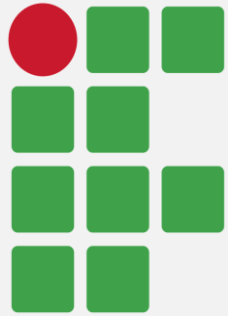
ENGENHARIA DE SOFTWARE

Introdução à Engenharia, Atividades e Processos de Software

Prof. Giovani Fonseca Ravagnani Disperati
IFSP – Câmpus Guarulhos

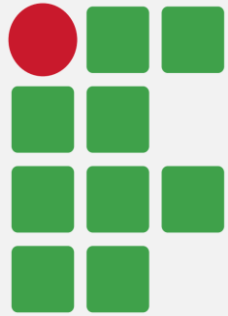
CONTEÚDO PROGRAMÁTICO



- Introdução à Engenharia de Software
- Crise do software
- O que é Engenharia de Software
- Produtos de software
- Características de um bom software
- Processos de Software
- Modelos de Processos de Software
- Atividades dos Processos de Software
- Melhorias de Processos de Software

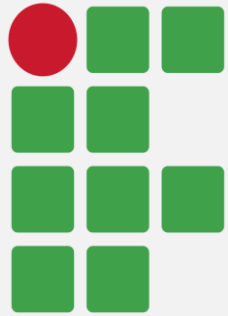
INTRODUÇÃO À ENGENHARIA DE SOFTWARE

ENGENHARIA DE SOFTWARE



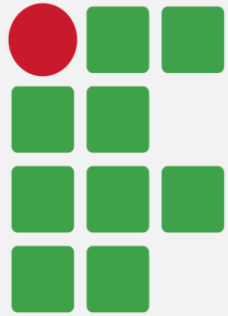
- As economias de TODAS as nações desenvolvidas são dependentes de software
- Cada vez mais sistemas são controlados por software
  - Atualmente é inconcebível pensar em um mundo sem sistemas de software

ENGENHARIA DE SOFTWARE



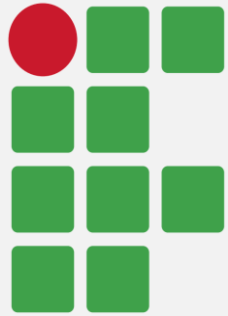
- Grande parte das infraestruturas e serviços das nações são controlados por sistemas computacionais
  - A maioria dos produtos elétricos possuem um computador e um software de controle
- O sistema financeiro é controlado de maneira informatizada, bem como a quase totalidade das indústrias e manufaturas

ENGENHARIA DE SOFTWARE



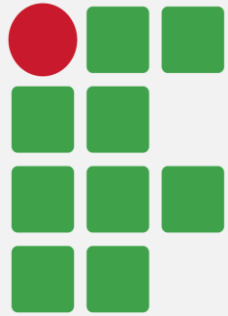
- Os sistemas de software, por sua vez, são intangíveis e abstratos
- Como afirma Sommerville (2012), “eles não são restringidos pelas propriedades materiais, nem governados pelas leis da física ou pelos processos de manufatura”;
    - Tais características intrínsecas aos softwares simplificam a engenharia de software, já que não há limites naturais para o potencial de software;

ENGENHARIA DE SOFTWARE



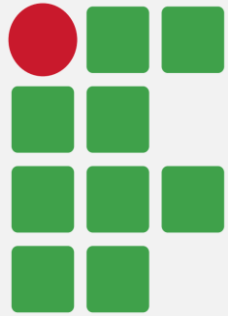
- Contudo, essas mesmas faltas de restrições físicas podem levar os sistemas de software a, rapidamente, se tornarem complexos, difíceis de entender e caros para alterar.

ENGENHARIA DE SOFTWARE



- Os custos de software geralmente dominam os custos de sistemas de computador
- Custos de software em um PC geralmente são maiores que o custo de hardware

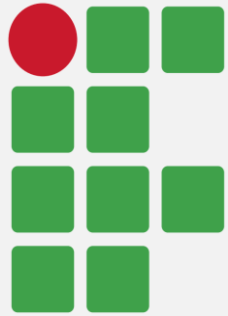
ENGENHARIA DE SOFTWARE



- Software custa mais para manter do que para desenvolver.
    - Para sistemas com vida útil longa, os custos de manutenção podem ultrapassar em várias vezes os custos de desenvolvimento
- A engenharia de software está preocupada, portanto, com o desenvolvimento de software **custo-efetivo (*cost-effective*)**

CRISE DO SOFTWARE

CRISE DO SOFTWARE



- Em meados dos anos 1960 ocorreu um expressivo número de falhas em grandes projetos de software
- Tais falhas afetaram mesmo grandes companhias, como a IBM, levando-as quase ao colapso

CRISE DO SOFTWARE

- A partir do final dos anos 1960 e começo dos anos 1970 intensifica-se a crise do Software.
- Em 1968 a OTAN propõe uma conferência sobre Engenharia de Software, a fim de reunir um número de estudiosos e cientistas para discutir os problemas de atrasos em projetos de sistemas, além do grande número de bugs e produtos inflexíveis.

CRISE DO SOFTWARE

- Nesta época, o que mais chocou os envolvidos foi que tais falhas se deram em um período de tempo em que o progresso da computação foi substancial: a criação das linguagens de terceira geração rodando sob processadores que aumentavam de capacidade exponencialmente.

- Ou seja, custo crescente do software e custo decrescente do hardware;

CRISE DO SOFTWARE

- Mesmo sob tais condições favoráveis os projetos estavam ficando significativamente acima do custo, atrasados ou, pior, estavam sendo abandonados

- Em 1972, Barry Boehm, que não havia frequentado a conferência da OTAN, alertava que os softwares estavam “por vezes atrasados e não-confiáveis, e que os custos estavam crescendo”.

CRISE DO SOFTWARE

- Um grande exemplo disso foi o IBM System/360, gerenciado por Frederick Brooks, que, em 1964, foi um dos maiores projetos já conduzidos.
- O tamanho do projeto, não visto até então, elevou em várias ordens de magnitude os problemas de programação até então conhecidos.
    - A contratação de mais e mais programadores, contudo, não resolveu tais problemas;

CRISE DO SOFTWARE

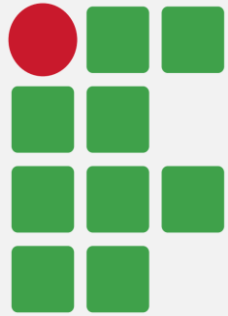
- A partir disto, em 1975, Brooks, em seu livro O mítico homem mês, defendeu a tese de que adicionar mais programadores em um projeto já atrasado o atrasa ainda mais.

O QUE É ENGENHARIA DE SOFTWARE

O que é Engenharia de Software

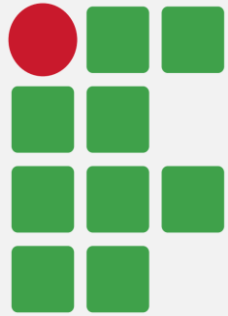
- Nesse âmbito da Crise do Software os métodos e processos de desenvolvimento de sistemas de software foram florescendo e a Engenharia de Software se consolidando enquanto campo de estudo, dada a necessidade de solucionar os problemas enfrentados

O que é Engenharia de Software



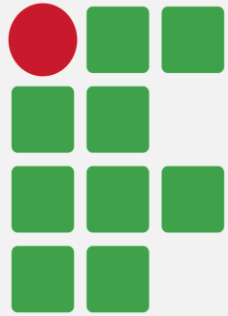
- Como podemos definir Engenharia de Software?
  - Para Pressman Engenharia de Software é uma **disciplina de engenharia que se encarrega de todos os aspectos da produção de um software**, dentro os quais podemos destacar

O que é Engenharia de Software



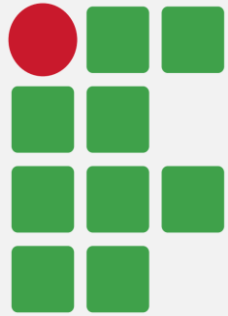
- **Processos**
  - Processos racionalizam o desenvolvimento de Software;
- **Métodos**
  - Conhecimento técnico, “como” fazer;
- **Ferramentas**
  - Suporte automatizado para os processos e métodos;

O que é Engenharia de Software



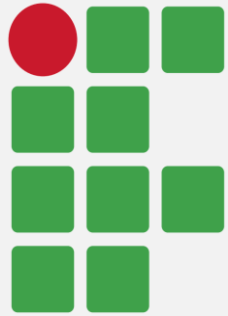
- Já para Sommerville, Engenharia de Software é uma **disciplina de engenharia que se preocupa com todos os aspectos da produção de software**, desde os estágios iniciais da especificação do sistema até a manutenção do sistema depois que ele entra em uso

O que é Engenharia de Software



- Obter software de qualidade com foco em desenvolvimento, operação e manutenção;
- Utilizar modelos, formalismos, técnicas e ferramentas para o desenvolvimento sistemático de software;
- Aplicar métodos, técnicas e ferramentas para a gestão do processo de desenvolvimento de software;
- Produção de documentação formal para comunicação entre equipe de desenvolvimento e usuários

O que é Engenharia de Software



- Podemos dizer que Engenharia de Software é um subconjunto da Engenharia de Sistemas – que abrange aspectos sistêmicos como hardware, pessoas e outros sistemas – incluindo-se, aí, a Engenharia de Software;

O que é Engenharia de Software

- Dentre os problemas gerais que afetam o Software e causam aumento da complexidade na Engenharia de Software, podemos citar
  - Heterogeneidade: cada vez mais os sistemas precisam operar como sistemas distribuídos em redes que incluem diferentes tipos de computadores e dispositivos móveis;

O que é Engenharia de Software

- Mudança social e de negócios: os negócios e a sociedade estão mudando de maneira incrivelmente rápida à medida que as economias emergentes se desenvolvem e novas tecnologias se tornam disponíveis. É preciso ser capaz de alterar o software existente e desenvolver rapidamente um novo software.

O que é Engenharia de Software

- Segurança e confiança: como o software está entrelaçado com todos os aspectos de nossas vidas, é essencial que possamos confiar nesse software.

- Escala: o software deve ser desenvolvido em uma ampla gama de escalas, desde sistemas embarcados muito pequenos em dispositivos portáteis até sistemas baseados em Internet que atendem a uma comunidade global.

O que é Engenharia de Software

- Além disso, existem muitos tipos diferentes de sistema de software e não existe um conjunto universal de técnicas de desenvolvimento software aplicáveis a todos eles.
- Os métodos e ferramentas de Engenharia de Software utilizados dependem do tipo de aplicativo que está sendo desenvolvido, dos requisitos do cliente e do background da equipe de desenvolvimento.

PRODUTOS DE SOFTWARE

Produtos de Software

- Produtos genéricos
  - Sistemas independentes que são comercializados e vendidos para qualquer cliente que deseje comprá-los
  - Exemplos: software para PC, como programas gráficos, ferramentas de gerenciamento de projetos, CAD; software para mercados específicos, como sistemas de consultas para dentistas, dentre outros

Produtos de Software

- Produtos customizados
  - Software encomendado por um cliente específico para atender às suas próprias necessidades
  - Exemplos - sistemas de controle embarcados, software de controle de tráfego aéreo, dentre outros

Produtos de Software

- Em relação às especificações dos Produtos de Software, pode-se dizer que:
  - Nos produtos genéricos, a especificação do que o software deve fazer é de propriedade do desenvolvedor do software e as decisões sobre alteração do software são tomadas pelo desenvolvedor.

Produtos de Software

- Já nos produtos customizados, a especificação do que o software deve fazer é de propriedade do cliente para o software e eles tomam decisões sobre as alterações de software necessárias.

Produtos de Software

- Em relação aos tipos de produtos de software, há uma grande variedade, dentre os quais:
  - Aplicações stand-alone: Estes são sistemas de aplicativos executados em um computador local, como um PC. Eles incluem todas as funcionalidades necessárias e não precisam estar conectados a uma rede.

Produtos de Software

- Aplicativos interativos baseados em transações: aplicativos executados em um computador remoto e acessados pelos usuários a partir de seus próprios PCs ou terminais. Isso inclui aplicativos da web, como aplicativos de comércio eletrônico.

Produtos de Software

- Sistemas de controle embarcados: estes são sistemas de controle de software que controlam e gerenciam dispositivos de hardware. Numericamente, provavelmente existem mais sistemas embarcados do que qualquer outro tipo de sistema.

Produtos de Software

- Sistemas de processamento em lote: estes são sistemas de negócios projetados para processar dados em grandes lotes. Eles processam um grande número de entradas individuais para criar saídas correspondentes.

Produtos de Software

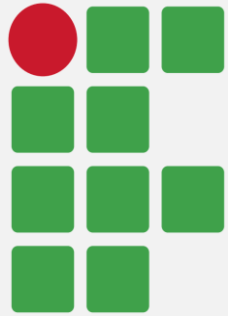
- Sistemas de entretenimento: estes são sistemas que são principalmente para uso pessoal e que visam entreter o usuário.
- Sistemas para modelagem e simulação: são sistemas desenvolvidos por cientistas e engenheiros para modelar processos ou situações físicas, que incluem muitos objetos separados e em interação.

Produtos de Software

- Sistemas de coleta de dados: estes são sistemas que coletam dados de seu ambiente usando um conjunto de sensores e enviam esses dados para outros sistemas para processamento.

CARACTERÍSTICAS DE UM BOM SOFTWARE

Características de um bom Software



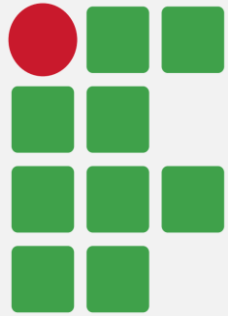
- Para o desenvolvimento de sistemas, alguns princípios fundamentais se aplicam a todos os tipos de softwares, independentemente das técnicas de desenvolvimento utilizadas.

Características de um bom Software

- Os sistemas devem ser desenvolvidos usando um processo de desenvolvimento gerenciado e compreendido.

- Obviamente, diferentes processos são usados para diferentes tipos de software.

Características de um bom Software



- Confiabilidade e desempenho são importantes para todos os tipos de sistema.

- É importante entender e gerenciar as especificações e os requisitos do software (o que o software deve fazer).

- Onde apropriado, você deve reutilizar o software que já foi desenvolvido, em vez de escrever um novo software.

Características de um bom Software

- Podemos entender por bom sistema aquele que está cumprindo todos os requisitos do cliente, está funcionando perfeitamente em todas as circunstâncias e que, por fim, mesmo que possua um escopo para melhorar, a manutenção e melhoria do software sejam eficazes quanto ao custo.

Características de um bom Software

Característica do Produto	Descrição
Manutenibilidade	O software deve ser escrito de tal maneira que possa evoluir para atender às novas necessidades dos clientes. Esse é um atributo crítico, pois a mudança de software é um requisito inevitável de um ambiente de negócios em mudança.
Confiabilidade e segurança	A confiabilidade do software inclui uma série de características, incluindo segurança e proteção. Um software confiável não deve causar danos físicos ou econômicos no caso de falha do sistema. Usuários mal-intencionados não devem poder acessar ou danificar o sistema.
Eficiência	O software não deve fazer uso desnecessário dos recursos do sistema, como ciclos de memória e processador. Portanto, a eficiência inclui capacidade de resposta, tempo de processamento, utilização de memória, etc.
Aceitabilidade	O software deve ser aceitável para o tipo de usuário para o qual foi projetado. Isso significa que deve ser compreensível, utilizável e compatível com outros sistemas que eles

PERGUNTAS E RESPOSTAS

Perguntas e respostas

Pergunta	Resposta
O que é software?	Softwares são programas de computador e documentação associada. Produtos de software podem ser desenvolvidos para um cliente específico ou para o mercado em geral.
Quais são os atributos de um bom software?	Um bom software deve prover a funcionalidade e o desempenho requeridos pelo usuário; além disso, deve ser confiável e fácil de manter e usar.
O que é Engenharia de Software?	É uma disciplina de engenharia que se preocupa com todos os aspectos de produção de software.
Quais são as principais atividades da engenharia de software?	Especificação de software, desenvolvimento de software, validação de software e evolução de software.

Perguntas e respostas

Pergunta	Resposta
Qual a diferença entre engenharia de software e ciência da computação?	Ciência da computação foca a teoria e os fundamentos; engenharia de software preocupa-se com o lado prático do desenvolvimento e entrega de softwares úteis.
Quais são os principais desafios da engenharia de software?	Lidar com o aumento de diversidade, demandas pela diminuição do tempo para entrega e desenvolvimento de software confiável.
Qual a diferença entre engenharia de software e engenharia de sistemas?	Engenharia de sistemas se preocupa com todos os aspectos do desenvolvimento de sistemas computacionais, incluindo engenharia de hardware, software e processo. Engenharia de software é uma parte específica desse processo mais genérico.

Perguntas e respostas

Pergunta	Resposta
Quais são os custos da engenharia de software?	Aproximadamente 60% dos custos de software são de desenvolvimento; 40% são custos de testes. Para software customizado, os custos de evolução frequentemente superam os custos de desenvolvimento.
Quais são as melhores técnicas e métodos da engenharia de software?	Enquanto todos os projetos de software devem ser gerenciados e desenvolvidos profissionalmente, técnicas diferentes são adequadas para tipos de sistemas diferentes. Por exemplo, jogos devem ser sempre desenvolvidos usando uma série de protótipos, enquanto sistemas de controle críticos de segurança requerem uma especificação analisável e completa. Portanto, não se pode dizer que um método é melhor que outro

Perguntas e respostas

Pergunta	Resposta
Quais diferenças foram feitas pela Internet na engenharia de software?	A Internet tornou serviços de software disponíveis e possibilitou o desenvolvimento de sistemas altamente distribuídos baseados em serviços. O desenvolvimento de sistemas baseados em Web gerou importantes avanços nas linguagens de programação e reúso de software.

PROCESSOS DE SOFTWARE

Processos de software

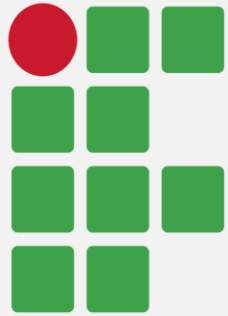
- Segundo Sommerville, um processo de software é “um conjunto de atividades relacionadas que levam a produção de um produto de software.”

- Tais atividades podem envolver o desenvolvimento de software a partir do zero bem como a extensão e modificação de sistemas existentes ou mesmo a configuração e integração de componentes de um sistema existente com “sistemas de prateleira”

Processos de software

- Para Sommerville, todos os processos de software devem incluir ao menos quatro atividades: Especificação, Projeto e implementação, Validação e Evolução.

Processos de software



- Especificação de Software – onde as funcionalidades do software e as restrições ao seu funcionamento devem ser definidas.
- Projeto e implementação de Software – onde o software é construído de forma a atender às especificações.

Processos de software

- Validação de Software – o software deve ser validado para garantir que atende às demandas do cliente.
- Evolução de Software – o software deve evoluir para atender às necessidades de mudanças dos clientes.

Processos de software

- De alguma forma, essas atividades fazem parte de todos os processos de software.
- Na prática, são atividades complexas em si mesmas, que incluem subatividades como validação de requisitos, projeto de arquitetura, testes unitários, etc.

Processos de software

- Existem também as atividades que dão apoio ao processo, como documentação e gerenciamento de configuração de software;

Processos de software

- Os processos de software são complexos e, como todos os processos intelectuais e criativos, dependem de pessoas para tomar decisões e fazer julgamentos.

- Não existe um processo ideal, a maioria das organizações desenvolve os próprios processos de desenvolvimento de software;

Processos de software

- Os processos têm evoluído de maneira a tirarem melhor proveito das capacidades das pessoas em uma organização, bem como das características específicas do sistema em desenvolvimento.

Processos de software

- Para alguns sistemas, como sistemas críticos, é necessário um processo de desenvolvimento muito bem estruturado.

- Já para sistemas de negócios, com requisitos que se alteram rapidamente, provavelmente será mais eficaz um processo menos formal e mais flexível.

Processos de software

- Ao descrever e discutir os processos, costumamos falar sobre suas atividades, como a especificação de um modelo de dados, o projeto de interface de usuário etc., bem como a organização dessas atividades.

Processos de software

- No entanto, assim como as atividades, as descrições do processo também podem incluir:
  - 1. Produtos, que são os resultados de uma das atividades do processo. Por exemplo, o resultado da atividade de projeto de arquitetura pode ser um modelo da arquitetura de software;
  - 2. Papéis, que refletem as responsabilidades das pessoas envolvidas no processo. Exemplos de papéis são: gerente de projeto, gerente de configuração, programador etc;

Processos de software

- 3. Pré e pós-condições, que são declarações verdadeiras antes e depois de uma atividade do processo ou da produção de um produto.

- Por exemplo, antes do projeto de arquitetura ser iniciado, pode haver uma pré-condição de que todos os requisitos tenham sido aprovados pelo cliente e, após a conclusão dessa atividade, uma pós-condição poderia ser a de que os modelos UML que descrevem a arquitetura tenham sido revisados.

Processos de software

- Sommervile afirma que os processos de Software são normalmente categorizados como dirigidos a planos ou processos ágeis;
    - Na terminologia ágil, muitos se referem à “processos pesados” vs “processos leves/ágeis”

Processos de software

- Processos dirigidos a planos são aqueles em que as atividades são planejadas com antecedência, sendo o processo avaliado por comparação com o planejamento inicial.

- Já nos processos ágeis, o planejamento acontece de forma gradativa, sendo mais fácil alterar o processo de maneira a refletir as necessidades de mudanças dos clientes.

Processos de software

- Processos dirigidos a planos são aqueles em que as atividades são planejadas com antecedência, sendo o processo avaliado por comparação com o planejamento inicial.

- Já nos processos ágeis, o planejamento acontece de forma gradativa, sendo mais fácil alterar o processo de maneira a refletir as necessidades de mudanças dos clientes.

Processos de software

- Embora não exista um processo “ideal” de software, há espaço, em muitas organizações, para melhorias no processo de software;

- Os processos podem incluir técnicas ultrapassadas ou não aproveitar as melhores práticas de engenharia de software da indústria.

    - De fato, muitas empresas ainda não se aproveitam dos métodos da engenharia de software em seu desenvolvimento de software;

MODELOS DE PROCESSOS DE SOFTWARE

Modelos de processos de software

- Um modelo de processo de Software é uma representação simplificada de um processo de Software.

- Cada modelo representa uma perspectiva de um processo, fornecendo informações parciais sobre ele;
    - Por exemplo, um modelo de atividade do processo pode mostrar as atividades e sua sequência, mas não mostrar os papéis das pessoas envolvidas

Modelos de processos de software

- Esses modelos genéricos, entretanto, não são descrições definitivas dos processos de software. Pelo contrário, são abstrações que podem ser usadas para explicar diferentes abordagens de desenvolvimento de software.

- Eles podem ser vistos como frameworks de processos que podem ser ampliados e adaptados para criar processos de engenharia de software mais específicos.

Modelos de processos de software

- Desenvolvimento em cascata
  - Esse modelo considera as atividades fundamentais do processo de especificação, desenvolvimento, validação e evolução, e representa cada uma delas como fases distintas, como: especificação de requisitos, projeto de software, implementação, teste e assim por diante.

Modelos de processos de software

- Desenvolvimento incremental.
  - Essa abordagem intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona funcionalidade a anterior.

Modelos de processos de software

- Engenharia de software orientada a reuso.
  - Essa abordagem é baseada na existência de um número significativo de componentes reusáveis. O processo de desenvolvimento do sistema concentra-se na integração desses componentes em um sistema já existente em vez de desenvolver um sistema a partir do zero.

Modelos de processos de software

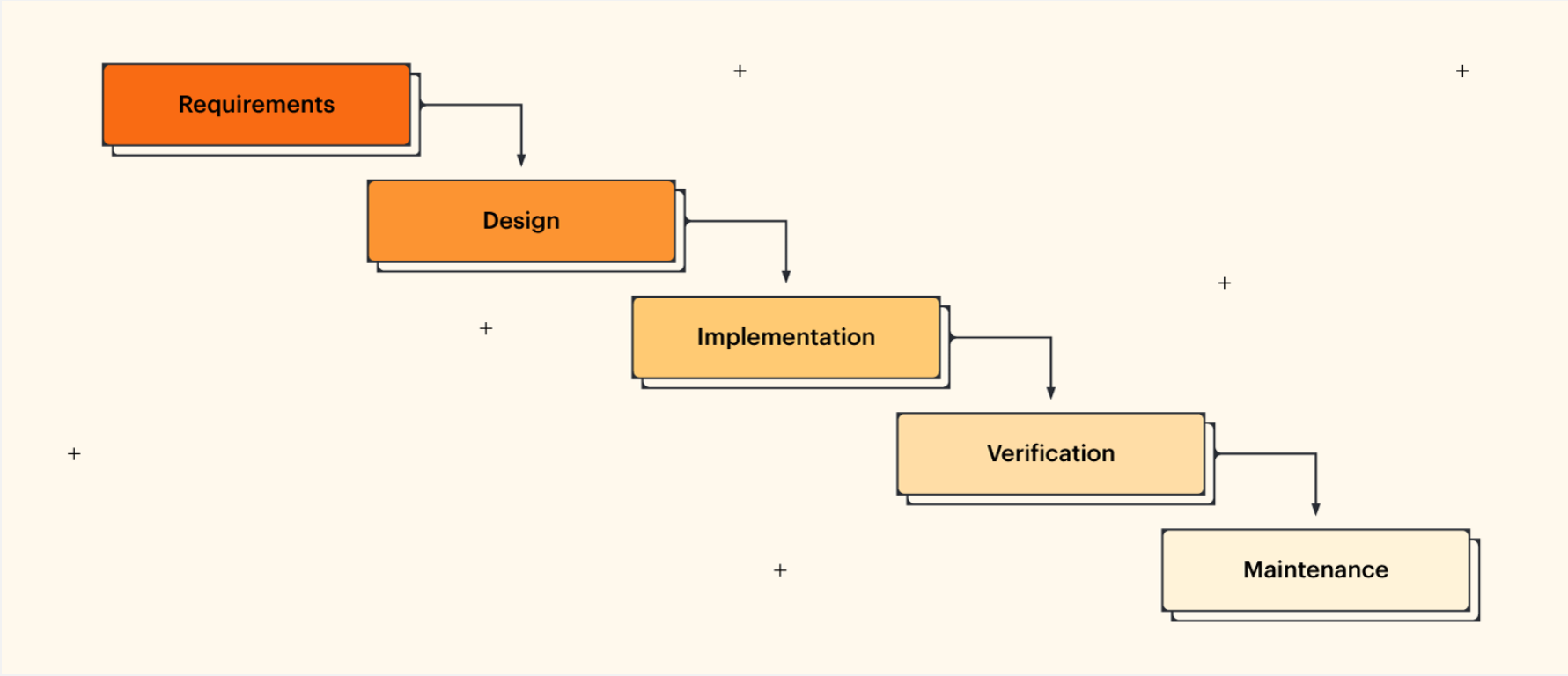
- Para Sommerville
  - “Esses modelos não são mutuamente exclusivos e muitas vezes são usados em conjunto, especialmente para o desenvolvimento de sistemas de grande porte. Para sistemas de grande porte, faz sentido combinar algumas das melhores características do modelo em cascata e dos modelos de desenvolvimento incremental. [...]”

MODELO CASCATA

Modelo cascata

- O primeiro modelo do processo de desenvolvimento de software a ser publicado foi derivado de processos mais gerais da engenharia de sistemas (ROYCE, 1970).
- Devido ao encadeamento entre uma fase e outra esse modelo é conhecido como “modelo em cascata”.

Modelo cascata



Modelo cascata

- O modelo em cascata é um exemplo de um processo dirigido a planos — em princípio, você deve planejar e programar todas as atividades do processo antes de começar a trabalhar nelas.

Modelo cascata

- Os principais estágios do modelo em cascata refletem diretamente as atividades fundamentais do desenvolvimento:
  - 1. Análise e definição de requisitos. Os serviços, restrições e metas do sistema são estabelecidos por meio de consulta aos usuários. Em seguida, são definidos em detalhes e funcionam como uma especificação do sistema;

Modelo cascata

- 2. Projeto de sistema e software. O processo de projeto de sistemas aloca os requisitos tanto para sistemas de hardware como para sistemas de software, por meio da definição de uma arquitetura geral do sistema. O projeto de software envolve identificação e descrição das abstrações fundamentais do sistema de software e seus relacionamentos;

Modelo cascata

- 3. Implementação e teste unitário. Durante esse estágio, o projeto do software é desenvolvido como um conjunto de programas ou unidades de programa. O teste unitário envolve a verificação de que cada unidade atenda a sua especificação;

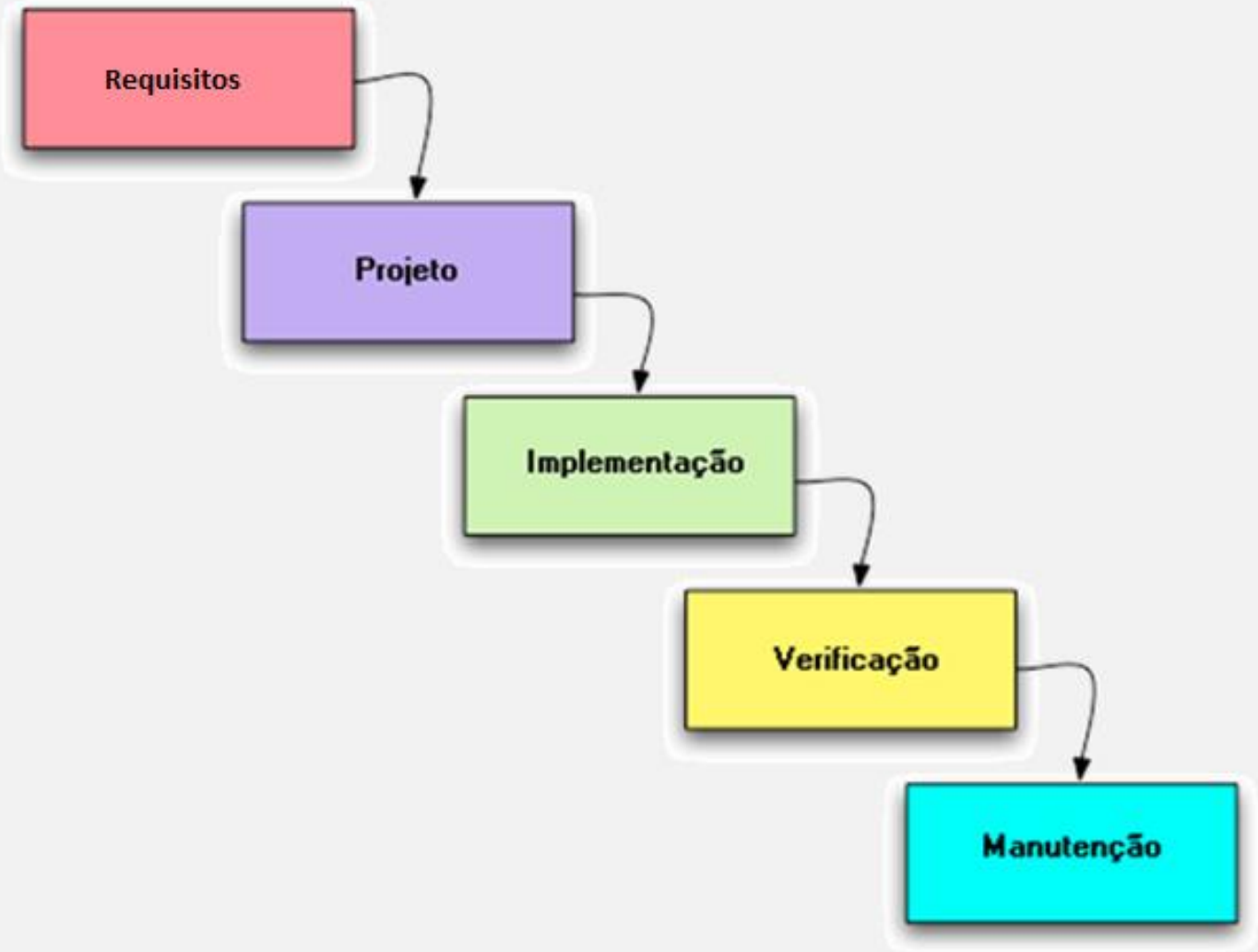
Modelo cascata

- 4. Integração e teste de sistema. As unidades individuais do programa ou programas são integradas e testadas como um sistema completo para assegurar que os requisitos do software tenham sido atendidos. Após o teste, o sistema de software é entregue ao cliente;

Modelo cascata

- 5. Operação e manutenção. Normalmente (embora não necessariamente), essa é a fase mais longa do ciclo de vida. O sistema é instalado e colocado em uso. A manutenção envolve a correção de erros que não foram descobertos em estágios iniciais do ciclo de vida, com melhora da implementação das unidades do sistema e ampliação de seus serviços em resposta as descobertas de novos requisitos;

Modelo cascata



Modelo cascata

- Em princípio, o resultado de cada estágio é a aprovação de um ou mais documentos ('assinados'). O estágio seguinte não deve ser iniciado até que a fase anterior seja concluída.

- Na prática, esses estágios se sobrepõem e alimentam uns aos outros de informações (cascata com retroalimentação).

Modelo cascata

- Durante o projeto, os problemas com os requisitos são identificados.

- Durante a codificação, problemas de projeto são encontrados; e assim por diante, nas demais fases do processo.

- O processo de software não é um modelo linear simples, pois envolve o feedback de uma fase para outra. Assim, os documentos produzidos em cada fase podem ser modificados para refletir as alterações feitas em cada um deles.

Modelo cascata

- Devido os custos de produção e aprovação de documentos, as iterações podem ser dispendiosas e envolver significativo retrabalho.

- Assim, após um pequeno número de iterações, é normal se congelarem partes do desenvolvimento, como a especificação, e dar-se continuidade aos estágios posteriores do processo.

Modelo cascata

- A solução dos problemas fica para mais tarde, ignorada ou programada, quando possível.

- Esse congelamento prematuro dos requisitos pode significar que o sistema não fará o que o usuário quer. Também pode levar a sistemas mal estruturados, quando os problemas de projeto são contornados por “artifícios de implementação”.

Modelo cascata

- O modelo em cascata é consistente com outros modelos de processos de engenharia, e a documentação é produzida em cada fase do ciclo.

- Desta forma, o processo torna-se visível, e os gerentes podem monitorar o progresso de acordo com o plano de desenvolvimento.

Modelo cascata

- Seu maior problema é a divisão inflexível do projeto em estágios distintos.

- Os compromissos devem ser assumidos em um estágio inicial do processo, o que dificulta que atendam as mudanças de requisitos dos clientes.

Modelo cascata

- Em princípio, o modelo em cascata deve ser usado apenas quando os requisitos são bem compreendidos e pouco provavelmente venham a ser radicalmente alterados durante o desenvolvimento do sistema.

Modelo cascata

- Como é mais fácil usar um modelo de gerenciamento comum para todo o projeto, processos de software baseados no modelo em cascata ainda são comumente utilizados.

Modelo cascata

- O modelo final de Winston W. Royce incorporou melhoramentos sobre seu "modelo em cascata" inicial, ilustrando que o feedback poderia levar a soluções desde o projeto do design à especificação de requisitos

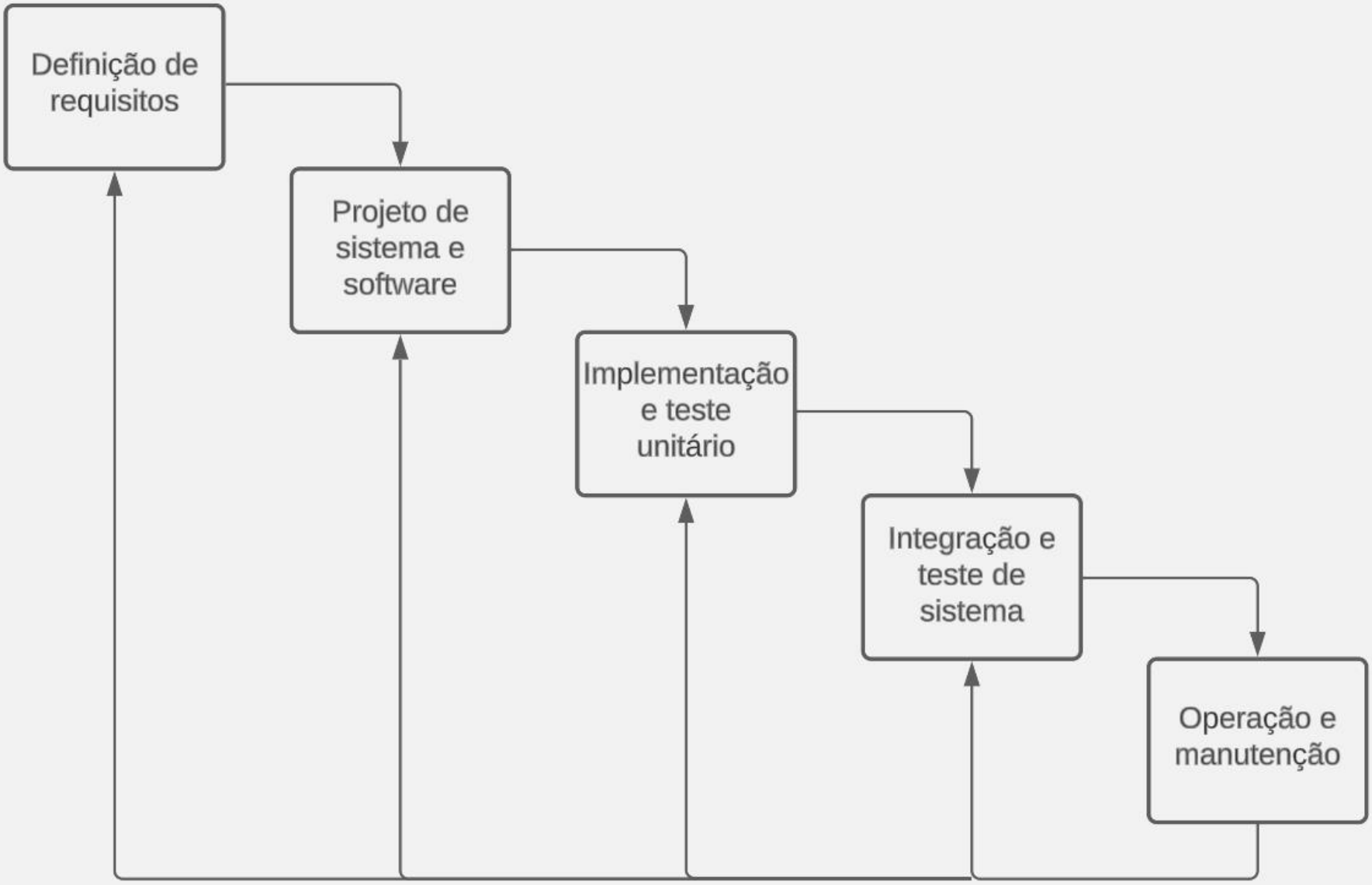
Modelo cascata

- O modelo final de Winston W. Royce incorporou melhoramentos sobre seu "modelo em cascata" inicial, ilustrando que o feedback poderia levar a soluções desde o projeto do design à especificação de requisitos.

Modelo cascata

- No mesmo artigo, Royce também defendia grandes quantidades de documentação, fazendo o trabalho "duas vezes se possível" (um sentimento semelhante ao de Fred Brooks, famoso por escrever o livro *Mythical Man Month*, um livro influente em gerenciamento de projetos de software, que defendia planejar, "jogar fora" e envolver o cliente o máximo possível).

Modelo cascata



Modelo cascata

- Uma variação importante do modelo em cascata é o desenvolvimento formal de um sistema, em que se cria um modelo matemático de uma especificação do sistema.

- Esse modelo é refinado, usando transformações matemáticas que preservam sua consistência, em código executável.

Modelo cascata

- Partindo do pressuposto de que suas transformações matemáticas estão corretas, você pode, portanto, usar um forte argumento de que um programa gerado dessa forma é consistente com suas especificações.

Modelo cascata

- Processos formais de desenvolvimento, como os baseados no método B (SCHNEIDER, 2001; WORDSWORTH, 1996), são particularmente adequados para o desenvolvimento de sistemas com requisitos rigorosos de segurança, confiabilidade e proteção.

Modelo cascata

- A abordagem formal simplifica a produção de casos de segurança ou proteção.

- Isso demonstra aos clientes ou reguladores que o sistema realmente cumpre com seus requisitos de proteção ou segurança.

Modelo cascata

- Processos baseados em transformações formais são geralmente usados apenas no desenvolvimento de sistemas críticos de segurança ou de proteção.

- Eles exigem conhecimentos especializados.

- Para a maioria dos sistemas, esse processo não oferece custo-benefício significativo sobre outras abordagens para o desenvolvimento de sistemas.

Modelo cascata

- Processos baseados em transformações formais são geralmente usados apenas no desenvolvimento de sistemas críticos de segurança ou de proteção.

- Eles exigem conhecimentos especializados.

- Para a maioria dos sistemas, esse processo não oferece custo-benefício significativo sobre outras abordagens para o desenvolvimento de sistemas.

MODELO INCREMENTAL

Modelo incremental

- O desenvolvimento incremental é baseado na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido.

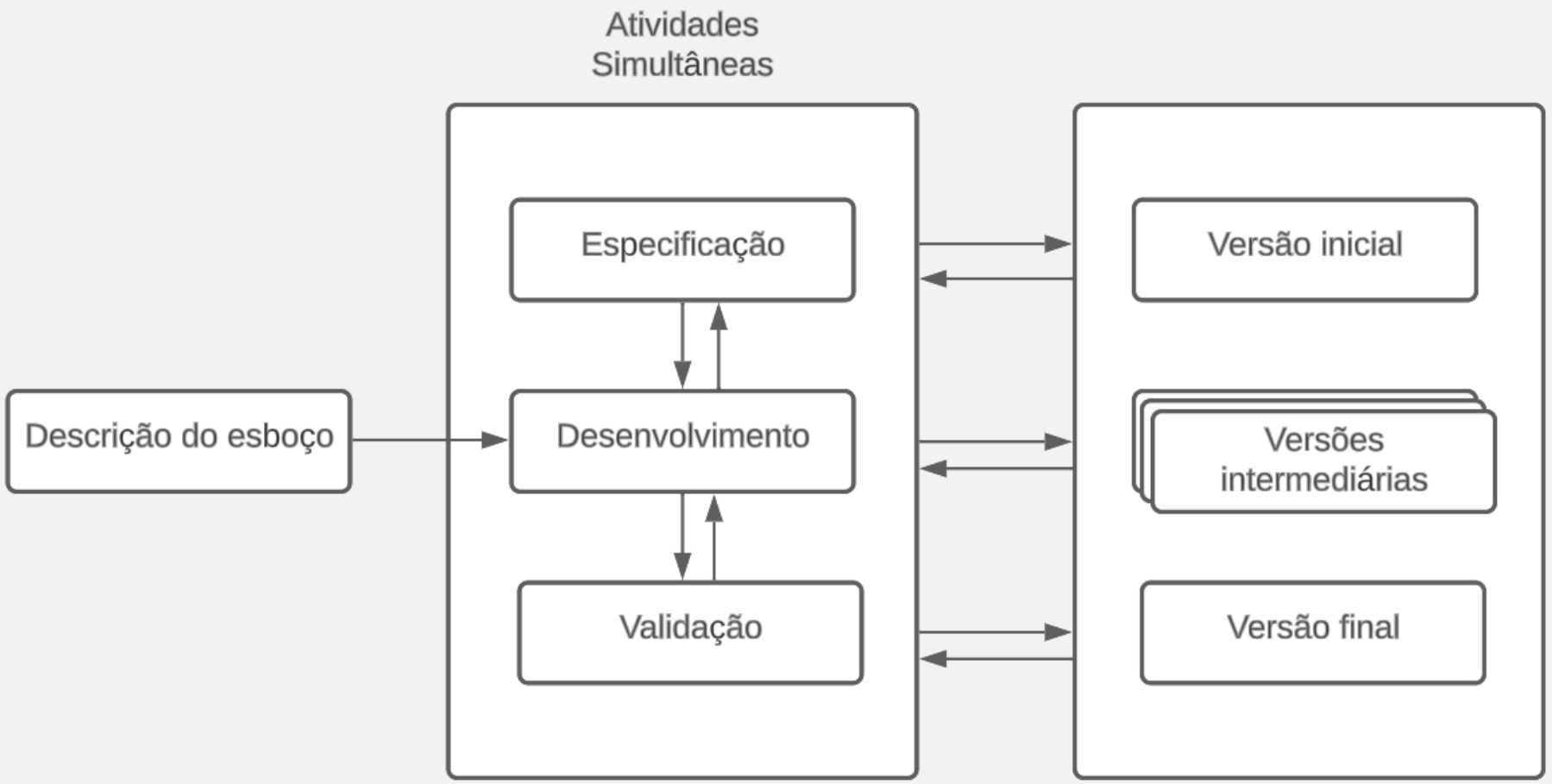
- Atividades de especificação, desenvolvimento e validação são intercaladas, e não separadas, com rápido feedback entre todas as atividades.

Modelo incremental

- O desenvolvimento incremental de software, que é uma parte fundamental das abordagens ágeis, é melhor do que uma abordagem em cascata para a maioria dos sistemas de negócios, e-commerce e sistemas pessoais.

- Desenvolvimento incremental reflete a maneira como resolvemos os problemas.
  - Raramente elaboramos uma solução completa do problema com antecedência; geralmente movemo-nos passo a passo em direção a uma solução, recuando quando percebemos que cometemos um erro.

Modelo incremental

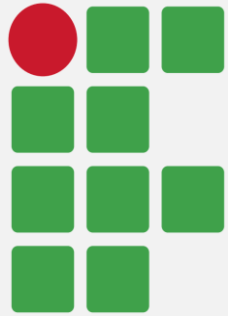


Modelo incremental

- Ao desenvolver um software de forma incremental, é mais barato e mais fácil fazer mudanças no software durante seu desenvolvimento.

- Cada incremento ou versão do sistema incorpora alguma funcionalidade necessária para o cliente:
  - Frequentemente, os incrementos iniciais incluem a funcionalidade mais importante ou mais urgente.

Modelo incremental



- Isso significa que o cliente pode avaliar o sistema em um estágio relativamente inicial do desenvolvimento para ver se ele oferece o que foi requisitado.

- Em caso negativo, só o incremento que estiver em desenvolvimento no momento precisará ser alterado e, possivelmente, nova funcionalidade deverá ser definida para incrementos posteriores.

Modelo incremental

- O desenvolvimento incremental tem três diferenças importantes quando comparado ao modelo em cascata:

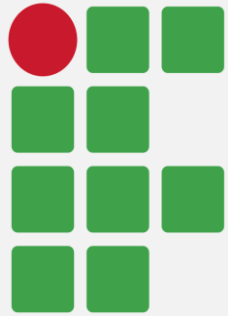
  - 1. O custo de acomodar as mudanças nos requisitos do cliente e reduzido. A quantidade de analise e documentação a ser refeita e muito menor do que o necessário no modelo em cascata.

Modelo incremental

- 2. É mais fácil obter feedback dos clientes sobre o desenvolvimento que foi feito.

- Os clientes podem fazer comentários sobre as demonstrações do software e ver o quanto foi implementado. Os clientes têm dificuldade em avaliar a evolução por meio de documentos de projeto de software.

Modelo incremental



- 3. É possível obter entrega e implementação rápida de um software útil ao cliente, mesmo se toda a funcionalidade não for incluída.

- Os clientes podem usar e obter ganhos a partir do software inicial antes do que é possível com um processo em cascata.

Modelo incremental

- O desenvolvimento incremental, atualmente, é a abordagem mais comum para o desenvolvimento de sistemas.

- Essa abordagem pode ser tanto dirigida a planos, ágil, ou, o mais comum, uma mescla dessas abordagens.

- Em uma abordagem dirigida a planos, os incrementos do sistema são identificados previamente; se uma abordagem ágil for adotada, os incrementos iniciais são identificados, mas o desenvolvimento de incrementos posteriores depende do progresso e das prioridades dos clientes.

Modelo incremental

- O desenvolvimento incremental, atualmente, é a abordagem mais comum para o desenvolvimento de sistemas.

- Essa abordagem pode ser tanto dirigida a planos, ágil, ou, o mais comum, uma mescla dessas abordagens.

- Em uma abordagem dirigida a planos, os incrementos do sistema são identificados previamente; se uma abordagem ágil for adotada, os incrementos iniciais são identificados, mas o desenvolvimento de incrementos posteriores depende do progresso e das prioridades dos clientes.

Modelo incremental

- Dentre os modelos incrementais, um dos mais conhecidos é o modelo espiral de Boehm.

- Trata-se de um framework de processo de software dirigido a riscos, proposto por Boehm em 1988.

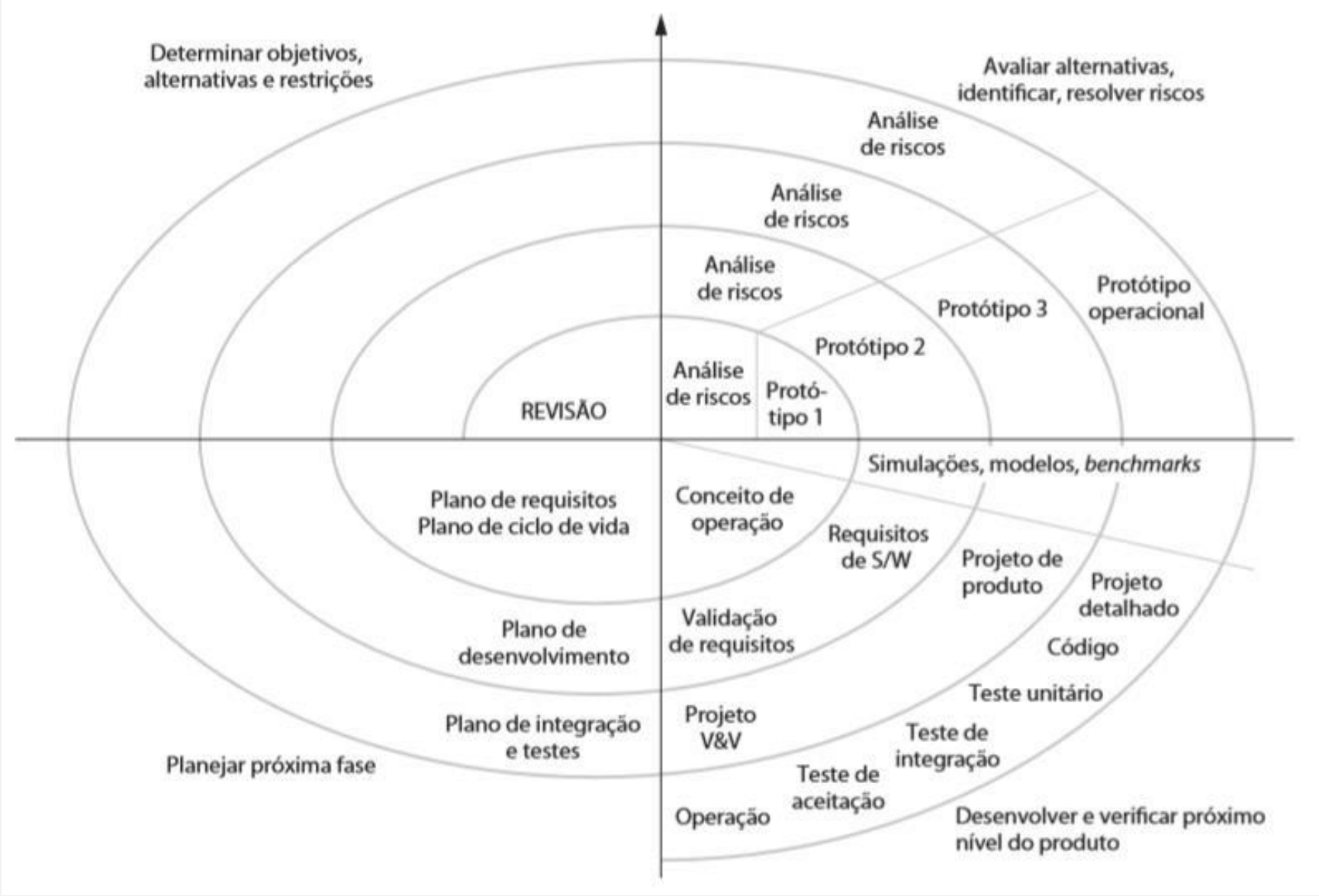
Modelo incremental

- No modelo de Boehm o processo de software é representado como uma espiral, e não como uma sequência de atividades com alguns retornos de uma para outra.

- Cada volta na espiral representa uma fase do processo de software.

- Dessa forma, a volta mais interna pode preocupar-se com a viabilidade do sistema; o ciclo seguinte, com definição de requisitos; o seguinte, com o projeto do sistema, e assim por diante.

Modelo incremental



Modelo incremental

- O modelo em espiral combina prevenção e tolerância a mudanças, assume que mudanças são um resultado de riscos de projeto e inclui atividades explícitas de gerenciamento de riscos para sua redução.

- Cada volta da espiral é dividida em quatro setores:
  - 1. Definição de objetivos. Objetivos específicos para essa fase do projeto são definidos; restrições ao processo e ao produto são identificadas, e um plano de gerenciamento detalhado é elaborado; os riscos do projeto são identificados. Podem ser planejadas estratégias alternativas em função desses riscos.

Modelo incremental

- 2. Avaliação e redução de riscos. Para cada um dos riscos identificados do projeto, é feita uma análise detalhada. Medidas para redução do risco são tomadas.
    - Por exemplo, se houver risco dos requisitos serem inadequados, um protótipo de sistema pode ser desenvolvido.

Modelo incremental

- 3. Desenvolvimento e validação. Após a avaliação dos riscos, é selecionado um modelo de desenvolvimento para o sistema.
    - Por exemplo, a prototipação descartável pode ser a melhor abordagem de desenvolvimento de interface de usuário se os riscos forem dominantes.

Modelo incremental

- Ou seja:
  - Se os riscos de segurança forem a principal consideração, o desenvolvimento baseado em transformações formais pode ser o processo mais adequado, e assim por diante.
  - Se o principal risco identificado for a integração de subsistemas, o modelo em cascata pode ser a melhor opção.

Modelo incremental

- 4. Planejamento. O projeto é revisado, e uma decisão é tomada a respeito da continuidade do modelo com mais uma volta da espiral. Caso se decida pela continuidade, planos são elaborados para a próxima fase do projeto.

Modelo incremental

- A principal diferença entre o modelo espiral e outros modelos de processo de software é seu reconhecimento explícito do risco.

- Um ciclo da espiral começa com a definição de objetivos, como desempenho e funcionalidade.
    - Em seguida, são enumeradas formas alternativas de atingir tais objetivos e de lidar com as restrições de cada um deles.

Modelo incremental

- Cada alternativa é avaliada em função de cada objetivo, e as fontes de risco do projeto são identificadas.

- O próximo passo é resolver esses riscos por meio de atividades de coleta de informações, como análise mais detalhada, prototipação e simulação.

Modelo incremental

- Após a avaliação dos riscos, algum desenvolvimento é efetivado, seguido por uma atividade de planejamento para a próxima fase do processo.

- De maneira informal dizemos que o risco significa, simplesmente, algo que pode dar errado.

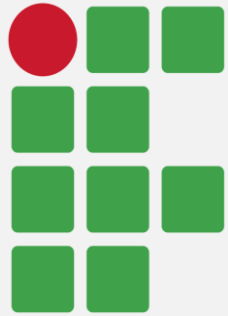
Modelo incremental

- Do ponto de vista do gerenciamento, a abordagem incremental tem dois problemas:
  - 1. O processo não é visível. Os gerentes precisam de entregas regulares para mensurar o progresso. Se os sistemas são desenvolvidos com rapidez, não é economicamente viável produzir documentos que reflitam cada uma das versões do sistema.

Modelo incremental

- 2. A estrutura do sistema tende a se degradar com a adição dos novos incrementos. A menos que tempo e dinheiro sejam despendidos em refatoração para melhoria do software, as constantes mudanças tendem a corromper sua estrutura. Incorporar futuras mudanças do software torna-se cada vez mais difícil e oneroso.

Modelo incremental



- Os problemas do desenvolvimento incremental são particularmente críticos para os sistemas de vida-longa, grandes e complexos, nos quais várias equipes desenvolvem diferentes partes do sistema.

Modelo incremental

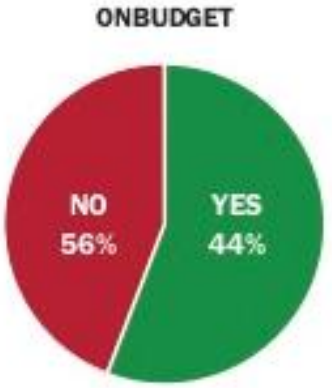
- Sistemas de grande porte necessitam de um framework ou arquitetura estável, e as responsabilidades das diferentes equipes de trabalho do sistema precisam ser claramente definidas, respeitando essa arquitetura.

- Isso deve ser planejado com antecedência, e não desenvolvido de forma incremental.

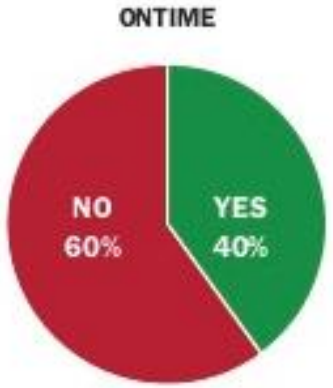
Modelo incremental

- Uma das principais causas devido à qual a maioria dos projetos de desenvolvimento de software falham é a escolha do modelo - por isso, esta deve ser feita com grande preocupação.

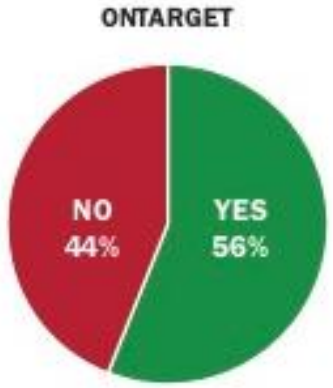
Modelo incremental



The percentage of projects that were OnBudget from FY2011–2015 within the new CHAOS database.



The percentage of projects that were OnTime from FY2011–2015 within the new CHAOS database.



The percentage of projects that were On Target from FY2011–2015 within the new CHAOS database.

TRADITIONAL RESOLUTION FOR ALL PROJECTS

	2011	2012	2013	2014	2015
SUCCESSFUL	39%	37%	41%	36%	36%
CHALLENGED	39%	46%	40%	47%	45%
FAILED	22%	17%	19%	17%	19%

The Traditional resolution of all software projects from FY2011–2015 within the new CHAOS database.

MODELO ORIENTADO A REUSO

Modelo orientado a reuso

- Na maioria dos projetos de software, há algum reuso de software. Isso acontece muitas vezes informalmente, quando as pessoas envolvidas no projeto sabem de projetos ou códigos semelhantes ao que é exigido.

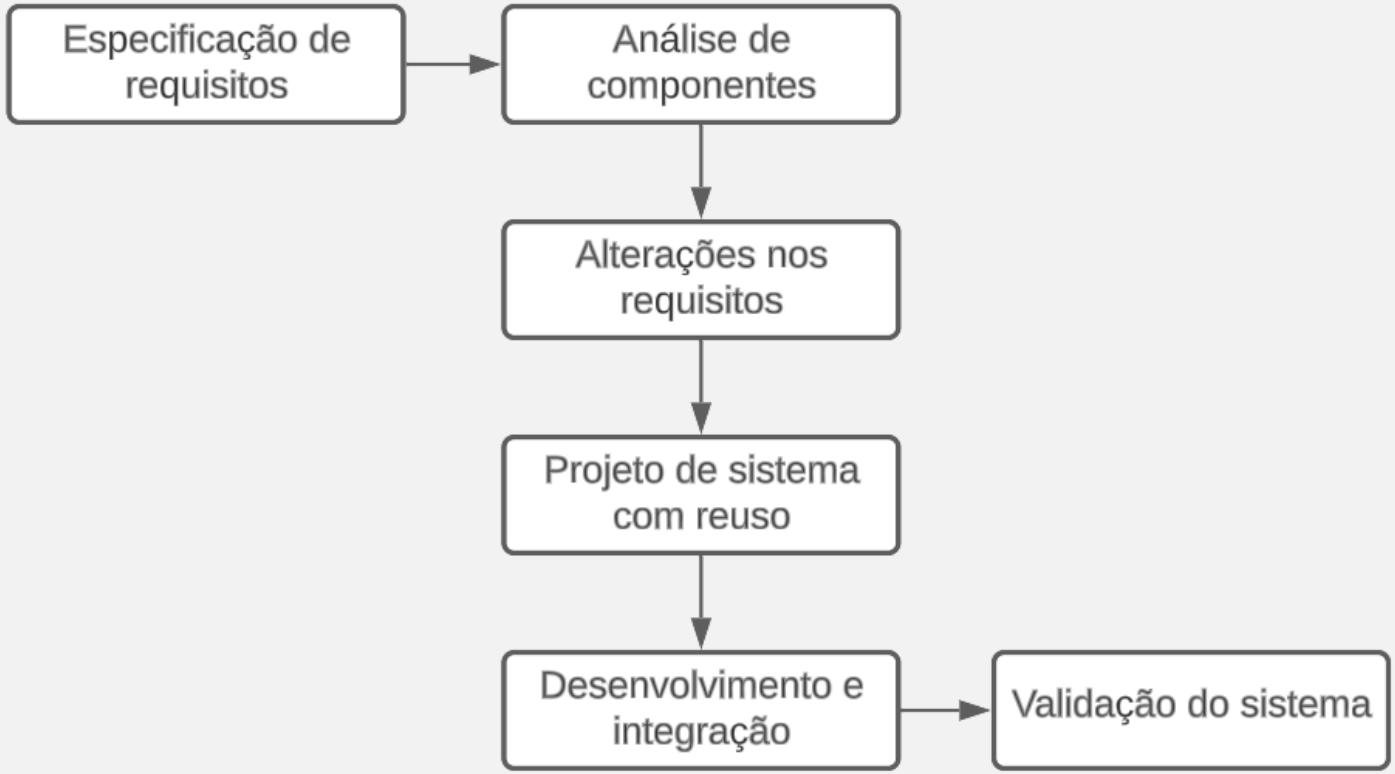
- Elas os buscam, fazem as modificações necessárias e incorporam-nos a seus sistemas.

Modelo orientado a reuso

- Esse reuso informal ocorre independentemente do processo de desenvolvimento que se use. No entanto, nos últimos anos, processos de desenvolvimento de software com foco no reuso de software existente tornaram-se amplamente usados.

Modelo orientado a reuso

- Um modelo de processo geral de desenvolvimento baseado no reuso é mostrado na imagem abaixo.



Modelo orientado a reuso

- Esses estágios são:
  - 1. Análise de componentes. Dada a especificação de requisitos, é feita uma busca por componentes para implementar essa especificação. Em geral, não há correspondência exata, e os componentes que podem ser usados apenas fornecem alguma funcionalidade necessária;

Modelo orientado a reuso

- 2. Modificação de requisitos. Durante esse estágio, os requisitos são analisados usando-se informações sobre os componentes que foram descobertos. Em seguida, estes serão modificados para refletir os componentes disponíveis. No caso de modificações impossíveis, a atividade de análise dos componentes pode ser reinserida na busca por soluções alternativas.

Modelo orientado a reuso

- 3. Projeto do sistema com reuso. Durante esse estágio, o framework do sistema é projetado ou algo existente e reusado. Os projetistas têm em mente os componentes que serão reusados e organizam o framework para reuso. Alguns softwares novos podem ser necessários, se componentes reusáveis não estiverem disponíveis.

Modelo orientado a reuso

- 4. Desenvolvimento e integração. Softwares que não podem ser adquiridos externamente são desenvolvidos, e os componentes e sistemas COTS (*Commercial off-the-shelf*) são integrados para criar o novo sistema. A integração de sistemas, nesse modelo, pode ser parte do processo de desenvolvimento, em vez de uma atividade separada.

Modelo orientado a reuso

- Existem três tipos de componentes de software que podem ser usados em um processo orientado a reuso:
  - 1. Web services desenvolvidos de acordo com os padrões de serviço e que estão disponíveis para invocação remota.
  - 2. Coleções de objetos que são desenvolvidas como um pacote a ser integrado com um framework de componentes, como .NET ou JEE.
  - 3. Sistemas de software *stand-alone* configurados para uso em um ambiente particular.

Modelo orientado a reuso

- Engenharia de software orientada ao reuso tem a vantagem óbvia de reduzir a quantidade de software a ser desenvolvido e, assim, reduzir os custos e riscos.

- Geralmente, também proporciona a entrega mais rápida do software.

Modelo orientado a reuso

- No entanto, compromissos com os requisitos são inevitáveis, e isso pode levar a um sistema que não atende às reais necessidades dos usuários.

- Além disso, algum controle sobre a evolução do sistema é perdido, pois as novas versões dos componentes reusáveis não estão sob o controle da organização que os está utilizando.

ATIVIDADES DO PROCESSO

Atividades do processo

- Todos os modelos de processo contém quatro atividades base: especificação, desenvolvimento, validação e evolução.

- Estas são organizadas de forma diferente conforme o processo de desenvolvimento.

Atividades do processo

- No modelo em cascata são organizadas em sequência, enquanto no desenvolvimento incremental são intercaladas.

- A maneira como essas atividades serão feitas depende do tipo de software, das pessoas e das estruturas organizacionais envolvidas.

Atividades do processo

- Em extreme programming, por exemplo, as especificações estão escritas em cartões.

- Testes são executáveis e desenvolvidos antes do próprio programa.

- A evolução pode demandar reestruturação substancial do sistema ou fatoração.

Atividades do processo

- Processos reais de software são intercalados com sequências de atividades técnicas, de colaboração e de gerência, com o intuito de especificar, projetar, implementar e testar um sistema de software.

- Os desenvolvedores de software usam uma variedade de diferentes ferramentas de software em seu trabalho.

Atividades do processo

- As ferramentas são especialmente úteis para apoiar a edição de diferentes tipos de documentos e para gerenciar o imenso volume de informações detalhadas que é gerado em um projeto de grande porte.

ESPECIFICAÇÃO DE SOFTWARE

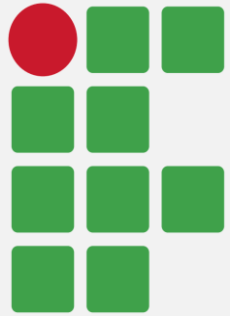
Especificação de software

- Especificação de software, ou engenharia de requisitos, é o processo de compreensão e definição dos serviços requisitados do sistema e identificação de restrições relativas à operação e ao desenvolvimento do sistema.

Especificação de software

- O processo de engenharia de requisitos tem como objetivo produzir um documento de requisitos acordados que especifica um sistema que satisfaz os requisitos dos stakeholders.

Especificação de software



- Levantamento e análise de requisitos: o que as partes interessadas do sistema exigem ou esperam do sistema?

- Especificação de requisitos: definição os requisitos em detalhes

- Validação de requisitos: verificar a validade dos requisitos

Especificação de software

- Requisitos são geralmente apresentados em dois níveis de detalhe
  - Os usuários finais e os clientes precisam de uma declaração de requisitos em alto nível.
  - Desenvolvedores de sistemas precisam de uma especificação mais detalhada do sistema.

Especificação de software

- Existem quatro atividades principais do processo de engenharia de requisitos: estudo de viabilidade, elicitação e análise de requisitos, especificação de requisitos e validação de requisitos.

Especificação de software

- 1. Estudo de viabilidade:
  - É feita uma estimativa acerca da possibilidade de se satisfazerem as necessidades do usuário identificado usando-se tecnologias atuais de software e hardware. O estudo considera se o sistema proposto será rentável a partir de um ponto de vista de negócio e se ele pode ser desenvolvido no âmbito das atuais restrições orçamentais. Um estudo de viabilidade deve ser relativamente barato e rápido. O resultado deve informar a decisão de avançar ou não, com uma análise mais detalhada.

Especificação de software

- 2. Elicitação e análise de requisitos
  - Esse é o processo de derivação dos requisitos do sistema por meio da observação dos sistemas existentes, além de discussões com os potenciais usuários e compradores, análise de tarefas, entre outras etapas.

  - Essa parte do processo pode envolver o desenvolvimento de um ou mais modelos de sistemas e protótipos, os quais nos ajudam a entender o sistema a ser especificado.

Especificação de software

- 3. Especificação de requisitos
  - É a atividade de traduzir as informações obtidas durante a atividade de análise em um documento que define um conjunto de requisitos.
  - Dois tipos de requisitos podem ser incluídos nesse documento.
  - Requisitos do usuário são declarações abstratas dos requisitos do sistema para o cliente e usuário final do sistema; requisitos de sistema são uma descrição mais detalhada da funcionalidade a ser provida.

Especificação de software

- 4. Validação de requisitos
  - Essa atividade verifica os requisitos quanto à realismo, consistência e completude.
  - Durante esse processo, os erros no documento de requisitos são inevitavelmente descobertos.
  - Em seguida, o documento deve ser modificado para correção desses problemas.

Especificação de software

- Naturalmente, as atividades no processo de requisitos não são feitas em apenas uma sequência.

- A análise de requisitos continua durante a definição e especificação, e novos requisitos emergem durante o processo.

Especificação de software

- Portanto, as atividades de análise, definição e especificação são intercaladas.

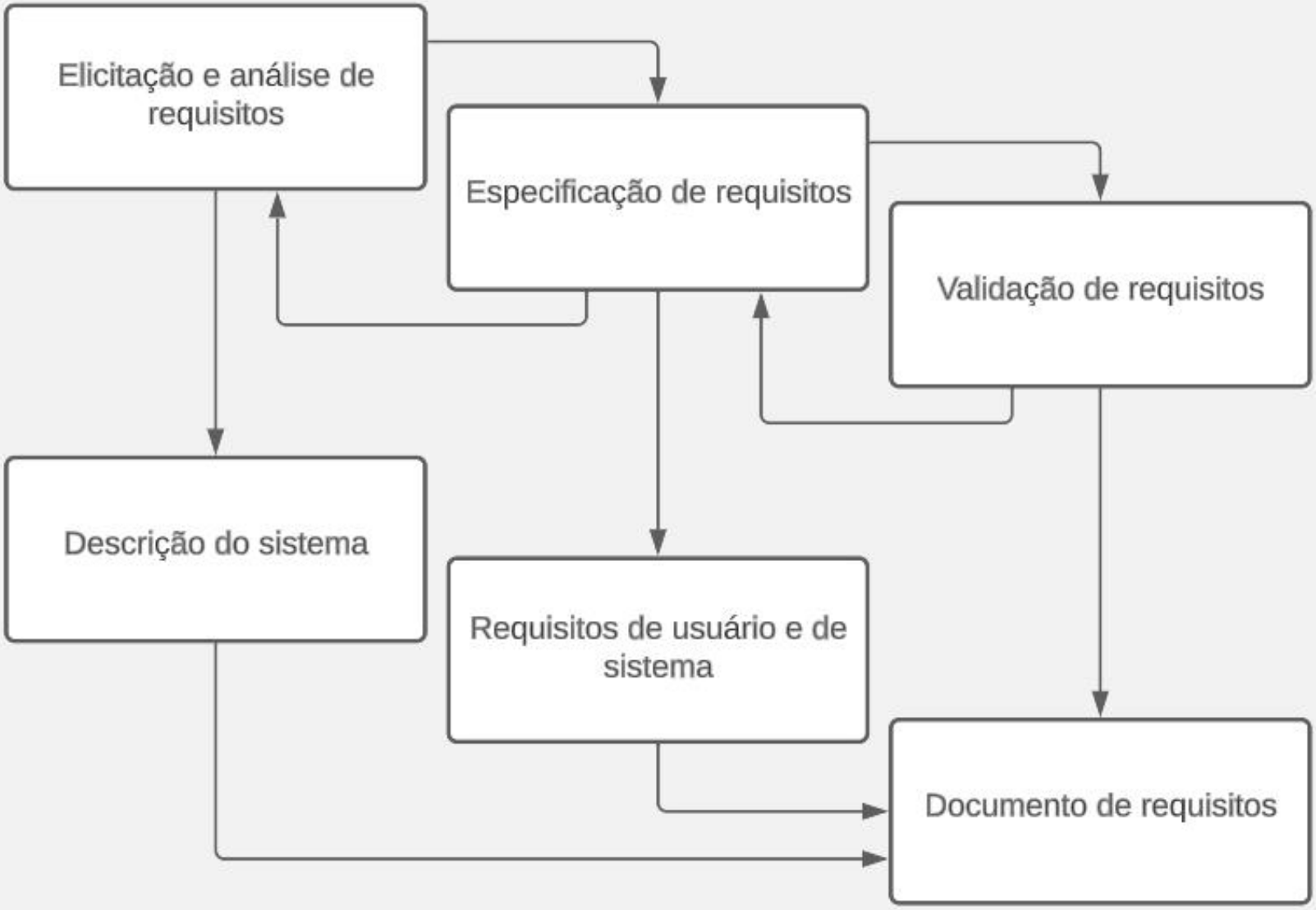
- Nos métodos ágeis, como *extreme programming*, requisitos são desenvolvidos de forma incremental, acordo com as prioridades do usuário, e a elicitação requisitos é feita pelos usuários que integram a equipe desenvolvimento.

Especificação de software

- Portanto, as atividades de análise, definição e especificação são intercaladas.

- Nos métodos ágeis, como *extreme programming*, requisitos são desenvolvidos de forma incremental, acordo com as prioridades do usuário, e a elicitação requisitos é feita pelos usuários que integram a equipe desenvolvimento.

Especificação de software



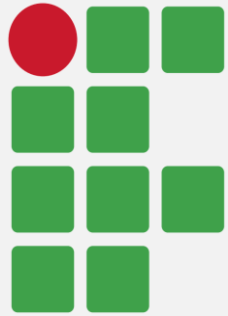
DESIGN E IMPLEMENTAÇÃO DE SOFTWARE

Design e implementação de software

- O estágio de implementação do desenvolvimento de software é o processo de conversão de uma especificação do sistema em um sistema executável.

- Sempre envolve processos de projeto e programação de software, mas, se for usada uma abordagem incremental para o desenvolvimento, também pode envolver o refinamento da especificação do software.

Design e implementação de software



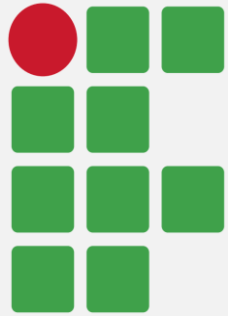
- Design de software
  - Projeto de uma estrutura de software que atenda às especificações.
- Implementação
  - Tradução desta estrutura em um programa executável.

- As atividades de design e implementação estão intimamente relacionadas e podem ser intercaladas.

Design e implementação de software

- Um projeto de software é uma descrição da estrutura do software a ser implementado, dos modelos e estruturas de dados usados pelo sistema, das interfaces entre os componentes do sistema e, as vezes, dos algoritmos usados.

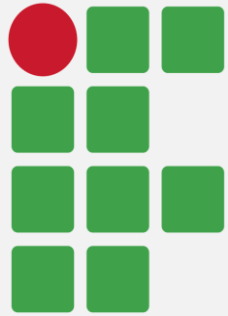
Design e implementação de software



- Os projetistas não chegam a um projeto final imediatamente, mas desenvolvem-no de forma iterativa.

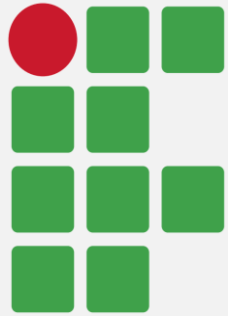
- Eles acrescentam formalidade e detalhes, enquanto desenvolvem seu projeto por meio de revisões constantes para correção de projetos anteriores.

Design e implementação de software



- Dentre as atividades do projeto de software, estão:
  - Projeto arquitetural, onde se identifica a estrutura geral do sistema, os principais componentes (subsistemas ou módulos), seus relacionamentos e como eles são distribuídos.
  - Design do banco de dados, onde se projeta as estruturas de dados do sistema e como elas devem ser representadas em um banco de dados.

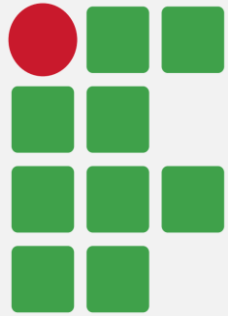
Design e implementação de software



- Design de interface, onde se define as interfaces entre os componentes do sistema.

- Seleção e design de componentes, onde se procura por componentes reutilizáveis. Se indisponível, você projeta como ele funcionará.

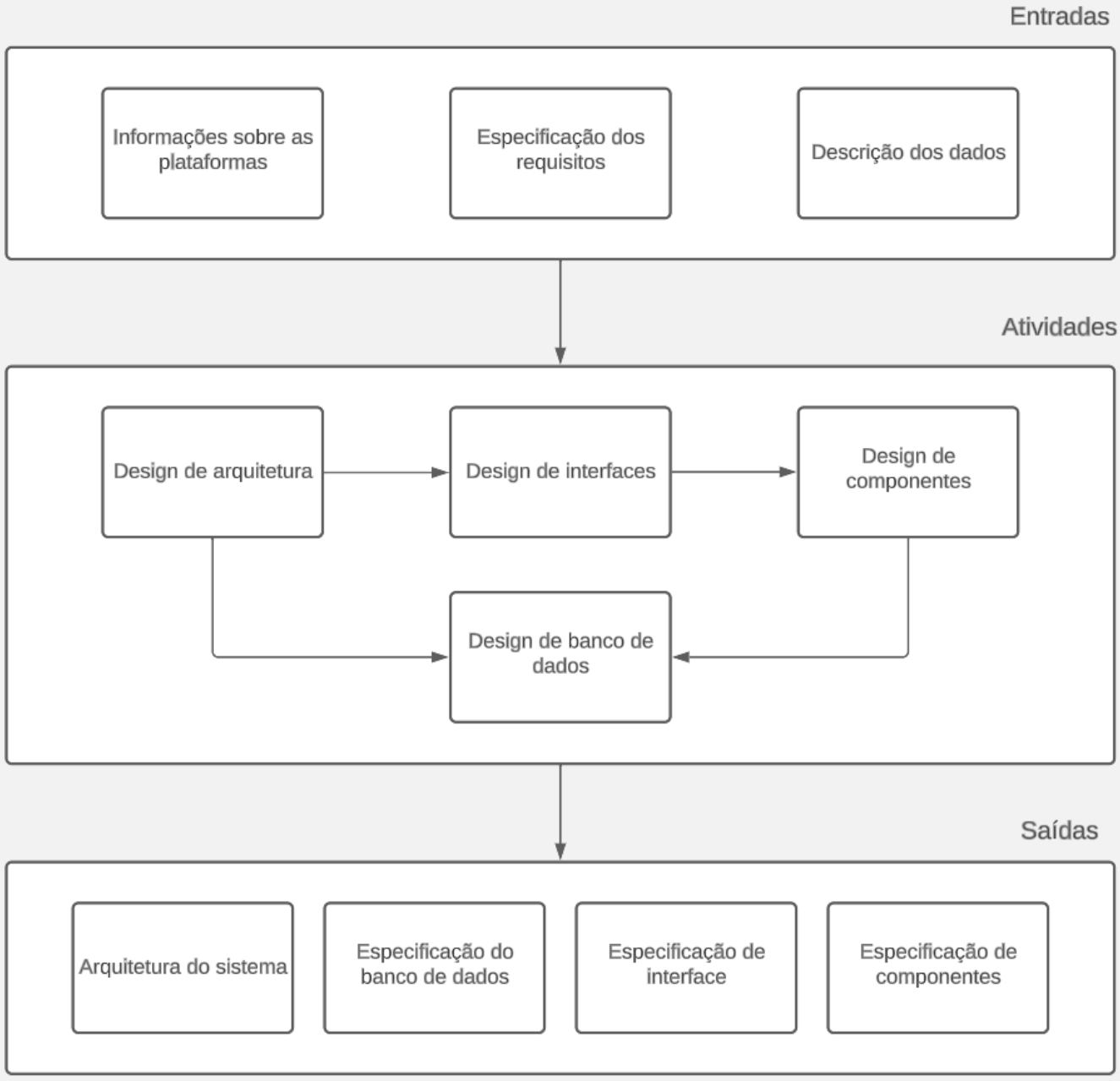
Design e implementação de software



- O software é implementado pelo desenvolvimento de um programa, ou programas, ou pela configuração de um sistema.

- A programação é uma atividade individual sem processo padrão (mas existem práticas que podem/devem ser seguidas)
    - A depuração é a atividade de encontrar falhas do programa e corrigi-las

Design e implementação de software



VALIDAÇÃO DE SOFTWARE

Validação de software

- A verificação e validação (V & V) tem como objetivo mostrar que um sistema está em conformidade com sua especificação e atende aos requisitos do cliente do sistema.

- Envolve processos de verificação, revisão e teste do sistema.

Validação de software

- O teste de sistema envolve a execução do sistema com casos de teste derivados da especificação dos dados reais a serem processados pelo sistema.

- O teste é a atividade de V&V mais comumente usada.

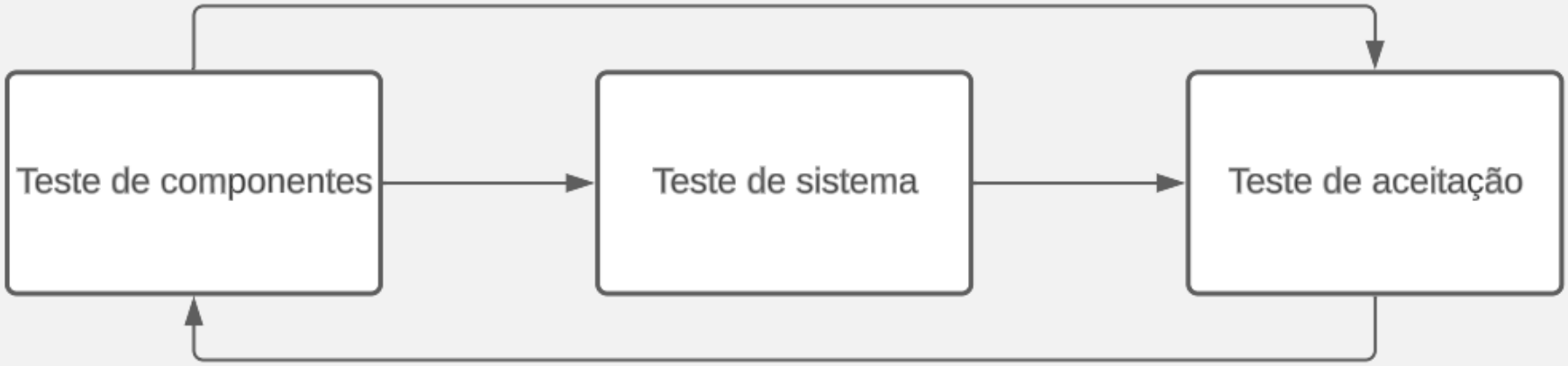
Validação de software

- Teste de componentes
  - Os componentes individuais são testados independentemente.
  - Os componentes podem ser funções, objetos ou agrupamentos coerentes dessas entidade.

Validação de software

- Teste de sistema
  - Teste do sistema como um todo. O teste de propriedades emergentes (propriedades que resultam da interação dos diversos componentes do sistema) é particularmente importante.
- Teste de cliente
  - Teste com dados do cliente para verificar se o sistema atende às necessidades do cliente.

Validação de software



EVOLUÇÃO DE SOFTWARE

Evolução de software

- O software é inerentemente flexível e pode mudar.

- À medida que os requisitos mudam devido às mudanças nas circunstâncias de negócios, o software que oferece suporte aos negócios também deve evoluir e mudar.

- Embora tenha havido uma demarcação entre desenvolvimento e evolução (manutenção), isso é cada vez mais irrelevante, pois cada vez menos sistemas são completamente novos.

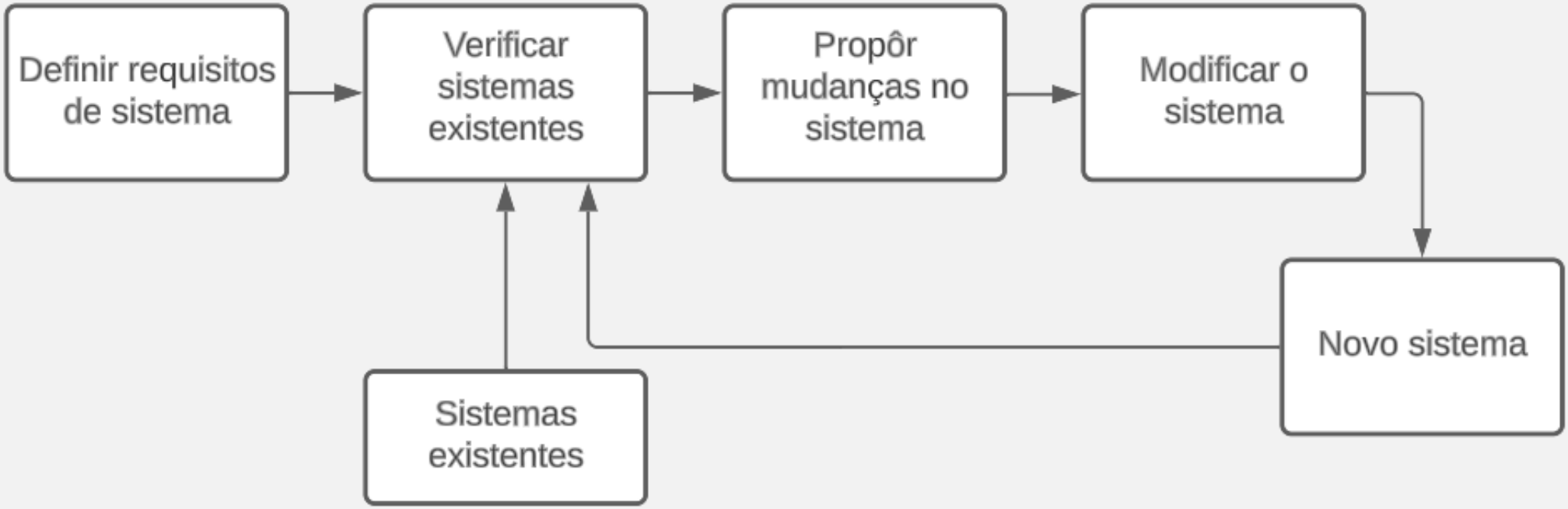
Evolução de software

- O software é inerentemente flexível e pode mudar.

- À medida que os requisitos mudam devido às mudanças nas circunstâncias de negócios, o software que oferece suporte aos negócios também deve evoluir e mudar.

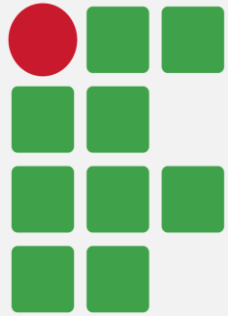
- Embora tenha havido uma demarcação entre desenvolvimento e evolução (manutenção), isso é cada vez mais irrelevante, pois cada vez menos sistemas são completamente novos.

Evolução de software



LIDANDO COM MUDANÇAS

Lidando com mudanças



- A mudança é inevitável em todos os grandes projetos de software.

- Mudanças de negócios levam a requisitos de sistema novos e alterados.

- Novas tecnologias abrem novas possibilidades para melhorar as implementações.

Lidando com mudanças

- Mudança leva a retrabalho, de modo que os custos de mudança incluem tanto o retrabalho (por exemplo, re-analisar requisitos), bem como os custos de implementação de novas funcionalidades.

Lidando com mudanças

- Para redução de custos de retrabalho pode-se incluir técnicas de antecipação de mudanças, dado que o processo de software inclua atividades que possam antecipar possíveis mudanças antes que um retrabalho significativo seja necessário.

- Por exemplo, um sistema de protótipo pode ser desenvolvido para mostrar alguns recursos-chave do sistema aos clientes

Lidando com mudanças

- Outra forma de redução de custos é o aumento da tolerância à mudança, em que o processo é projetado de forma que as mudanças possam ser acomodadas a um custo relativamente baixo.
  - Isso normalmente envolve alguma forma de desenvolvimento incremental. As mudanças propostas podem ser implementadas em incrementos que ainda não foram desenvolvidos

Lidando com mudanças

- As técnicas utilizadas para isso são:
  - Prototipagem do sistema, onde uma versão do sistema ou parte do sistema é desenvolvida rapidamente para verificar os requisitos do cliente e a viabilidade das decisões de design. Essa abordagem suporta a antecipação de mudanças.
  - Entrega incremental, em que os incrementos do sistema são entregues ao cliente para comentários e experimentação. Isso incrementa tanto a prevenção quanto a tolerância à mudança.

Lidando com mudanças

- Um protótipo é uma versão inicial de um sistema usado para demonstrar conceitos e experimentar opções de design.

- Um protótipo pode ser usado em:
  - Processo de engenharia de requisitos para ajudar na elicitação e validação de requisitos;
  - Em processos de design para explorar opções e desenvolver um design de UI.

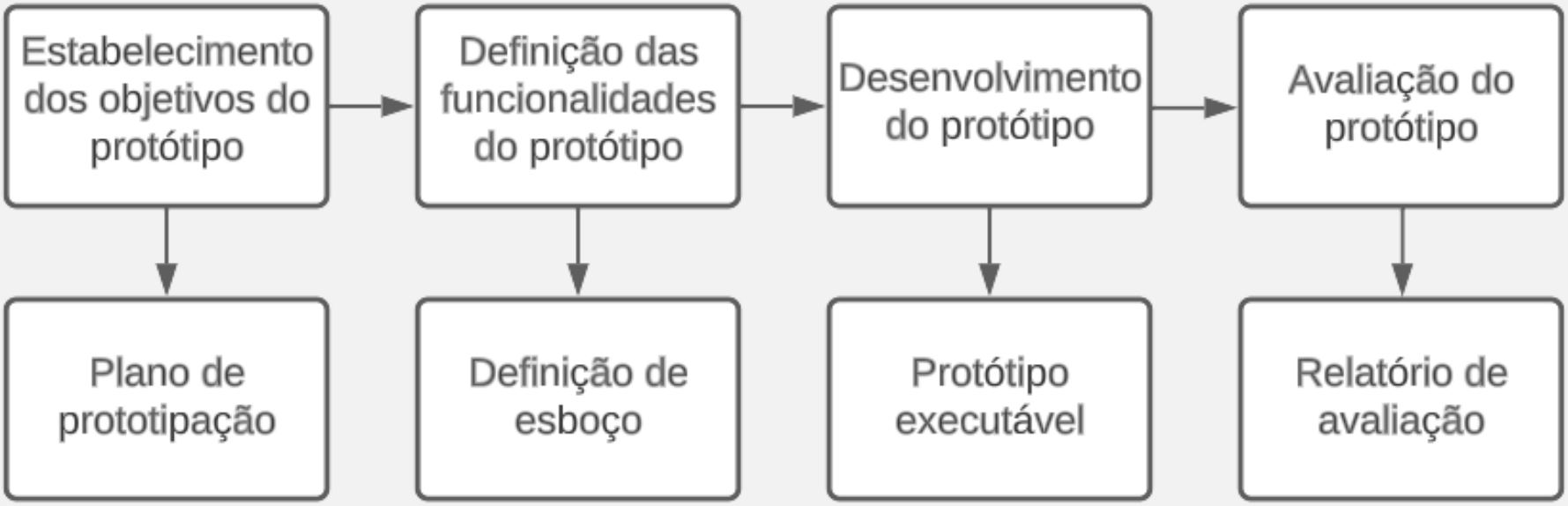
Lidando com mudanças

- Alguns dos benefícios da prototipagem incluem:
  - Melhor usabilidade do sistema.
  - Uma correspondência mais próxima às necessidades reais dos usuários.
  - Melhor qualidade de design.
  - Maior facilidade de manutenção.
  - Esforço de desenvolvimento reduzido.

Lidando com mudanças

- Alguns dos benefícios da prototipagem incluem:
  - Melhor usabilidade do sistema.
  - Uma correspondência mais próxima às necessidades reais dos usuários.
  - Melhor qualidade de design.
  - Maior facilidade de manutenção.
  - Esforço de desenvolvimento reduzido.

Lidando com mudanças



Lidando com mudanças

- Pode ser baseado em linguagens ou ferramentas de prototipagem rápida.

- O protótipo deve se concentrar em áreas do produto que não são bem compreendidas.
  - A verificação e recuperação de erros podem não estar incluídas no protótipo.
  - Concentre-se em requisitos funcionais em vez de não funcionais, como confiabilidade e segurança.

Lidando com mudanças

- Os protótipos devem ser descartados após o desenvolvimento, pois não são uma boa base para um sistema de produção.

- Pode ser impossível ajustar o sistema para atender aos requisitos não funcionais.

Lidando com mudanças

- Os protótipos normalmente não são documentados.

- A estrutura do protótipo geralmente é degradada por mudanças rápidas.

- O protótipo provavelmente não atenderá aos padrões normais de qualidade organizacional.

Lidando com mudanças

- Os protótipos normalmente não são documentados.

- A estrutura do protótipo geralmente é degradada por mudanças rápidas.

- O protótipo provavelmente não atenderá aos padrões normais de qualidade organizacional.

ENTREGA INCREMENTAL

Entrega incremental

- Em vez de entregar o sistema como uma entrega única, o desenvolvimento e a entrega são divididos em incrementos, com cada incremento entregando parte da funcionalidade necessária.

- Os requisitos do usuário são priorizados e os requisitos de prioridade mais alta são incluídos em incrementos iniciais.

Entrega incremental

- Depois que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, embora os requisitos para incrementos posteriores possam continuar a evoluir.

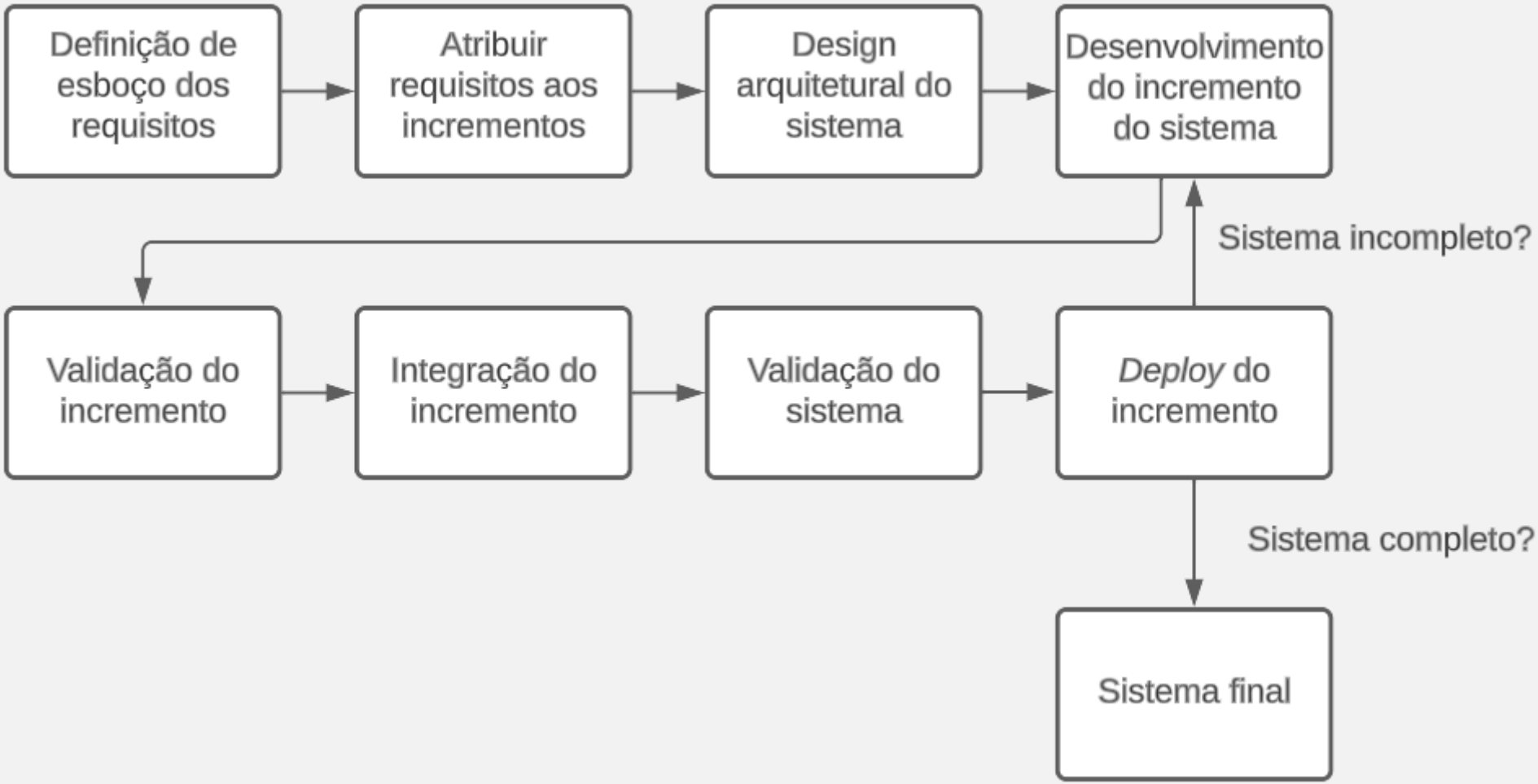
Entrega incremental

- Desenvolvimento incremental
  - Desenvolva o sistema em incrementos e avalie cada incremento antes de prosseguir para o desenvolvimento do próximo incremento.
  - Abordagem normalmente usada em métodos ágeis.
  - Avaliação feita por proxy do usuário / cliente.

Entrega incremental

- Entrega incremental
  - Funciona através de Implantação de um incremento para uso pelos usuários finais.
  - Avaliação mais realista sobre o uso prático do software.
  - Difícil de implementar para sistemas de substituição, pois os incrementos têm menos funcionalidade do que o sistema sendo substituído.

Entrega incremental



Entrega incremental

- Dentre as vantagens da entrega incremental
  - O valor do cliente pode ser entregue com cada incremento, de forma que a funcionalidade do sistema esteja disponível mais cedo.
  - Os incrementos iniciais atuam como um protótipo para ajudar a elicitar requisitos para incrementos posteriores.
  - Menor risco de falha geral do projeto.
  - Os serviços de sistema de prioridade mais alta tendem a receber mais testes.

Entrega incremental

- Dentre os possíveis problemas da entrega incremental, temos que a maioria dos sistemas requer um conjunto de recursos básicos que são usados por diferentes partes do sistema.
    - Como os requisitos não são definidos em detalhes até que um incremento seja implementado, pode ser difícil identificar recursos comuns que são necessários para todos os incrementos.

Entrega incremental

- Além disso, a essência dos processos iterativos é que a especificação é desenvolvida em conjunto com o software.

- No entanto, isso entra em conflito com o modelo de aquisição de muitas organizações, onde a especificação completa do sistema faz parte do contrato de desenvolvimento do sistema.

MELHORIA DE PROCESSOS

Melhoria de processos

- Muitas empresas de software têm se voltado para a melhoria de processos de software como forma de aumentar a qualidade de seu software, reduzindo custos ou acelerando seus processos de desenvolvimento.

- A melhoria do processo significa compreender os processos existentes e alterar esses processos para aumentar a qualidade do produto e / ou reduzir custos e tempo de desenvolvimento.

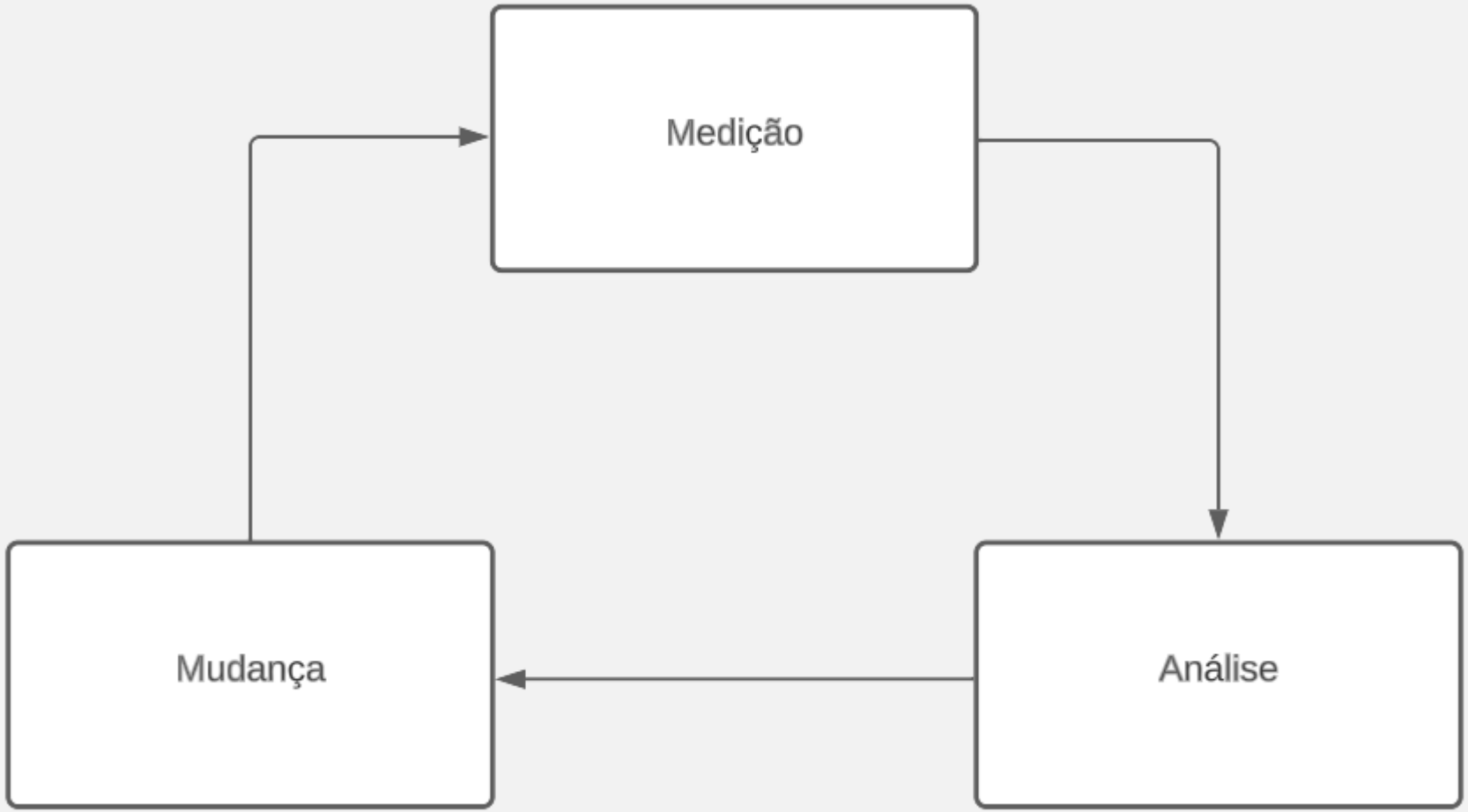
Melhoria de processos

- A abordagem da maturidade do processo, que se concentra na melhoria do gerenciamento de processos e projetos e na introdução de boas práticas de engenharia de software.
    - O nível de maturidade do processo reflete até que ponto as boas práticas técnicas e de gestão foram adotadas nos processos de desenvolvimento de software organizacional.

Melhoria de processos

- A abordagem ágil, que foca no desenvolvimento iterativo e na redução de overheads no processo de software.
    - As principais características dos métodos ágeis são a entrega rápida de funcionalidade e capacidade de resposta às mudanças nos requisitos do cliente.

Melhoria de processos



Melhoria de processos

- Medição de processo
  - Você mede um ou mais atributos do processo ou produto de software. Essas medições constituem uma linha de base que ajuda a decidir se as melhorias do processo foram eficazes.

Melhoria de processos

- Análise de processo
  - O processo atual é avaliado e as fraquezas e gargalos do processo são identificados. Modelos de processo (às vezes chamados de mapas de processo) que descrevem o processo podem ser desenvolvidos.

Melhoria de processos

- Mudança de processo
  - Mudanças no processo são propostas para resolver algumas das deficiências do processo identificadas. Eles são introduzidos e o ciclo é reiniciado para coletar dados sobre a eficácia das mudanças.

Melhoria de processos

- Sempre que possível, os dados quantitativos do processo devem ser coletados
    - No entanto, onde as organizações não têm padrões de processo claramente definidos, isso é muito difícil, pois você não sabe o que medir. Um processo pode ter que ser definido antes que qualquer medição seja possível.

Melhoria de processos

- As medições do processo devem ser usadas para avaliar as melhorias do processo
  - Mas isso não significa que as medições devem impulsionar as melhorias. O motivador da melhoria deve ser os objetivos organizacionais.

Melhoria de processos

- Tempo gasto para que as atividades do processo sejam concluídas
  - Por exemplo, tempo ou esforço do calendário para concluir uma atividade ou processo.

Melhoria de processos

- Tempo gasto para que as atividades do processo sejam concluídas
  - Por exemplo, tempo ou esforço do calendário para concluir uma atividade ou processo.

Melhoria de processos

- Recursos necessários para processos ou atividades
  - Por exemplo, esforço total em pessoa-dia.

- Número de ocorrências de um determinado evento
  - Por exemplo, número de defeitos descobertos.

PONTOS-CHAVE

Pontos-chaves

- Os processos de software são as atividades envolvidas na produção de um sistema de software.

- Os modelos de processo de software são representações abstratas desses processos.

Pontos-chaves

- Os modelos gerais de processos descrevem a organização dos processos de software.
    - Exemplos desses modelos gerais incluem o modelo "cascata", desenvolvimento incremental e desenvolvimento orientado para a reutilização.

Pontos-chaves

- Engenharia de requisitos é o processo de desenvolvimento de uma especificação de software.

- Os processos de design e implementação estão preocupados em transformar uma especificação de requisitos em um sistema de software executável.

Pontos-chaves

- A validação de software é o processo de verificação de que o sistema está de acordo com sua especificação e atende às reais necessidades dos usuários do sistema.

- A evolução do software ocorre quando você altera os sistemas de software existentes para atender a novos requisitos. O software deve evoluir para permanecer útil.

Pontos-chaves

- Os processos devem incluir atividades como prototipagem e entrega incremental para lidar com a mudança.

- Os processos podem ser estruturados para desenvolvimento e entrega iterativos, de modo que as alterações possam ser feitas sem interromper o sistema como um todo.

Pontos-chaves

- As principais abordagens para a melhoria de processos são abordagens ágeis, voltadas para reduzir sobrecargas de processos, e abordagens baseadas em maturidade baseadas em um melhor gerenciamento de processos e no uso de boas práticas de engenharia de software.

Referências

- PRESSMAN, R. S Engenharia de Software - Uma abordagem Profissional - 8 ed. Porto Alegre, RS: Bookman/Amgh Editora, 968 p. ISBN: 9788580555332, 2016.

- SOMMERVILLE, Ian. Engenharia de software 9 ed. São Paulo, SP: Pearson Prentice Hall, 568p. ISBN: 9788579361081, 2011.