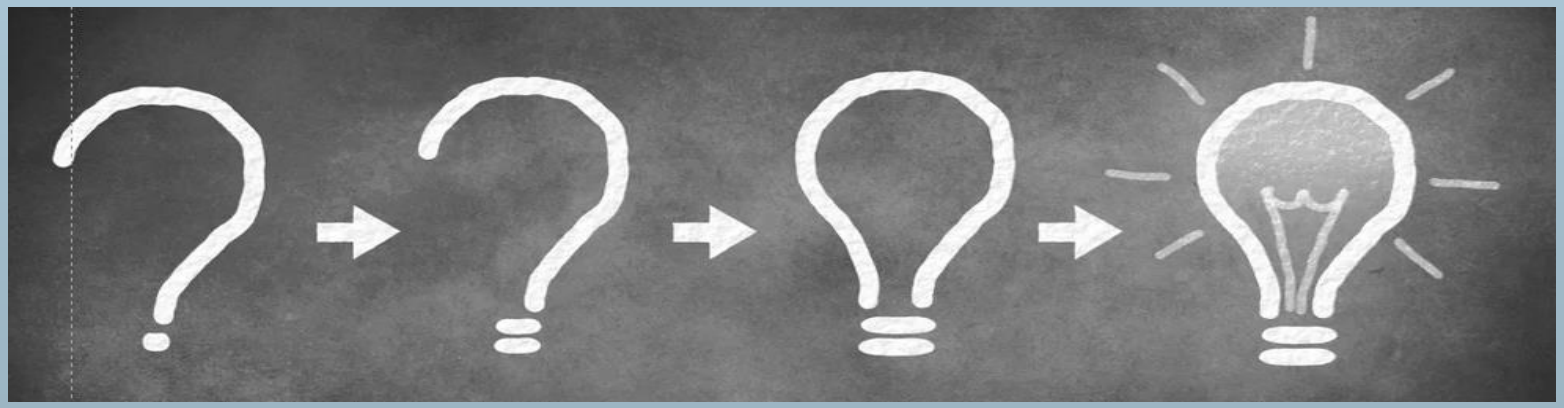
- When to use.
  - Excellent choice for systems requiring high reliability – i.e. hospital patient control applications (mission-critical computing systems).
  - If all requirements are known clearly, it is good.
  - If solution and technology are known well, it is good.

Desenvolvimento Incremental

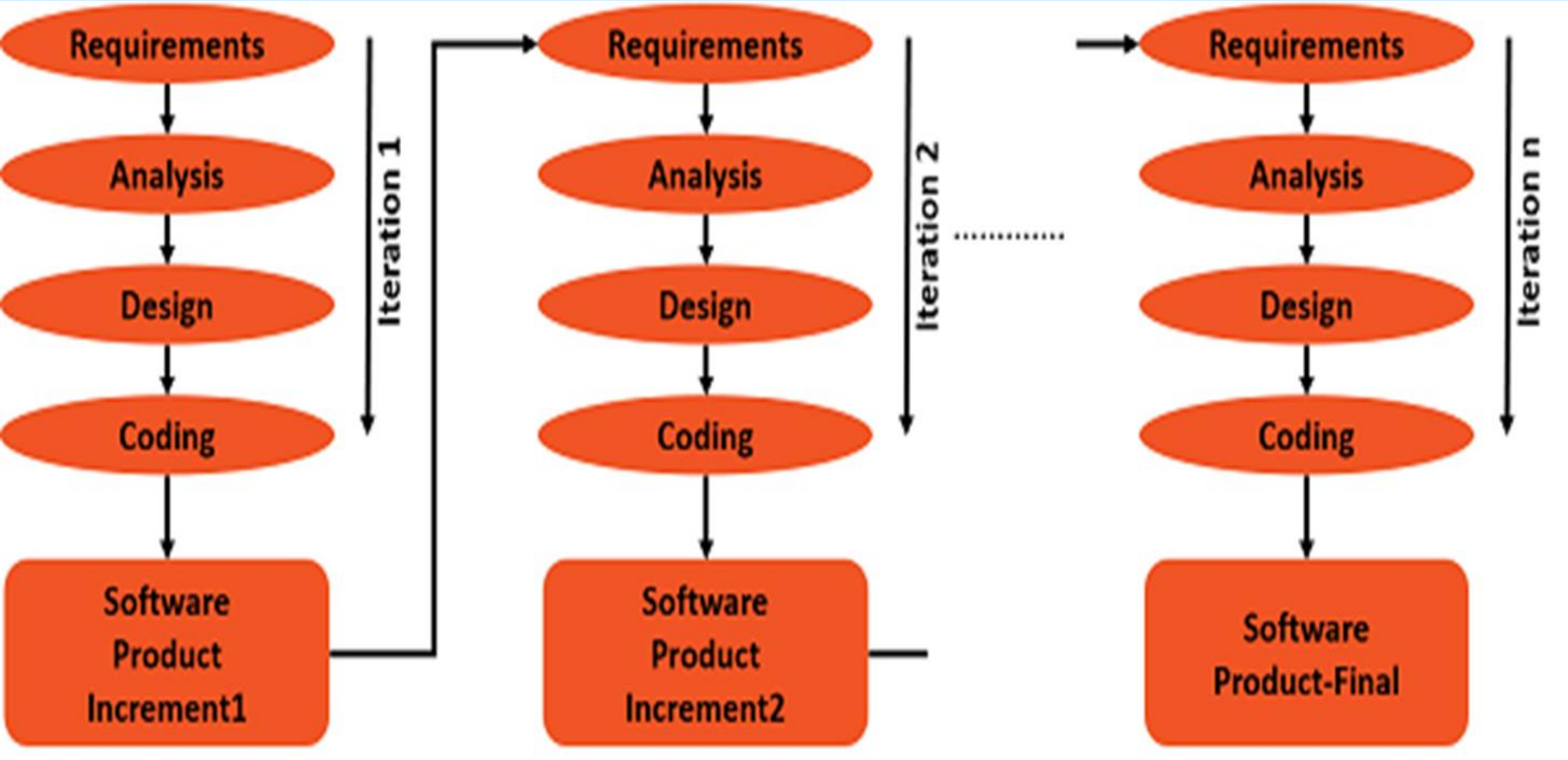
- O Desenvolvimento Incremental de Software, é mais apropriado do que uma abordagem em cascata para sistemas cujos requisitos estão propensos a mudar durante o processo de desenvolvimento.
  - É o caso da maioria dos sistemas de negócio e dos produtos de software.

Desenvolvimento Incremental (cont.)

- O Desenvolvimento Incremental reflete a maneira como se soluciona problemas.
  - Raramente se elabora uma solução completa para os problemas com antecedência, ou seja, em vez disso, caminha-se para uma solução em uma série de passos, retrocedendo quando se percebe que foi cometido um erro.
  - Ao se desenvolver um software de modo incremental, é mais barato e fácil fazer alterações nele durante o processo de desenvolvimento.



Desenvolvimento Incremental (cont.)



O Desenvolvimento Incremental se baseia na ideia de desenvolver uma implementação inicial, obter *feedback* dos usuários ou terceiros e fazer o software evoluir através de várias versões, até alcançar o sistema necessário.

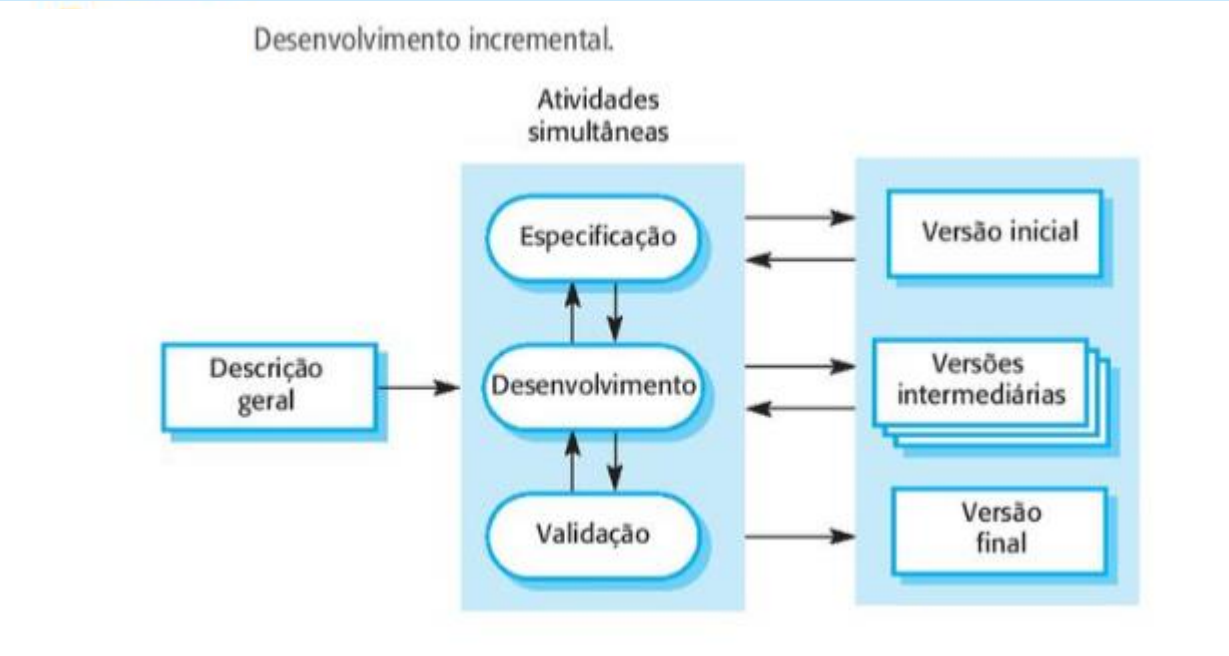
Desenvolvimento Incremental (cont.)

- Características.
  - A alteração dos requisitos do cliente é possível.
  - O custo das alterações de requisitos é reduzido.
  - A quantidade de análise e documentação para este modelo é muito menor do que o Modelo em Cascata.
  - É mais fácil obter *feedback* dos clientes durante o trabalho de desenvolvimento.
  - Os clientes podem comentar as demonstrações do software e ver o quanto foi implementado (isso aumenta a confiança do cliente no desenvolvedor).

Desenvolvimento Incremental (cont.)

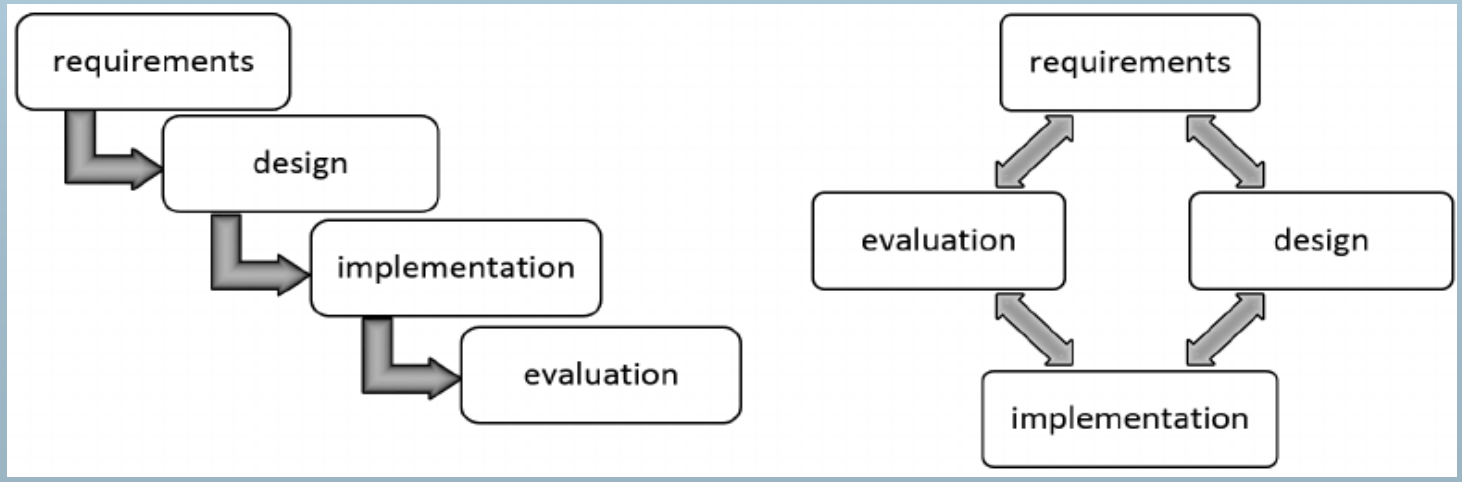
- Características.
  - É possível uma entrega e implantação mais rápidas de software útil para o cliente.
  - Os clientes podem usar o software e obter valor dele mais cedo (no cascata, a entrega de software aos clientes é mais tardia).

Desenvolvimento Incremental (cont.)



Incremental X Cascata

- O Desenvolvimento Incremental tem algumas grandes vantagens em relação ao Modelo em Cascata.
  - O custo de implementação das mudanças nos requisitos é reduzido.
  - A quantidade de análise e documentação que precisa ser refeita é significativamente menor do que a necessária ao modelo em cascata.

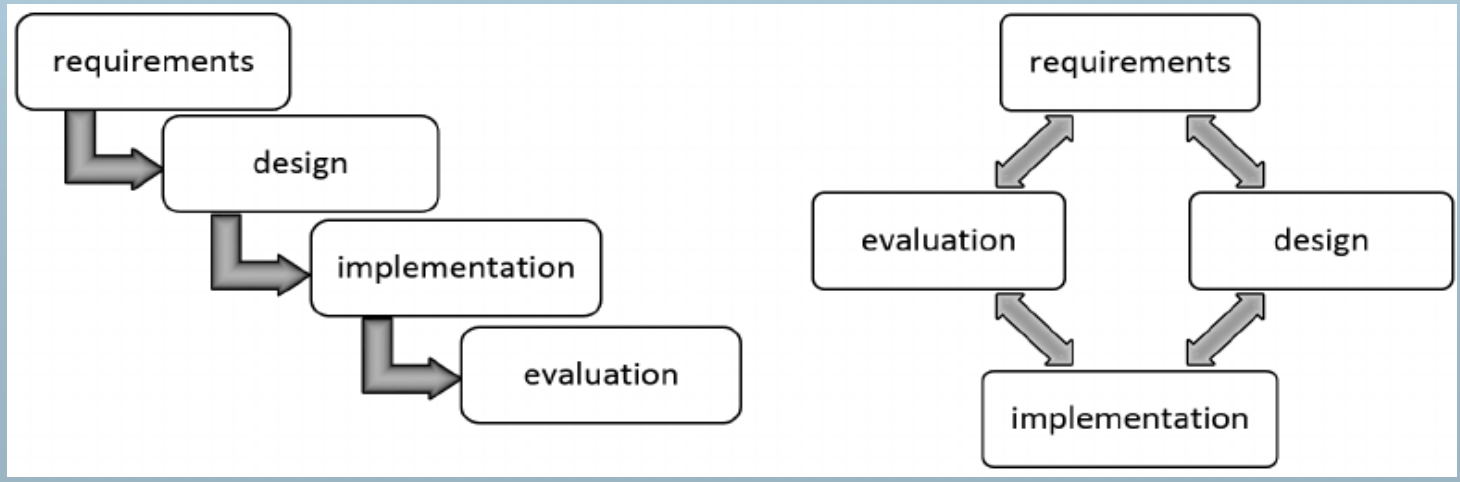


Incremental X Cascata (cont.)

- O Desenvolvimento Incremental tem algumas grandes vantagens em relação ao modelo em cascata.
  - É mais fácil obter *feedback* do cliente sobre o trabalho de desenvolvimento.
  - Os clientes podem comentar as demonstrações de software e ver o quanto foi implementado.
  - Para eles, é mais difícil julgar o progresso a partir dos documentos do projeto (*design*) de software.

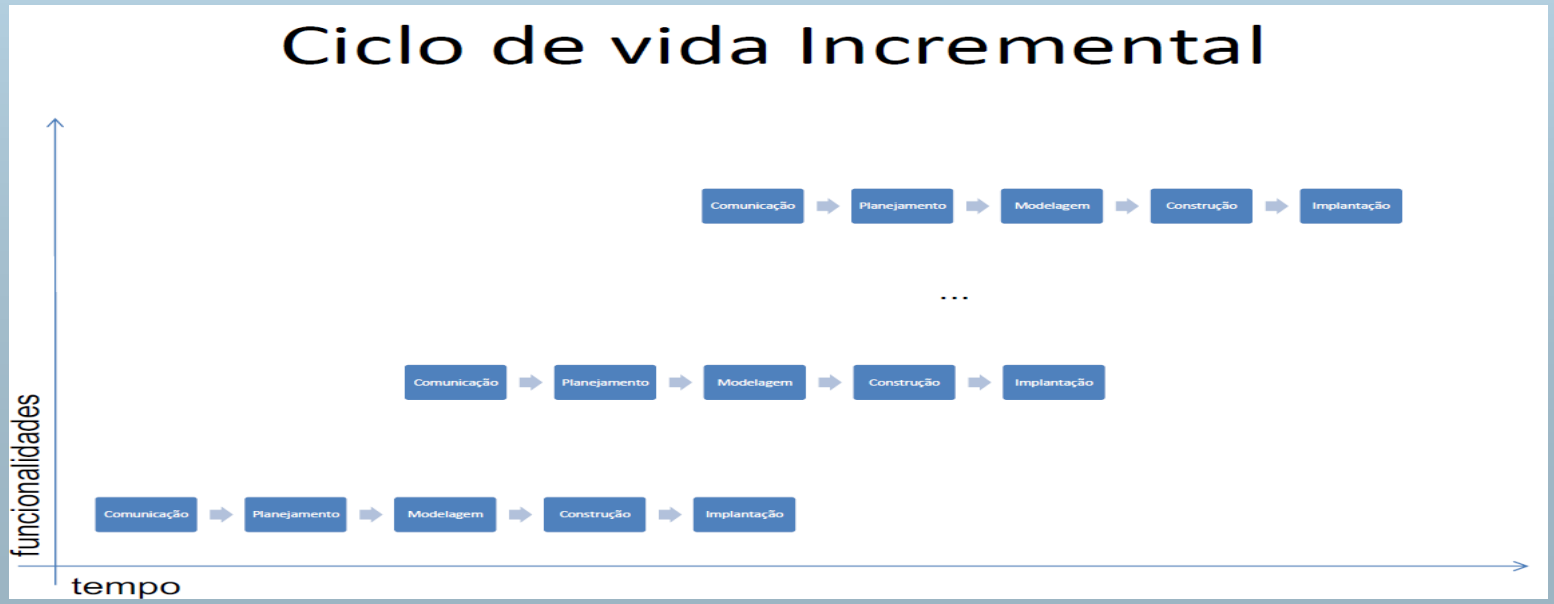
Incremental X Cascata (cont.)

- O Desenvolvimento Incremental tem algumas grandes vantagens em relação ao modelo em cascata:
  - A entrega e a implantação antecipadas de um software útil para o cliente são possíveis, mesmo se toda a funcionalidade não tiver sido incluída.
  - Os clientes são capazes de usar o software e de obter valor a partir dele mais cedo do que com um processo em cascata.



Problemas no Incremental

- O processo não é visível.
  - Os gerentes precisam de resultados regulares para medir o progresso.
  - Se os sistemas forem desenvolvidos rapidamente, não é econômico produzir documentos que reflitam cada versão do sistema.

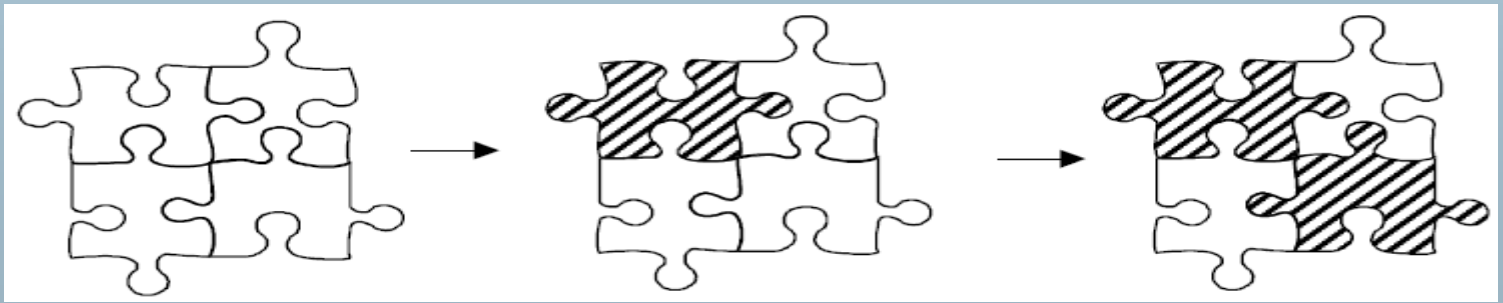


Problemas no Incremental (cont.)

- A estrutura do sistema tende a se degradar à medida que novos incrementos são adicionados.
  - Mudanças regulares deixam o código bagunçado, uma vez que novas funcionalidades são adicionadas de qualquer maneira possível.
  - Fica cada vez mais difícil e caro adicionar novas características a um sistema.
  - Para reduzir a degradação estrutural e a bagunça generalizada no código, os métodos ágeis sugerem que se refatore (melhore e reestruture, ou seja, que se faça *refactoring*) o software regularmente.

Problemas no Incremental (cont.)

- Os problemas do Desenvolvimento Incremental se tomam particularmente críticos nos sistemas extensos, complexos e de vida longa, nos quais diferentes times desenvolvem partes distintas do sistema.
  - Os sistemas extensos precisam de uma arquitetura estável, e as responsabilidades dos diferentes times trabalhando no sistema precisam ser claramente definidas em relação a essa arquitetura.
  - Isso deve ser planejado antecipadamente em vez de desenvolvido de modo incremental.

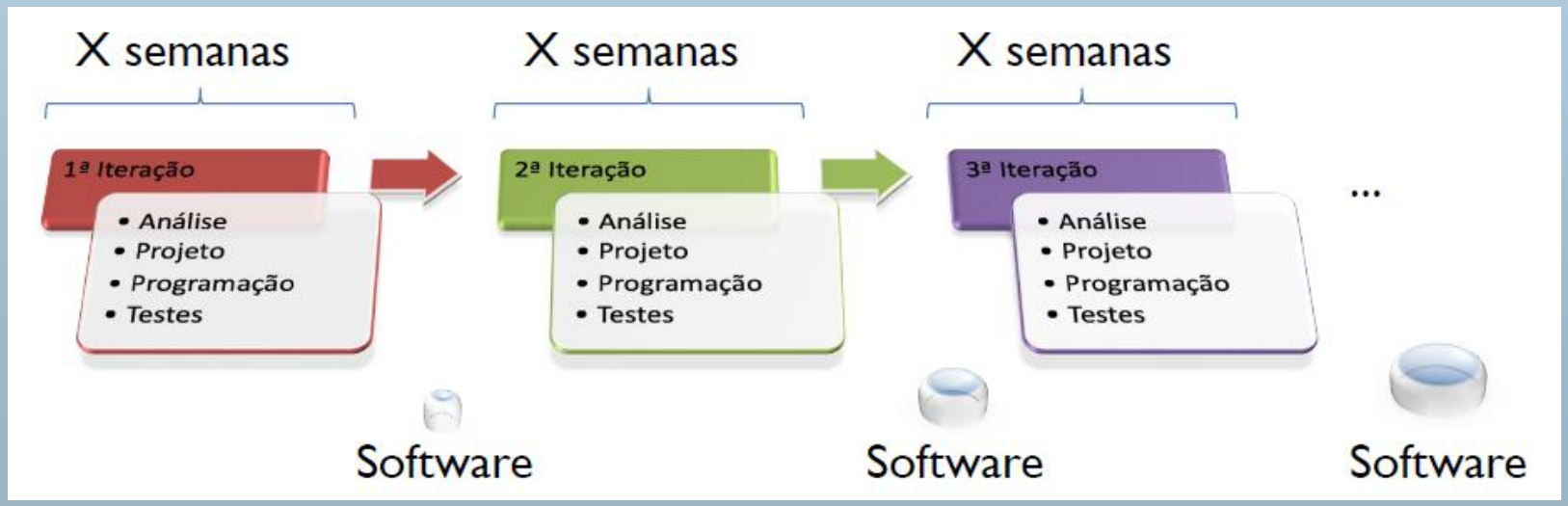


Desenvolvimento Iterativo de Software

- O Desenvolvimento Iterativo ocorre quando funcionalidades são especificadas, projetadas, construídas e testadas juntas em uma série de ciclos, geralmente com duração fixa.
  - Iterações podem envolver mudanças em funcionalidades desenvolvidas em iterações anteriores, juntamente com mudanças no escopo do projeto.
  - Cada iteração fornece software funcional que é um subconjunto crescente do sistema concebido até que o software final seja entregue ou o desenvolvimento seja interrompido.

Desenvolvimento Iterativo de Software (cont.)

- O Desenvolvimento Iterativo é organizado em “miniprojetos”.
  - Cada “miniprojeto” é uma iteração.
  - Cada iteração tem duração curta e fixa (de 2 a 6 semanas).
  - Cada iteração tem atividades de análise, projeto, programação e testes.
  - O produto de uma iteração é um software parcial.

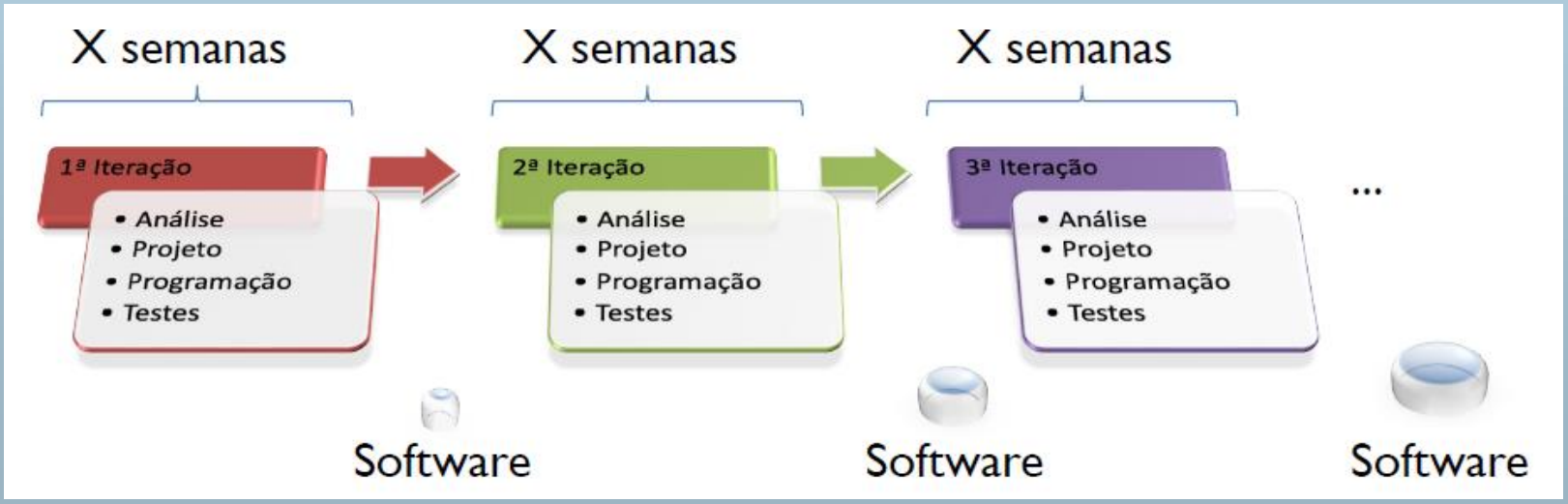


Desenvolvimento Iterativo de Software (cont.)

- A iteração deve ser fixa.
  - Tarefas podem ser removidas ou incluídas.
  - A iteração nunca deve passar da duração previamente estipulada.
- O resultado de cada iteração é um software.
  - Incompleto.
  - Em desenvolvimento (não pode ser colocado em produção).
  - Mas não é um protótipo!!!

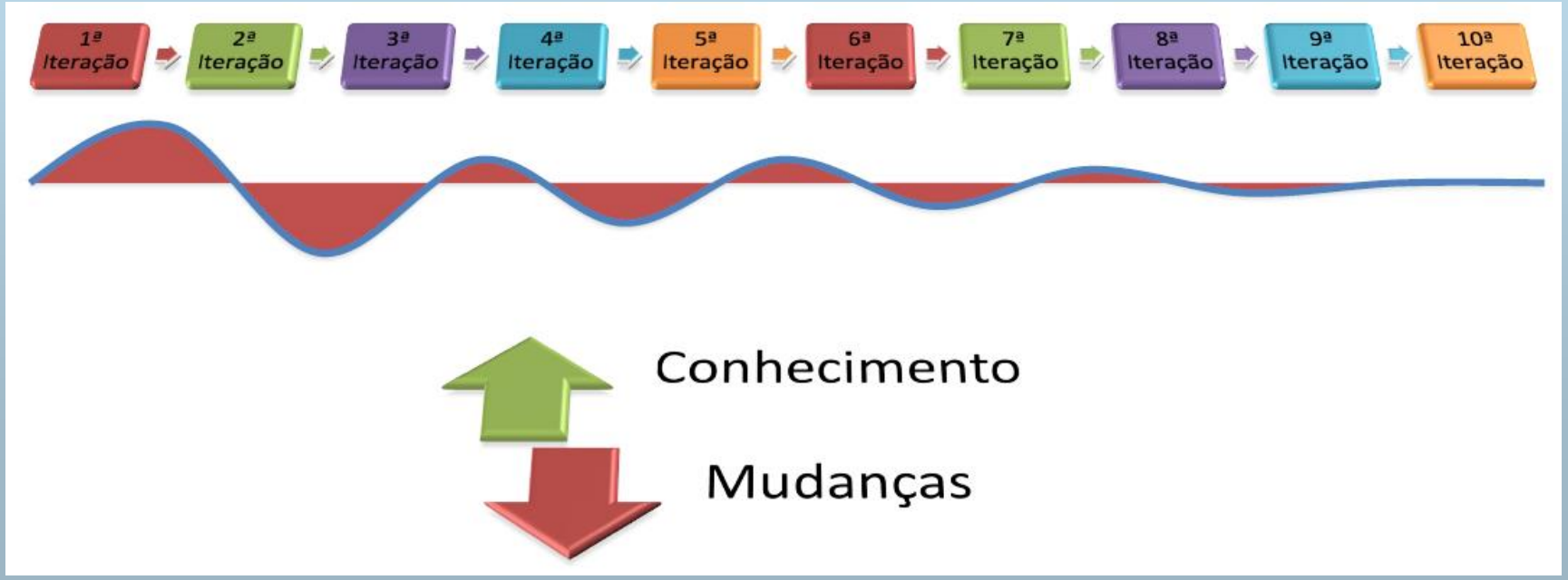
Desenvolvimento Iterativo de Software (cont.)

- Esse software pode ser verificado e validado parcialmente.
  - Por testes.
  - Pelos usuários.
- Podem ser necessárias diversas iterações (por exemplo, 10 a 15) para ter uma versão do sistema pronta para entrar em produção.



Desenvolvimento Iterativo de Software (cont.)

- Iterações curtas privilegiam a propagação de conhecimento.
  - Aumento do conhecimento sobre o software.
  - Diminuição das incertezas, que levam às mudanças.



Desenvolvimento Iterativo de Software (cont.)

- Nos Métodos Ágeis e em outros processos, há uma junção de Desenvolvimento Incremental e Desenvolvimento Iterativo.