Processos de Desenvolvimento de Software

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Marco Tulio Valente
Engenharia de Software Moderna

Conceitos

PROCESSO

Conjunto de **passos** parcialmente **ordenados** para atingir uma **meta**.

PROCESSO DE SOFTWARE

Conjunto de **atividades**, **métodos**, **práticas** e **transformações** que **pessoas utilizam** para **desenvolver** ou dar **manutenção** em software ou em seus produtos associados.

Modelos de Processo de Software

- Originalmente, modelos de processo prescritivos foram **propostos para trazer ordem ao caos existente na área de desenvolvimento** de software.
- Denominam-se “**prescritivos**” porque prescrevem um conjunto de elementos de processo – atividades metodológicas, ações de engenharia de software, tarefas, produtos de trabalho, garantia da qualidade e mecanismos de controle de mudanças.
- Cada modelo de processo também prescreve um **fluxo de processo** - a forma pela qual os elementos do processo estão inter-relacionados.

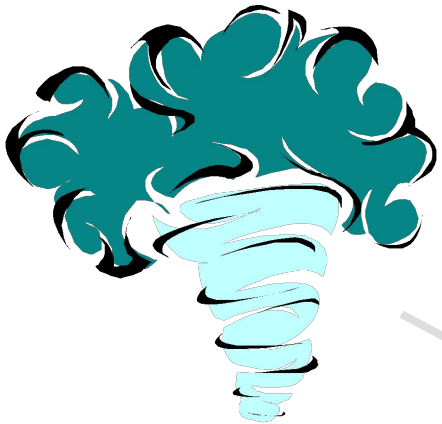
Modelos de Processo de Software

- **Codifica-Remenda:** modelo de processo sem um plano.
- **Modelos em Cascata:** primeiros modelos dirigidos a planos.
- **Desenvolvimento Incremental:** Especificação, desenvolvimento e validação são intercaladas. Pode ser dirigido a planos ou ágil.
- **Engenharia de software orientada a reuso:** pode ser dirigido a planos ou ágil. Inclui técnicas da 3ª geração (ex: RAD) e técnicas da 4ª geração (ex: RUP).
- **Metodologias Ágeis:** imprevisibilidade e foco em mudanças.

Processos de Software sem Plano ou Modelo

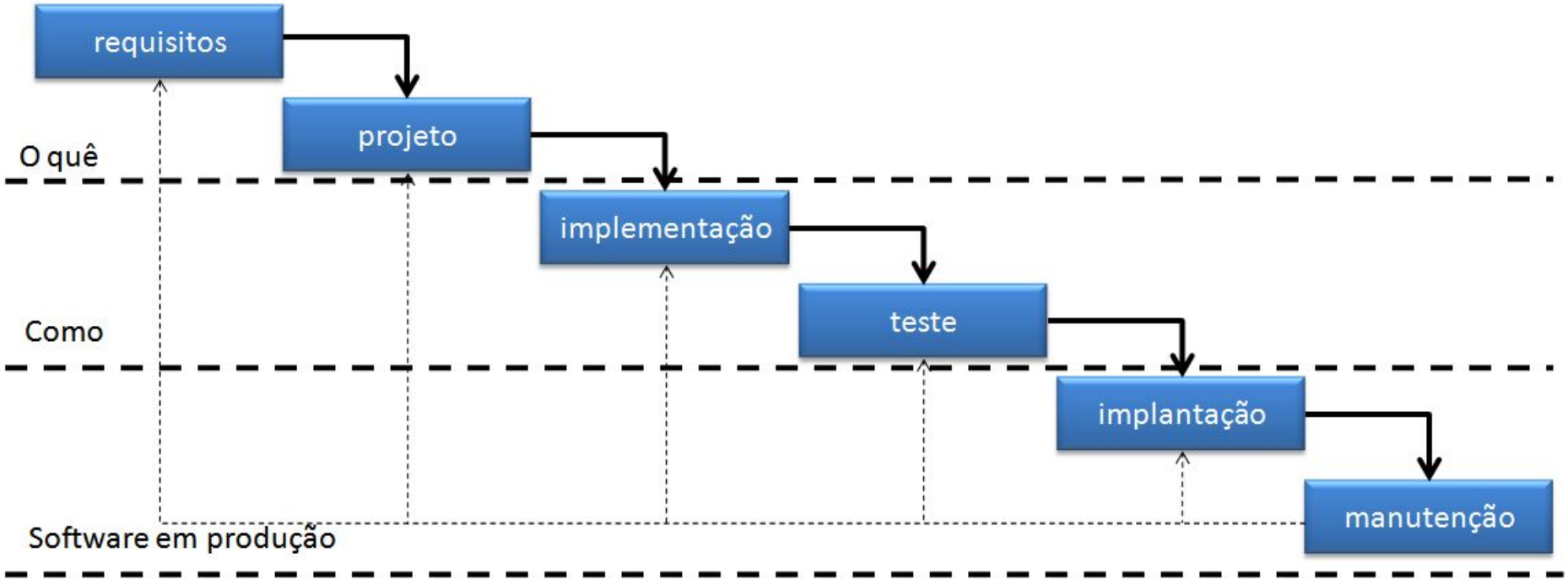
Codifica-Remenda

Especificação (???)



Produto

Modelo Cascata - Ciclo de Vida Clássico



Problemas do Modelo Cascata

- Divisão inflexível do projeto em estágios distintos torna difícil responder às mudanças nos requisitos do cliente
- Uma fase precisa ser completada antes de se mover para a próxima fase.
- Por isso, esse modelo só é apropriado quando os requisitos são bem entendidos e as mudanças durante o processo de projeto serão limitadas.
- Poucos sistemas de negócio possuem requisitos estáveis.

Lidando com Mudanças

- As mudanças são inevitáveis em todos grandes projetos de software.
  - Mudanças no negócio levam a novos e diferentes requisitos de sistema.
  - Novas tecnologias abrem novas possibilidades para melhorar implementações.
  - Mudanças de plataforma requerem mudanças na aplicação.
- As mudanças geram retrabalho, o que faz com que o custo das mudanças inclua o retrabalho (reanálise dos requisitos) assim como o custo de implementação de novas funções.

Lidando com Mudanças

- **Prevenção de mudanças**
  - processo de software inclui atividades que podem antecipar possíveis mudanças antes que o retrabalho se torne necessário. Por exemplo, a **prototipação**.
- **Tolerância a mudanças**
  - O processo é desenvolvido para que mudanças possam ser acomodadas a um custo relativamente baixo.
  - Geralmente envolve alguma forma de desenvolvimento incremental: as mudanças propostas podem ser implementadas em incrementos que ainda não foram desenvolvidos.

Prototipação de Software

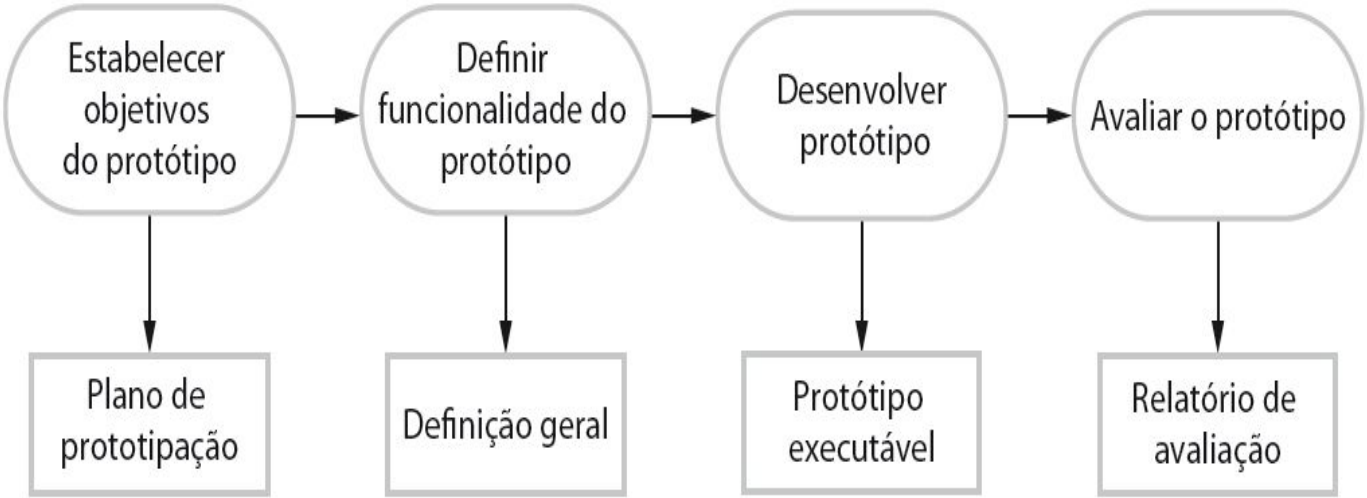
- Um protótipo é uma versão inicial de um sistema usada para demonstrar conceitos e testar opções de projeto.

- Um protótipo pode ser usado:
  - No processo de engenharia de requisitos para ajudar na elicitação e validação de requisitos;
  - Nos processos de projeto para explorar opções e desenvolver um projeto de interface de usuário;
  - No processo de testes para executar testes fim-a-fim.

Prototipação de Software

- Processo de Prototipação Rápida

O processo de desenvolvimento do protótipo

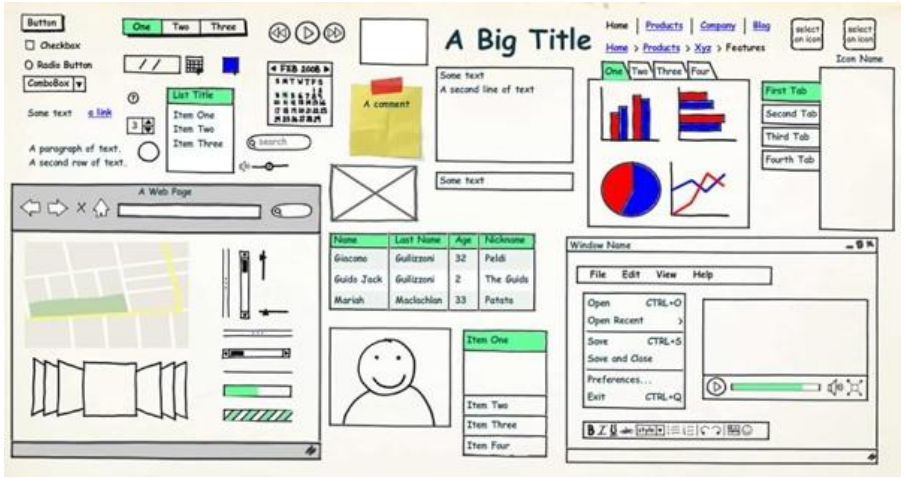


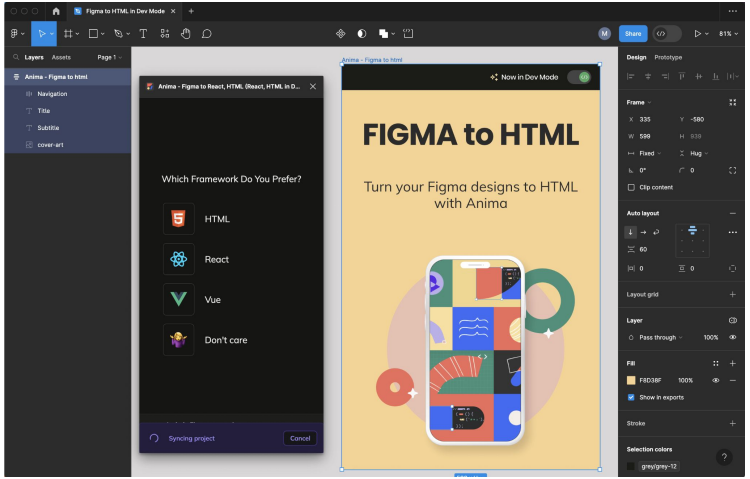
Desenvolvimento de protótipos

- Pode ser baseado em linguagens ou ferramentas de prototipagem rápida.
- Perigo: pode deixar a funcionalidade de fora do teste. A checagem de erros e recuperação podem não estar incluídas no protótipo.
- A prototipação deve focar em áreas do produto que não são bem entendidas. Foco em requisitos funcionais ao invés de não funcionais (ex: segurança).
- Muitos defendem que os protótipos devem ser descartados depois do desenvolvimento

Prototipação de Software

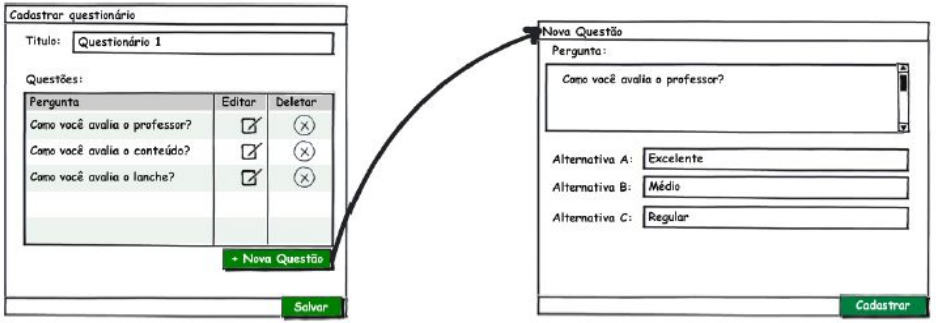
- Ferramentas automatizadas para prototipação:
  - Axure, Justinmind, PencilProject, Form, Fluid, Balsamiq, Mockup Builder, Cacoo, Mockabilly, Figma



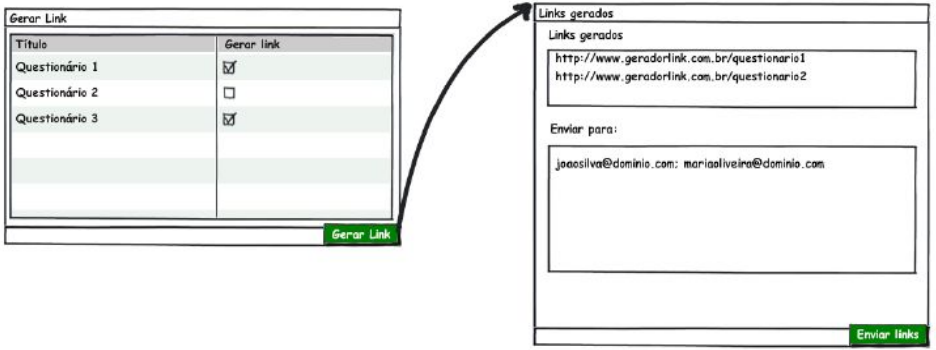


Prototipação de Software

Cadastrar Questionário



Gerar Link



Prototipação: Análise

- A prototipação deve ser usada quando há um alto grau de incerteza dos requisitos ou quando é necessário um rápido feedback dos usuários.
- Com o uso de protótipos é possível apresentar opções de sistema aos usuários e melhorar o entendimento.
- Permite também a detecção preventiva de problemas, reduz custos e melhora a qualidade do produto final.
- A prototipação é amplamente utilizada em metodologias ágeis (**Scrum, XP, Kanban, Lean**) e também em modelos de processo incremental (ex: **modelo espiral**).

Prototipação: Análise

- **Vantagens de prototipação**
  - Melhoria do uso do software.
  - Maior proximidade com as necessidades do usuário.
  - Melhorias na qualidade do projeto.
  - Maior manutenibilidade.
  - Reduzir esforços de desenvolvimento.

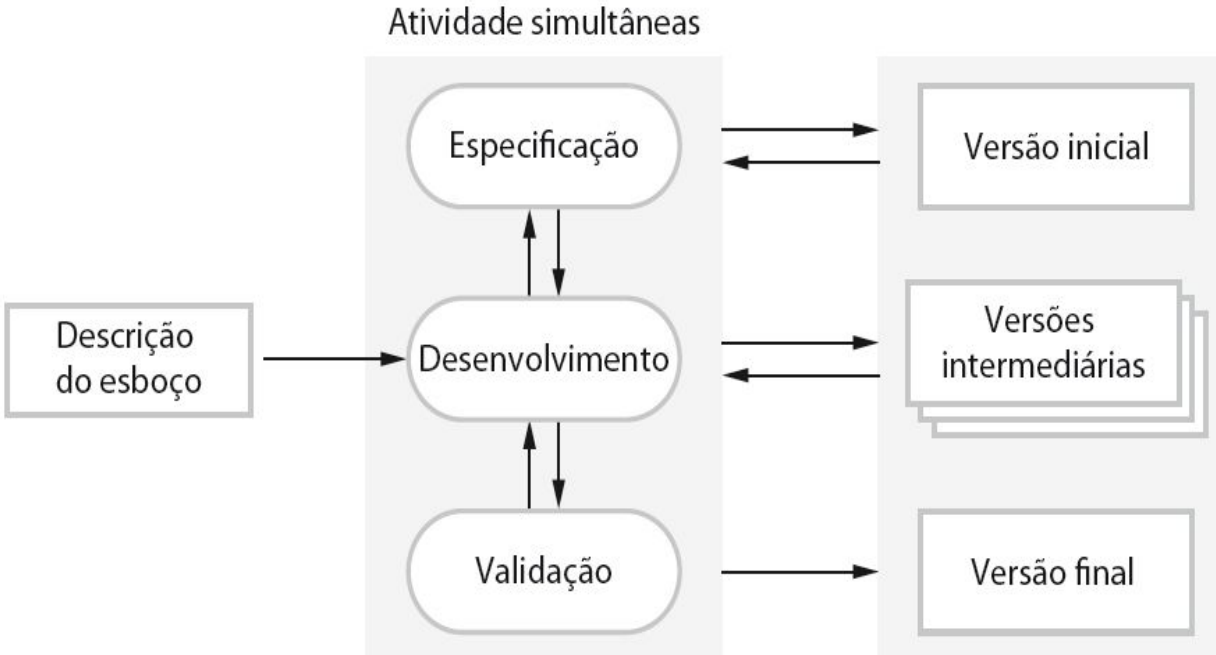
Desenvolvimento e Entrega Incremental

- Ao invés de entregar o sistema em uma única entrega, o desenvolvimento e a entrega são distribuídos em incrementos, nos quais cada incremento entrega parte da funcionalidade necessária.
- Os requisitos do usuário são priorizados e os requisitos de mais alta prioridade são incluídos nos primeiros incrementos.
- Assim que o desenvolvimento de um incremento é iniciado os requisitos são congelados, mas os requisitos dos incrementos posteriores podem continuar a evoluir.

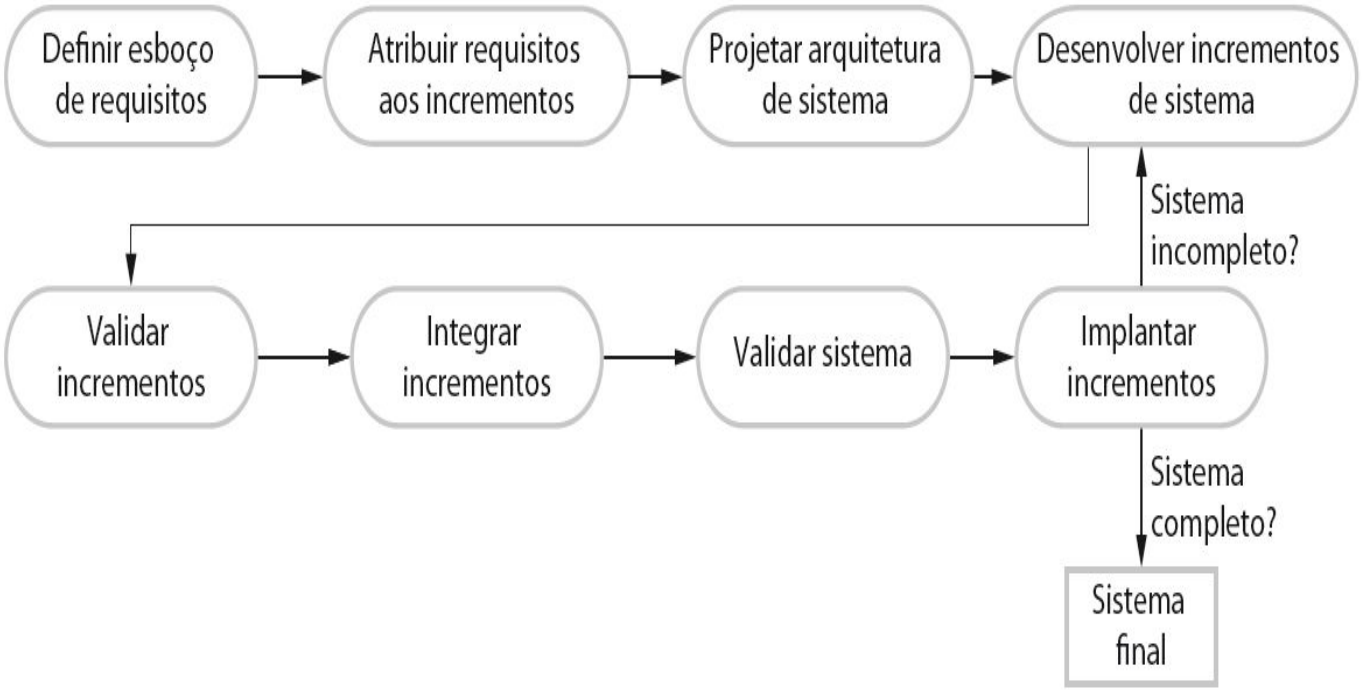
Desenvolvimento e Entrega Incremental

- Desenvolvimento incremental
  - Desenvolve o sistema em incrementos e avalia cada incremento antes de proceder com o desenvolvimento do próximo incremento;
  - Abordagem normalmente usada em métodos ágeis;
  - Avaliação feita por representantes do usuário/cliente.
- Entrega incremental
  - Implanta um incremento para uso do usuário-final;
  - Avaliação mais realística sobre o uso prático do software;
  - Difícil de implementar para sistemas substitutos devido aos incrementos possuírem menos funções do que o sistema que está sendo substituído.

Modelo Incremental



Modelo Incremental: Modelo Evolucionário



Modelo Evolucionário

- Aplicado quando um cliente tiver uma necessidade razoável mas não possui os detalhes.
- Um protótipo é desenvolvido quando requisitos simples foram aprovados e, posteriormente, outros podem ser levantados.
- A iteração ocorre à medida que o protótipo é ajustado para satisfazer às necessidades do cliente e o feedback é usado para refinar os requisitos.
- Não se deve sacrificar qualidade em nome de um protótipo defeituoso, mas sim aprovado.
- O cliente deve estar ciente de que aquele **não é um modelo definitivo do software.**

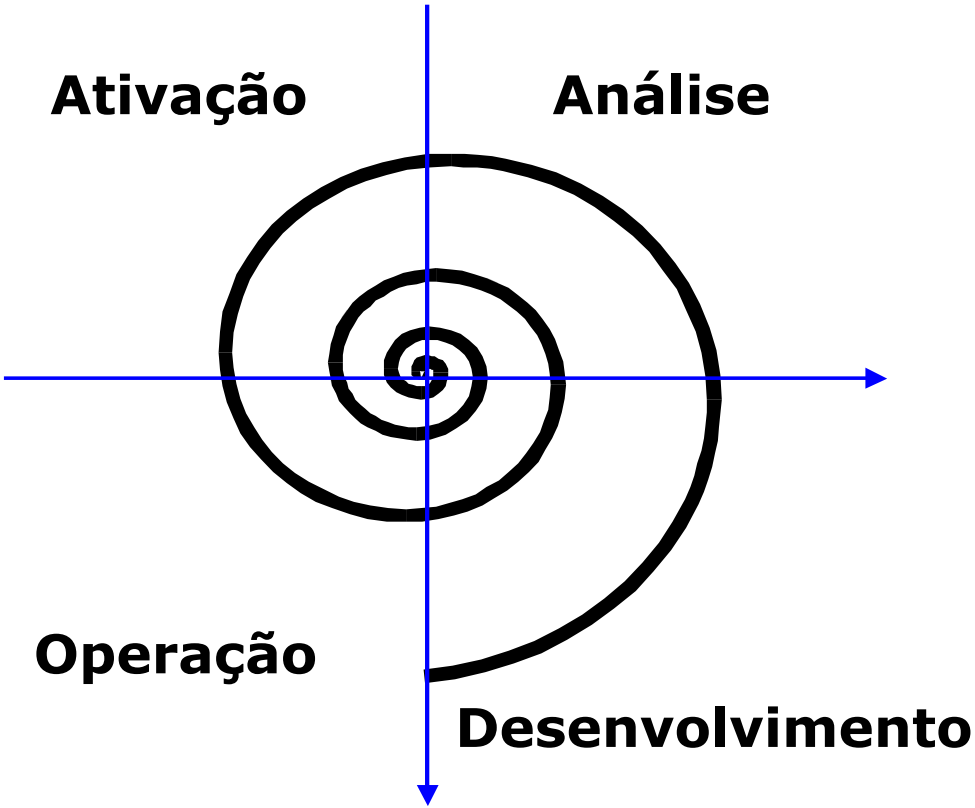
Vantagens da entrega incremental

- Os valores podem ser entregues ao cliente junto com cada incremento, e funções do sistema ficam disponíveis mais rapidamente.
- Primeiros incrementos agem como protótipos para ajudar a deduzir requisitos para incrementos posteriores.
- Menor risco de falha geral do projeto.
- Os serviço mais prioritários do sistema tendem a serem mais testados.

Problemas da entrega incremental

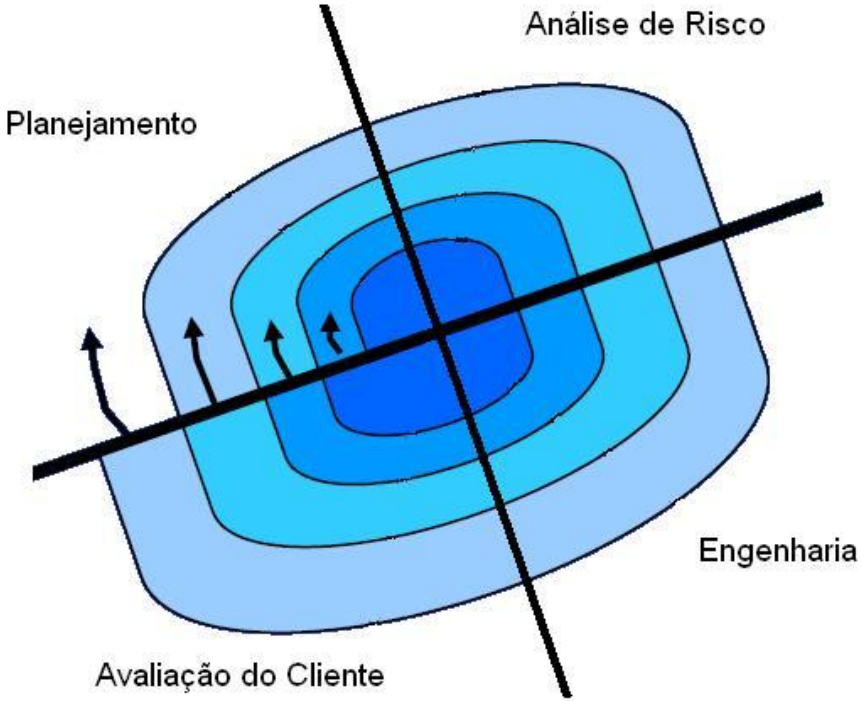
- A maioria dos sistemas requer um conjunto de funções básicas que são usadas por diferentes partes do sistema.
    - Como os requisitos não são definidos em detalhes até que um incremento seja implementado, pode ser difícil identificar funções comuns que são necessárias a todos os incrementos.
- A essência dos processos iterativos é que a especificação seja desenvolvida em conjunto com o software.
    - No entanto, essa pode entrar em conflito com o modelo de aquisição de muitas organizações, nos quais a especificação completa do sistema é parte do contrato de desenvolvimento do sistema.

Modelo em Espiral



- Combina a natureza iterativa da prototipagem com os aspectos controlados do Modelo Cascata
- Software evolui através de vários ciclos completos de especificação, projeto e desenvolvimento

Modelo em Espiral



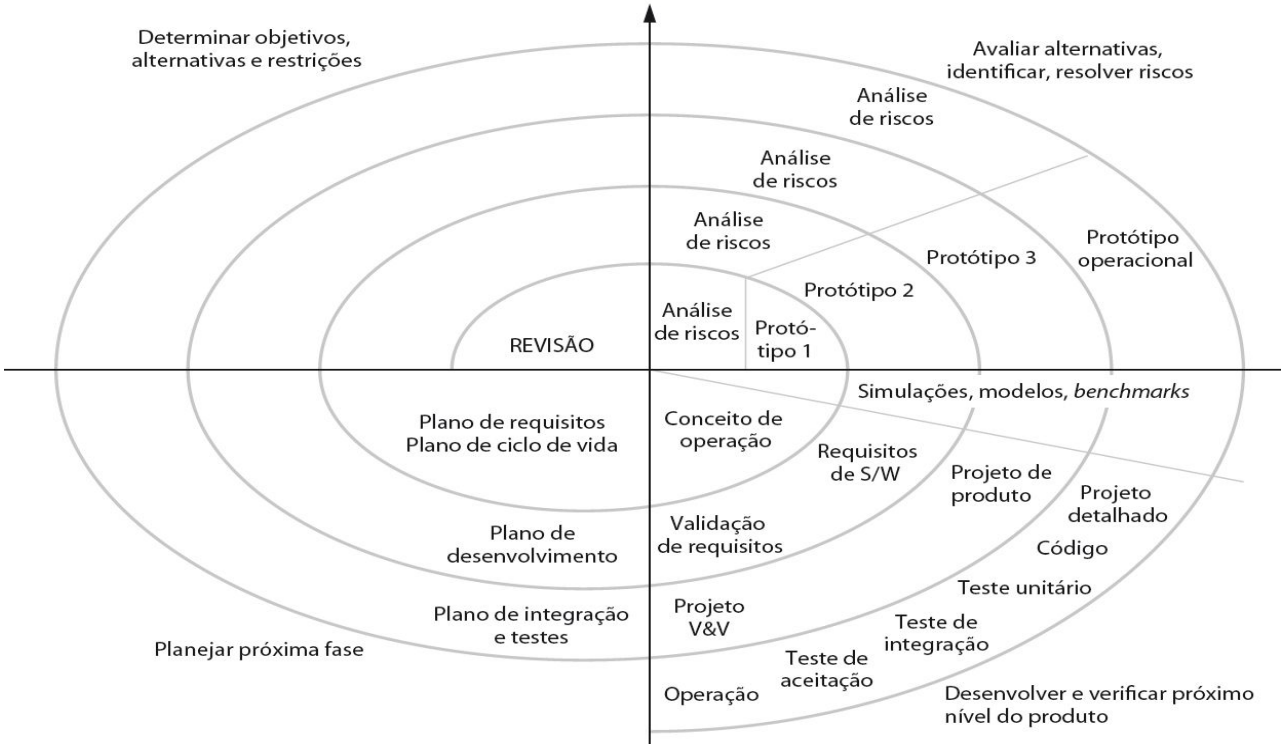
- Várias versões evolucionárias, que podem começar com um protótipo ou rascunho em papel e evoluir para um software mais completo.
- O custo e o cronograma são ajustados com base no feedback derivado do cliente após a entrega.

Pode ser difícil gerar contratos!

Modelo Espiral de Boehm

- O processo é representado como uma espiral ao invés de uma sequência de atividades com retornos.
- Cada loop na espiral representa uma fase do processo.
- Não existem fases fixas como especificação ou projeto – os loops na espiral são escolhidos de acordo com a necessidade.
- Os riscos são avaliados explicitamente e resolvidos no decorrer do processo.

Modelo Espiral de Boehm



Setores do Modelo Espiral

- Definição de objetivos
  - São identificados os objetivos específicos para cada fase.

- Avaliação e redução de riscos
  - Os riscos são avaliados e atividades executadas para reduzir os principais riscos.

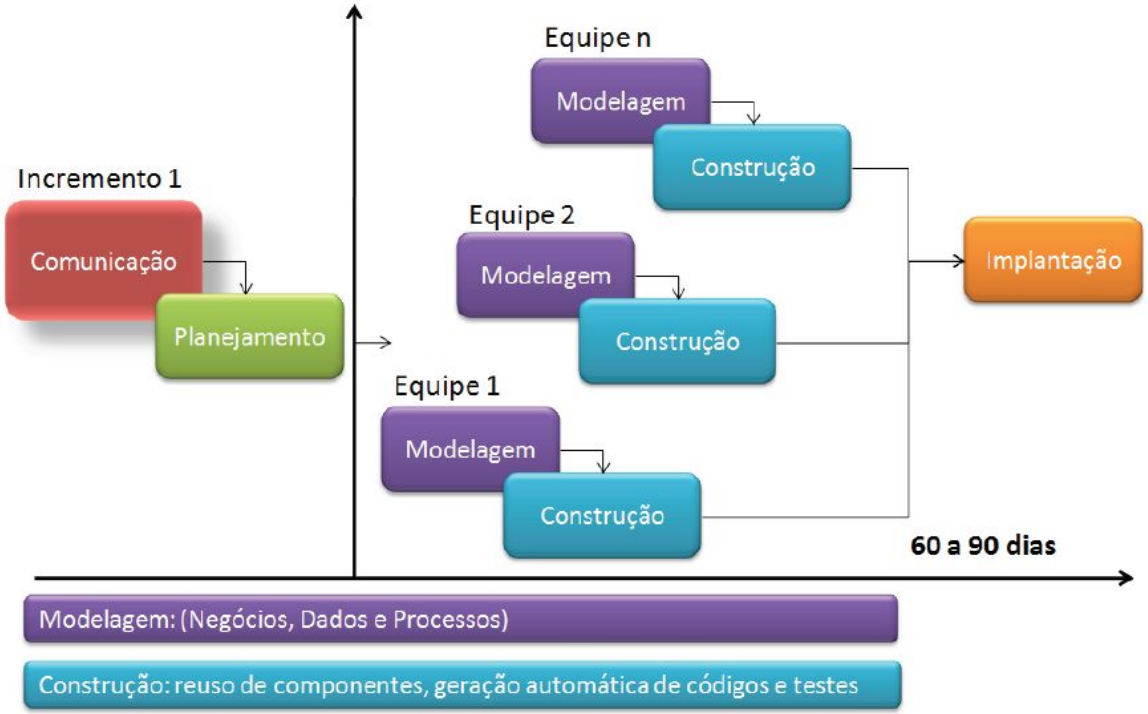
- Desenvolvimento e validação
  - Um modelo de desenvolvimento para o sistema é escolhido, pode ser qualquer um dos modelos genéricos.

- Planejamento
  - O projeto é revisto e a próxima fase da espiral é planejada.

RAD: Rapid Application Development

- **Modelo de desenvolvimento rápido de aplicação**
- Modelo incremental com foco no **desenvolvimento curto e construção baseada em componentes**.
- Adaptação em alta velocidade do modelo em cascata
- A modelagem abrange três fases: **modelagem de negócios, modelagem dos dados e modelagem dos processos**.
- A construção enfatiza o uso de componentes e **geração automática de código**
- A implantação estabelece base para iterações subsequentes se necessário.

RAD: Rapid Application Development



Modelos: Técnicas de Quarta Geração

- Concentra-se na capacidade de s**e especificar software a uma máquina em um nível que esteja próximo à linguagem natural** ou de se usar uma notação que comunique uma função significativa.
- O **código fonte é gerado automaticamente** a partir destas especificações.
- Caracteriza-se pelo **suporte automatizado à especificação de requisitos**.
- Atua bem quando existe necessidade de flexibilidade, tempo curto e permite documentação em todas as fases de forma automática

Modelo RUP: Rational Unified Process

- Processo proprietário criado pela **Rational Software Corporation**, adquirida pela IBM.
- RUP usa a **abordagem da orientação a objetos** em sua concepção e é projetado e documentado utilizando a notação UML para ilustrar os processos.
- É um processo considerado pesado e preferencialmente **aplicável** a grandes equipes de desenvolvimento e a **grandes projetos**.
- Porém o fato do RUP ser **amplamente customizável** torna possível que seja adaptado para projetos de qualquer escala.

Modelo RUP: Rational Unified Process

- É um processo genérico moderno, derivado do trabalho em UML e processos associados.

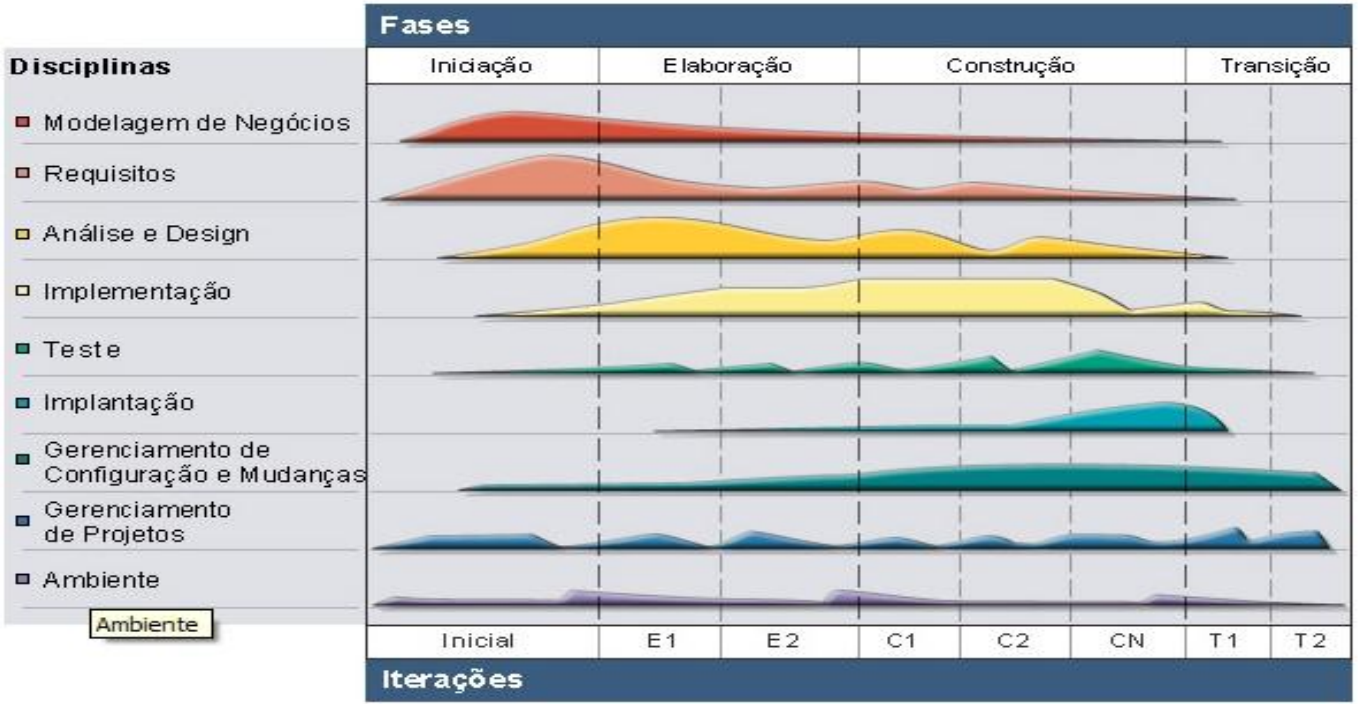
- Reúne aspectos dos 3 modelos genéricos discutidos previamente.

- Geralmente descrito por 3 perspectivas:
  - Uma perspectiva dinâmica que mostra fases no tempo;
  - Uma perspectiva estática que mostra atividades do processo;
  - Uma perspectiva prática que sugere boas práticas.

Modelo RUP: Rational Unified Process

- RUP divide o projeto em 4 fases distintas:
  a. Concepção: ênfase no escopo do sistema;
  b. Elaboração: ênfase na arquitetura;
  c. Construção: ênfase no desenvolvimento;
  d. Transição: ênfase na implantação.
- As fases são compostas de iterações que possuem prazo definido
- Todas as fases geram artefatos, que são utilizados nas próximas fases e documentam o projeto, além de permitir melhor acompanhamento.

Modelo RUP: Rational Unified Process



- Eixo horizontal – tempo e aspectos do ciclo de vida à medida que se desenvolve
- Eixo vertical – Disciplinas que agrupam logicamente as atividades

Fases do RUP

- Concepção
  - Estabelece o business case para o sistema.
- Elaboração
  - Desenvolve um entendimento da extensão do problema e da arquitetura do sistema.
- Construção
  - Projeta o sistema, programa e testa o sistema.
- Transição
  - Implanta o sistema no seu ambiente de operação.

Interação do RUP

- Iteração Intra-fase
  - Cada fase é iterativa aos resultados desenvolvidos incrementalmente

- Iteração Inter-fase
  - Como mostrado pelo loop no modelo RUP, o conjunto todo de fases pode ser executado incrementalmente.

Workflows estáticos do RUP

- Modelagem de Negócios
  - Os processos de negócios são modelados.
- Requisitos
  - Identificação de atores do sistema e elaboração dos casos de uso.
- Análise e Projeto
  - É criado um modelo de projeto, contendo modelo da arquitetura, de componentes, de objetos e de sequencia.
- Implementação
  - Componentes implementados e estruturados. Pode incluir geração automática de código.

Workflows estáticos do RUP

- Teste
  - O teste é um processo iterativo, realizado junto com a implementação e segue nas fases seguintes.
- Implantação
  - Uma release é gerada, distribuída aos clientes e implantada em cada um.
- Gerenciamento de configuração e mudanças
  - Gerencia o processo de mudanças e versões do sistema.
- Gerenciamento de projeto
  - Gerencia o desenvolvimento do sistema.
- Meio ambiente
  - Disponibilização de ferramentas para a equipe de desenvolvimento.

Boas práticas do RUP

- Desenvolver software iterativamente
  - Planejar incrementos baseando-se nas prioridades do cliente e entregar as de prioridade mais alta primeiro.

- Gerenciar os requisitos
  - Documentar explicitamente os requisitos do cliente e manter registros de mudanças desses requisitos.

- Usar arquiteturas baseadas em componentes
  - Organizar a arquitetura do sistema como um conjunto de componentes reusáveis.

Boas práticas do RUP

- Modelar o software visualmente
  - Use modelos de gráficos UML para representar visões dinâmicas e estáticas do software.

- Verificar a qualidade do software
  - Garanta que o software atenda aos padrões de qualidade organizacional.

- Controlar as mudanças do software
  - Gerencie as mudanças no software usando um sistema de gerenciamento de mudanças e ferramentas de gerenciamento de configuração.

Processos de Desenvolvimento de Software

Prof. Pedro Henrique Dias Valle

Adaptado de:
Prof. Marco Tulio Valente
Engenharia de Software Moderna