Engenharia de Software

Aula3: CICLO DE VIDA E MODELOS DE DESENVOLVIMENTO DE SOFTWARE

Cascata, Espiral, Prototipação, Modelo em V, RAD e RUP
- Ciclo de Vida Evolutivo, Iterativo e Incremental.

Dra. Ana Patrícia F. Magalhães Mascarenhas

anapatriciamagalhaes@gmail.com

PLANO DE AULA

- Objetivo
  - Definir processo e modelo de processo de desenvolvimento de software
  - Descrever os modelos de processo Cascata, Evolucionário e Incremental
  - Identificar e Utilizar modelos de processos de software.
  - Apresentar tecnologias para utilização de processos
- Bibliografia básica
  - SOMMERVILLE, I. Engenharia de Software. 9a edição. **Capítulo 2**. Pearson Addison Wesley. 2011.
  - PRESSMAN, R. , MAXIM, B. Engenharia de Software, Capítulo 02, 8th edição.
- Complementar
  - http://engenhariadesoftwareuesb.blogspot.com/2012/12/blog-post.html

Antes de começar...

- Vamos voltar ao problema do sistema de matrícula tratado na aula anterior...

Você é gerente de projetos de uma fábrica de software e foi alocado para gerenciar o projeto de desenvolvimento de um **sistema para informatização da matrícula da rede estadual de ensino**.

Atualmente a matrícula dos alunos da rede pública é bastante complicada, pois é realizada presencialmente em cada escola.

Essa prática gera muitas filas e problemas quando os alunos querem se transferir de uma escola para outra.

Assim, a secretaria de educação (SEC) contratou seus serviços para informatizar todo esse processo através da construção de um software para atender a essa necessidade

A solução proposta

- Construir um app para realização da matrícula
  - Alunos / responsáveis realizam pre matricula em período determinado
    - Re matricula
    - Transferência
  - Sistema distribui vagas conforme critérios
    - Alunos da casa tem garantia de vaga
    - Transferências respeitam critérios (localização, idade...)
  - Sistema envia comprovante de matrícula para alunos
- Central de atendimento para atender a problemas específicos

Atividade

COMO vocês fariam esse trabalho? Precisamos de um roteiro para ajudar a equipe a fazer esse desenvolvimento.

Identifique as principais atividades que deveriam ser realizadas

Essas atividades podem ser de desenvolvimento de sistemas, de gerenciamento, ...

Tarefa em grupo. Duração de 10 minutos

Atividade (2)

Pensando agora em **QUEM** fará cada uma dessas atividades que vocês elencaram.

Quem serão as pessoas envolvidas nesse projeto?

- Pense que não são apenas desenvolvedores de software, pois os funcionários da escola, por exemplo, também podem contribuir.
- Podem existir mais de uma pessoa em cada atividade?
- Haverá um responsável pela atividade?

Defina quais os papeis e responsabilidades de cada pessoa envolvida no projeto?

Tarefa em grupo. Duração de 10 minutos

Atividade (2)

Pensando agora em **O QUE** será definido em cada uma das atividades.

Para cada atividade desenvolvida, algum produto resultado desse trabalho deve ser gerado

Ex. documento com as funcionalidades, código, etc.

Defina o(s) produtos (artefatos) de cada atividade

Tarefa em grupo. Duração de 10 minutos

Voltando à definição de Engenharia de Software

- "A *engenharia de software* é um ramo da engenharia cujo foco é o desenvolvimento dentro de *custos adequados* de sistemas de software de *alta qualidade*." *Roger Pressman*

- *"Uma disciplina da engenharia relacionada com todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a sua manutenção, após entrar em operação." Ian Sommervile*

- *"Aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software." IEEE*

Portanto ...

Não podemos construir software de maneira caótica

Precisamos sistematizar o desenvolvimento e usar métodos adequados

Camadas da engenharia de software (Pressman, 2016)

1. Qualquer abordagem deve estar fundamentada no comprometimento com a qualidade
2. A base é a camada de processos – é a liga que mantem as camadas coesas
3. Os métodos fornecem as informações técnicas para desenvolver o software
4. As ferramentas fornecem suporte automatizado para o processo e para os métodos

O que é um Processo de software?

Conjunto de atividades e resultados associados que produzem um produto de software (sommerville, 2011)

Espécie de roteiro que defina, passo a passo, como criar um software de alta qualidade e no prazo estabelecido (Pressman, 2016 p.30)

As atividades são realizadas por papeis e produzem / consomem artefatos

- Processos são complexos, pois dependem do julgamento humano
- Não há um processo ideal, diferentes organizações usam diferentes abordagens

Quais são essas “atividades” de um processo de software?

- Em geral todo processo de software se baseia em um conjunto de atividades metodológicas genérico:
  - Comunicação – comunicação com o cliente para entender os **objetivos** do projeto e reunir os **requisitos** que ajudem a definir os recursos e as funções do sistema
  - Planejamento – **plano de projeto** de software com as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos necessários, os produtos resultantes a ser produzido e o cronograma de trabalho
  - Modelagem – Criação de um **esboço do sistema** e refinamento deste, quando necessário
  - Construção - **codificação e teste**
  - Entrega – software é entregue ao cliente que o avalia e fornece *feedback*

Existem também atividades de apoio ...

- Controle e acompanhamento do projeto – avaliar o progresso
- Administração de riscos
- Garantia da qualidade de software
- Revisões técnicas
- Medição – define e coleta medidas
- Gerenciamento da configuração – efeitos das mudanças ao longo do processo
- Gerenciamento da capacidade reutilização
- Preparo e produção de artefatos

Diferentes tipos de sistemas precisam de diferentes processos

Por isso existem vários **modelos de processo** diferentes

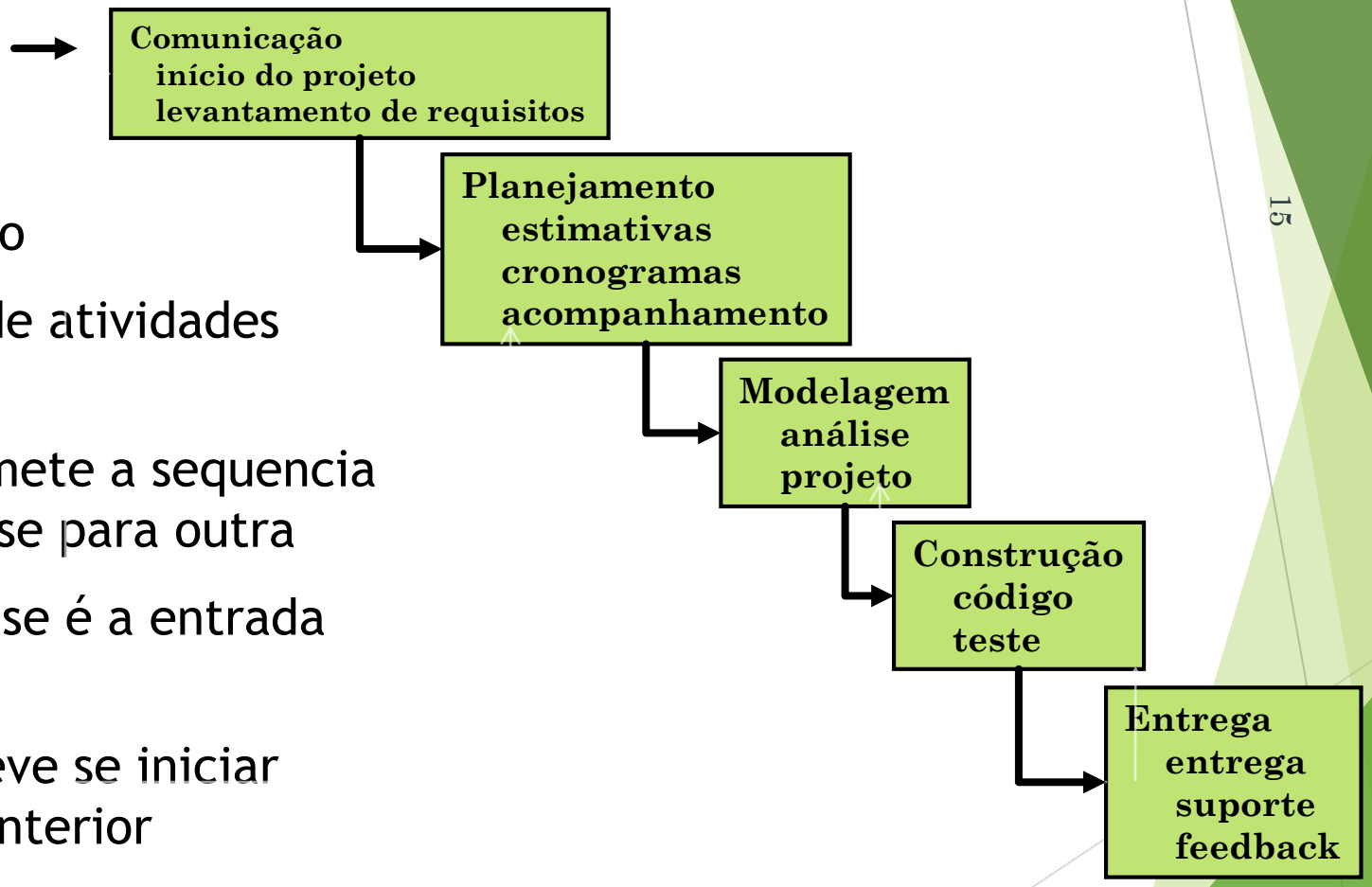
Cada um organiza as Atividades de diferentes maneiras

Os principais modelos de processo são:

- Modelo sequencial linear
  - Modelo em cascata
  - Modelo V
- Modelo incremental
- Modelo evolucionário
  - Prototipação
  - Espiral
- Modelo RAD (desenvolvimento rápido de aplicações)
- Modelos específicos
  - Reuso
  - Formal

MODELO SEQUENCIAL LINEAR EM CASCATA

- É o modelo mais antigo
- Utiliza o fluxo linear de atividades (prescritivo)
- O nome “Cascata” remete a sequencia em cascata de uma fase para outra
- O resultado de uma fase é a entrada da outra
- A fase seguinte não deve se iniciar antes do término da anterior



MODELO EM CASCATA (2)

- O modelo Cascata trouxe contribuições importantes para o processo de desenvolvimento de software:
  - imposição de disciplina, planejamento e gerenciamento
  - a implementação do produto deve ser postergada até que os objetivos tenham sido completamente entendidos

MODELO EM CASCATA (3)

- Problemas desse modelo
  - Projetos reais raramente seguem um fluxo sequencial
  - É difícil o cliente estabelecer explicitamente todas as necessidades logo no início
    - É difícil adaptação a mudanças de requisitos.
  - Iterações onerosas com grande retrabalho
  - O cliente deve ter paciência para esperar a versão ficar pronta.
    - Problemas descobertos tardiamente

Apropriado quanto todos os requisitos estão bem definidos.

MODELO EVOLUCIONÁRIO

O modelo evolucionário é projetado para desenvolver um produto que cresce e muda

(Pressman, 2016)

Modelos evolucionários são **iterativos**

Apresentam características que possibilitam desenvolver versões cada vez mais completas do software.

Duas técnicas podem ser usadas:

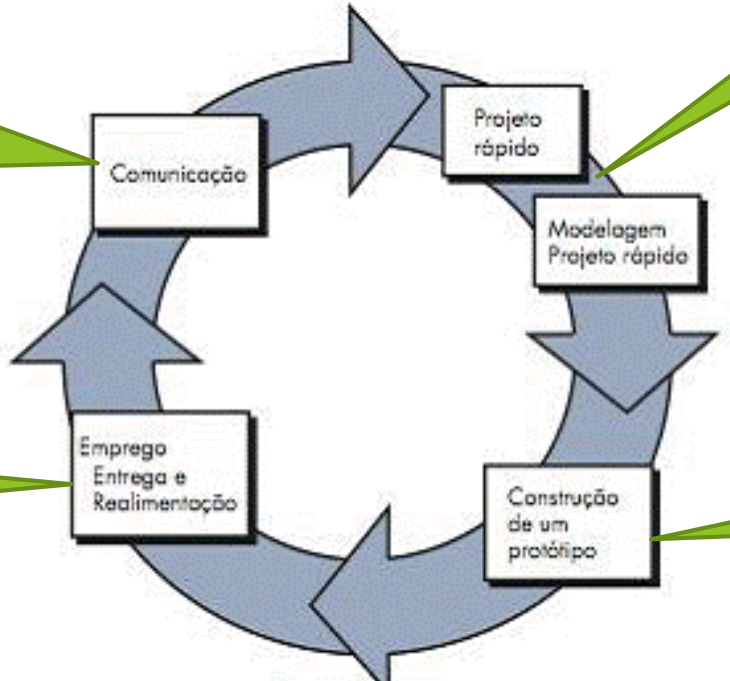
- Prototipação
- Modelo Espiral

MODELO EVOLUCIONÁRIO

Disponível em http://photobucket.com/gallery/user/Frederico_Viana_Almeida/media/bWVkaWFJZDoxNjUzNjE4OTU=/?ref=, acessado em 01/08/18

PROTOTIPAÇÃO

- Muitas vezes o cliente define os objetivos gerais do sistema, mas não identifica e detalha os requisitos
- Protótipos podem ser usados para um projeto rápido (ex. layout da interface) ou para identificar os requisitos (protótipo operacional)



representação dos aspectos do software que são visíveis ao usuário (abordagens de entrada e formatos de saída)

desenvolvedor e cliente definem os objetivos gerais do software, identificam quais requisitos são conhecidos e as áreas que necessitam de definições adicionais.

implementação rápida do projeto

cliente e desenvolvedor avaliam o protótipo

cliente e desenvolvedor refinam os requisitos do software a ser desenvolvido.

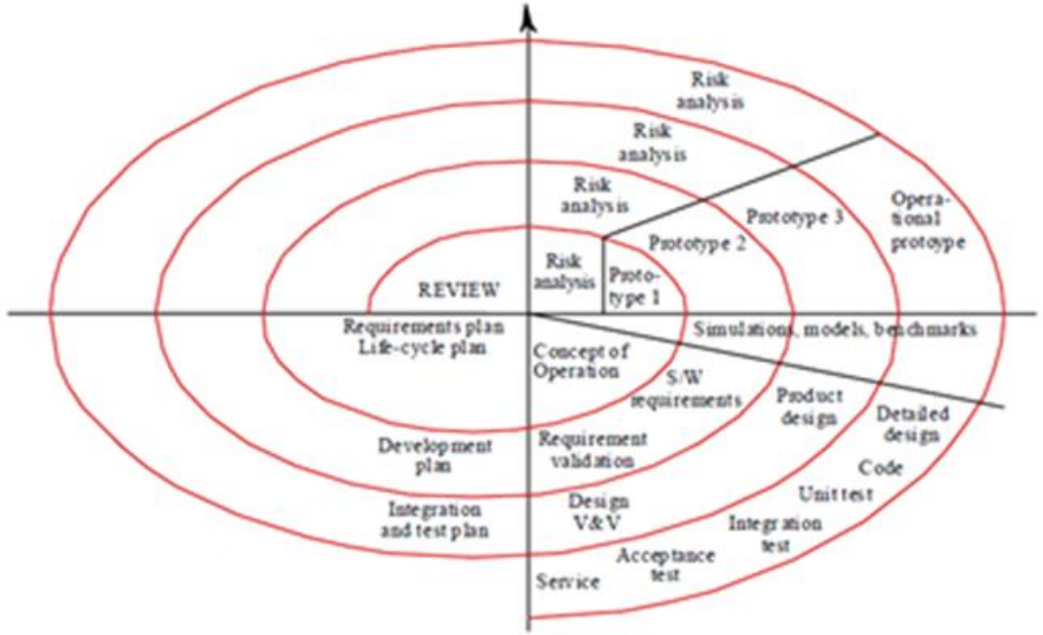
PROTOTIPAÇÃO

- identificados os requisitos, o protótipo deve ser descartado e a versão de produção deve ser construída considerando os critérios de qualidade.

- Alguns problemas da prototipação
  - cliente não sabe que o software que ele vê não considerou, durante o desenvolvimento, a qualidade global e a manutenibilidade a longo prazo
  - desenvolvedor frequentemente faz uma implementação comprometida (utilizando o que está disponível) com o objetivo de produzir rapidamente um protótipo

MODELO ESPIRAL (1)

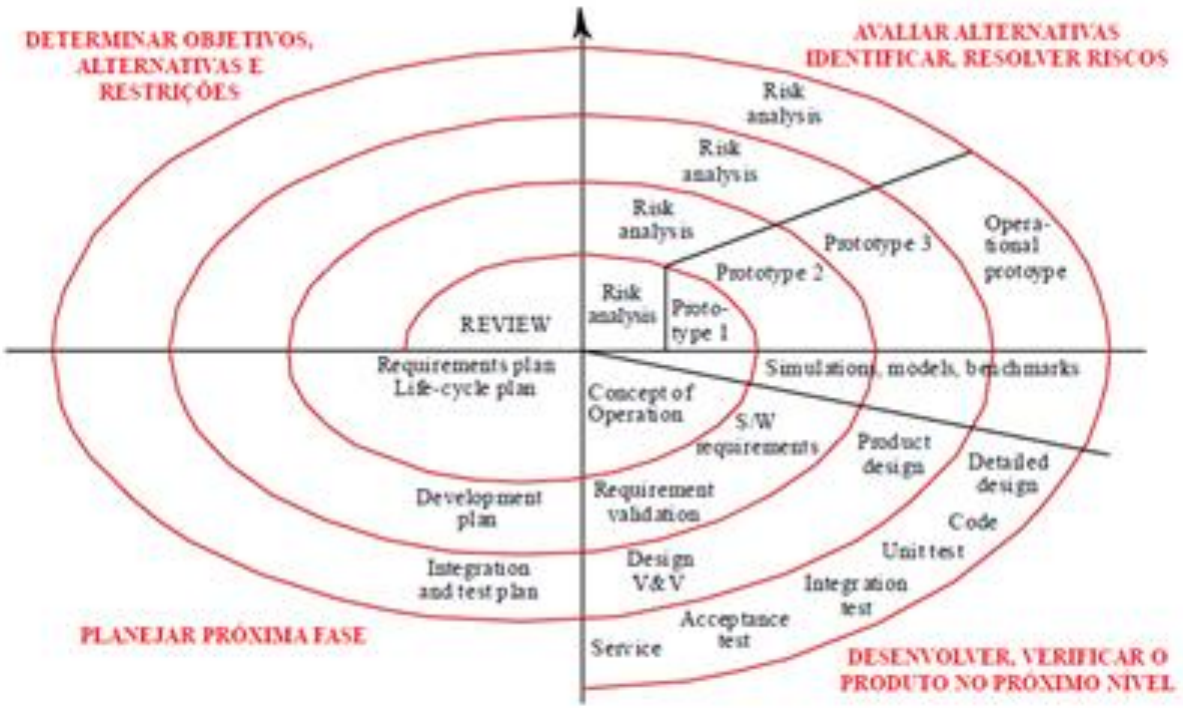
- Proposto por Boehm (1988)
- Em vez de representar o processo de software como uma sequencia de atividades com algum retorno de uma atividade para outra, o processo é representado como uma espiral



- Cada loop representa uma fase do processo
- Ex.
  - Loop 1: especificação do produto
  - Loop 2: protótipo
- Não existem fases fixas, a gerência decide como estruturar

MODELO ESPIRAL (2)

- Cada Loop é dividido em 4 setores



MODELO ESPIRAL (7)

- Combina prototipagem (iteração) com o modelo em cascata (controle e sistematização).
- Software é desenvolvido através de uma série de versões evolucionárias.
- Tem grande ênfase em gerenciamento de riscos
  - usa a Prototipação, em qualquer etapa da evolução do produto, como mecanismo de redução de riscos

MODELO INCREMENTAL (1)

<u>Modelo cascata</u>

- Requer que clientes definam os requisitos antes de iniciar o projeto
- Mudanças em requisitos são onerosas
- Leva a sistemas robustos e suscetíveis a mudanças

X

<u>Modelo evolucionário</u>

- Permite que requisitos e decisões sejam adiadas
- Leva a um software que pode ser mal estruturado, de difícil compreensão e manutenção

Modelo incremental (Mills, 1980)

Redução do retrabalho
Proporciona aos clientes oportunidade de adiar decisões sobre requisitos

MODELO INCREMENTAL (2)

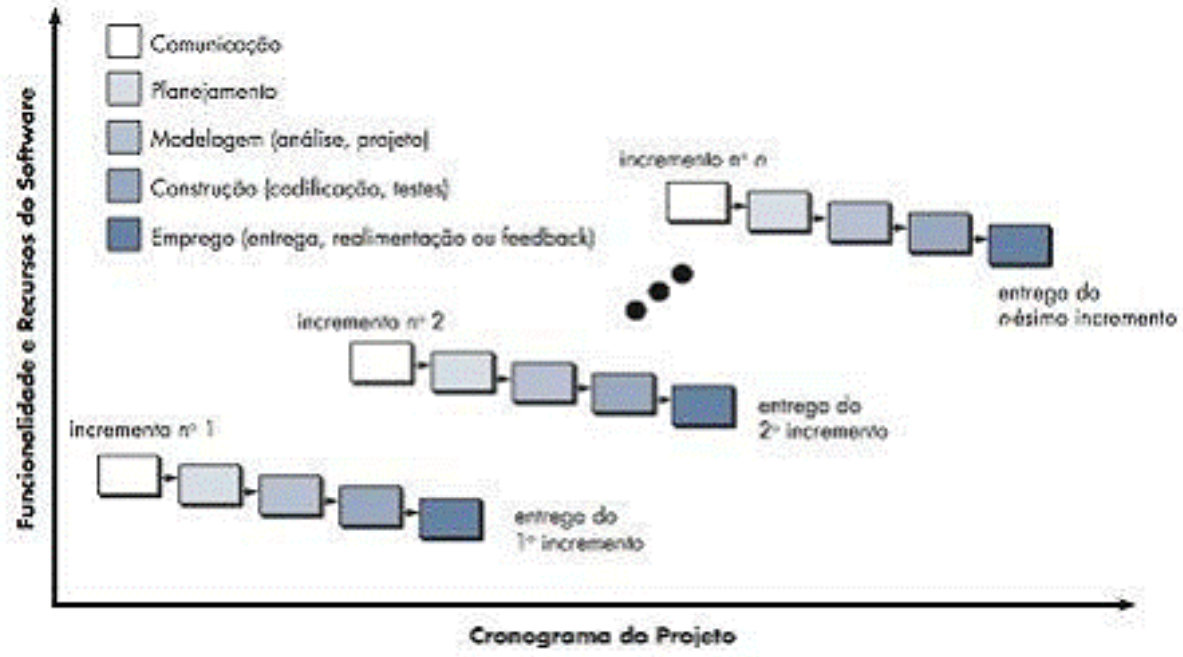
Disponível em http://photobucket.com/gallery/user/Frederico_Viana_Almeida/media/bWVkaWFJZDoxNjUzNjE2NTc=/?ref=
Acessado em 01/08/2018

MODELO INCREMENTAL (3)

- Usuários identificam, em um esboço, as funções a serem fornecidas pelo sistema
- Identificam quais funções são mais importantes e menos importantes
- Define estágios de entrega, onde cada estágio tem um subconjunto de funcionalidades
- A alocação da funcionalidade ao estágio depende da sua prioridade
- As funções prioritárias são entregues primeiramente
- Cada incremento é definido em detalhe e construído
- Durante esse incremento outras análises de requisitos para incrementos futuros podem ocorrer, mas mudanças nos requisitos do incremento atual não são aceitas
- O incremento concluído é colocado em operação
- Usuários recebem com antecedência versões implementadas, podem experimentá-las
- Novos incrementos são integrados aos anteriores

MODELO INCREMENTAL (4)

- Pode-se usar vários processos em incrementos diferentes (cascata, evolucionário)
- **Em cada incremento produz-se um sistema totalmente funcional embora não tenha todos os requisitos**



Modelo incremental (Pressman, 2016)

MODELO INCREMENTAL (5)

Vantagens

- O cliente não precisa esperar que o sistema seja entregue para usá-lo
- O primeiro estágio satisfaz os requisitos mais importantes
- Os primeiros incrementos podem ser usados como protótipo para obter experiência
- Existe menor risco de fracasso completo do sistema
- Como as funções prioritárias são entregues primeiro, os incrementos são integrados a esta primeira versão, por isso as funções mais importantes passam também por mais testes

Atualmente é a abordagem mais comum no desenvolvimento de software

MODELO INCREMENTAL (6)

Problemas

- Incrementos devem ser relativamente pequenos
- Cada incremento deve produzir alguma funcionalidade para o sistema, então pode ser difícil mapear os requisitos em incrementos de tamanho certo

MODELO DE DESENVOLVIMENTO RÁPIDO (RAD – RAPID APPLICATION DEVELOPMENT)

RAD (Rapid Application Development) é um modelo sequencial linear que enfatiza um ciclo de desenvolvimento extremamente curto

O desenvolvimento rápido é obtido usando uma abordagem de construção baseada em componentes.

- Desenvolvimento extremamente curto (60 a 90 dias)
- Reutiliza componentes
- Os requisitos devem ser bem entendidos e o alcance do projeto restrito
- Usado principalmente para aplicações de sistema de informação
- Cada função principal pode ser direcionada para uma equipe RAD separada e então integrada para formar o todo.

MODELO RAD

- Exige recursos humanos suficientes para todas as equipes
- Exige que desenvolvedores e clientes estejam comprometidos com as atividades de “fogo-rápido” a fim de terminar o projeto num prazo curto

- Nem todos os tipos de aplicação são apropriadas para o RAD:
  - deve ser possível a modularização efetiva da aplicação
  - se alto desempenho é uma característica e o desempenho é obtido sintonizando as interfaces dos componentes do sistema, a abordagem RAD pode não funcionar

MODELOS DE PROCESSO ESPECIALIZADOS

- **Inclui características dos modelos apresentados anteriormente**
- **Alguns exemplos são:**
  - **Modelo orientado a reuso**
  - **Modelo de métodos formais**

TECNOLOGIA QUE APOIAM A DEFINIÇÃO DE PROCESSOS

- Os modelos de processo são adaptados para serem utilizados pelas equipes de desenvolvimento de software
- Para isso existem linguagens (PML – Process Modeling Langauges) e ferramentas para apoiar a definição de processos
- A definição de processos envolve diversos elementos:
  - Fases e iterações
  - Atividades
    - E subatividades ou passos
  - Artefatos que serão produzidos e utilizados
  - Papeis responsáveis pelas atividades...

Falaremos um pouco mais depois sobre o a PML SPEM e a ferramenta EPF

MUDANÇAS

- Mudanças são inevitáveis em projetos de software
- Mudanças no negócio e na tecnologia são exemplos de fatores que levam a mudanças no software
- Mudanças geralmente significam trabalho, ou retrabalho, a ser feito e geram custo

- Duas abordagens podem ser adotadas para reduzir o custo do retrabalho
  - Prevenção de mudanças: o projeto inclui atividades para antecipar mudanças (ex. **protótipos**)
  - Tolerância a mudanças: o processo possibilita que mudanças sejam acomodadas a um custo relativamente baixo. (ex. **desenvolvimento incremental**)

ATIVIDADE

Qual o modelo de processo que você utilizaria para desenvolver o sistema da matrícula?

Justifique