Projeto de Sistemas Orientado a Objetos

Aula 02 - Processo de Desenvolvimento de Software

Capítulo 2

Processo de Desenvolvimento de Software

"Quanto mais livros você leu (ou escreveu), mais as aulas você assistiu (ou lecionou), mais linguagens de programação você aprendeu (ou projetou), mais software OO você examinou (ou produziu), mais documentos de requisitos você tentou decifrar (ou tornou decifrável), mais padrões de projeto você aprendeu (ou catalogou), mais reuniões você assistiu (ou conduziu), mais colegas de trabalho talentosos você teve (ou contratou), mais projetos você ajudou (ou gerenciou), tanto mais você estará equipado para lidar com um novo desenvolvimento." - Bertrand Meyer

“Software is hard…”

- Porcentagem de projetos que terminam dentro do prazo estimado: 10%
- Porcentagem de projetos que são descontinuados antes de chegarem ao fim: 25%
- Porcentagem de projetos acima do custo esperado: 60%
- Atraso médio nos projetos: um ano.

Princípios d

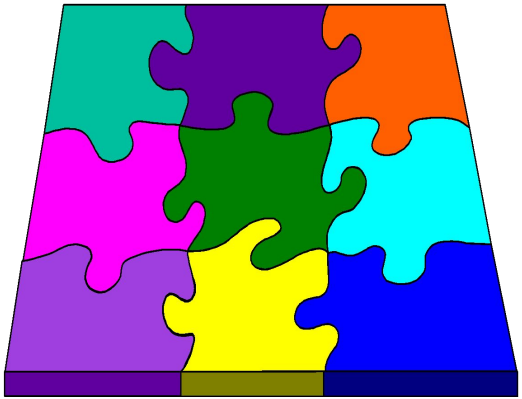
Fonte: Chaos Report (1994)

Processo de desenvolvimento

- Tentativas de lidar com a complexidade e de minimizar os problemas envolvidos no desenvolvimento de software envolvem a definição de ***processos de desenvolvimento de software***.
- Um processo de desenvolvimento de software (PDS) compreende todas as atividades necessárias para definir, desenvolver, testar e manter um produto de software.

Processo de desenvolvimento

- Exemplos de processos de desenvolvimento existentes:
  - ICONIX
  - RUP
  - EUP
  - XP
  - OPEN
- Alguns objetivos de um processo de desenvolvimento são:
  - Definir *quais* as atividades a serem executadas ao longo do projeto;
  - Definir *quando*, *como* e por *quem* tais atividades serão executadas;
  - Prover pontos de controle para verificar o andamento do desenvolvimento;
  - Padronizar a forma de desenvolver software em uma organização.

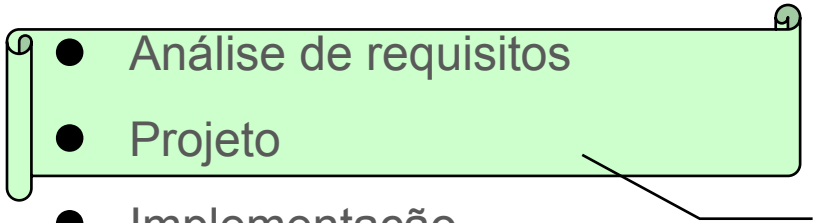


2.1 Atividades típicas de um PDS

2.2 O componente humano em um PDS

Atividades típicas de um PDS

- Levantamento de requisitos
- Análise de requisitos
- Projeto
- Implementação
- Testes
- Implantação



Foco do livro

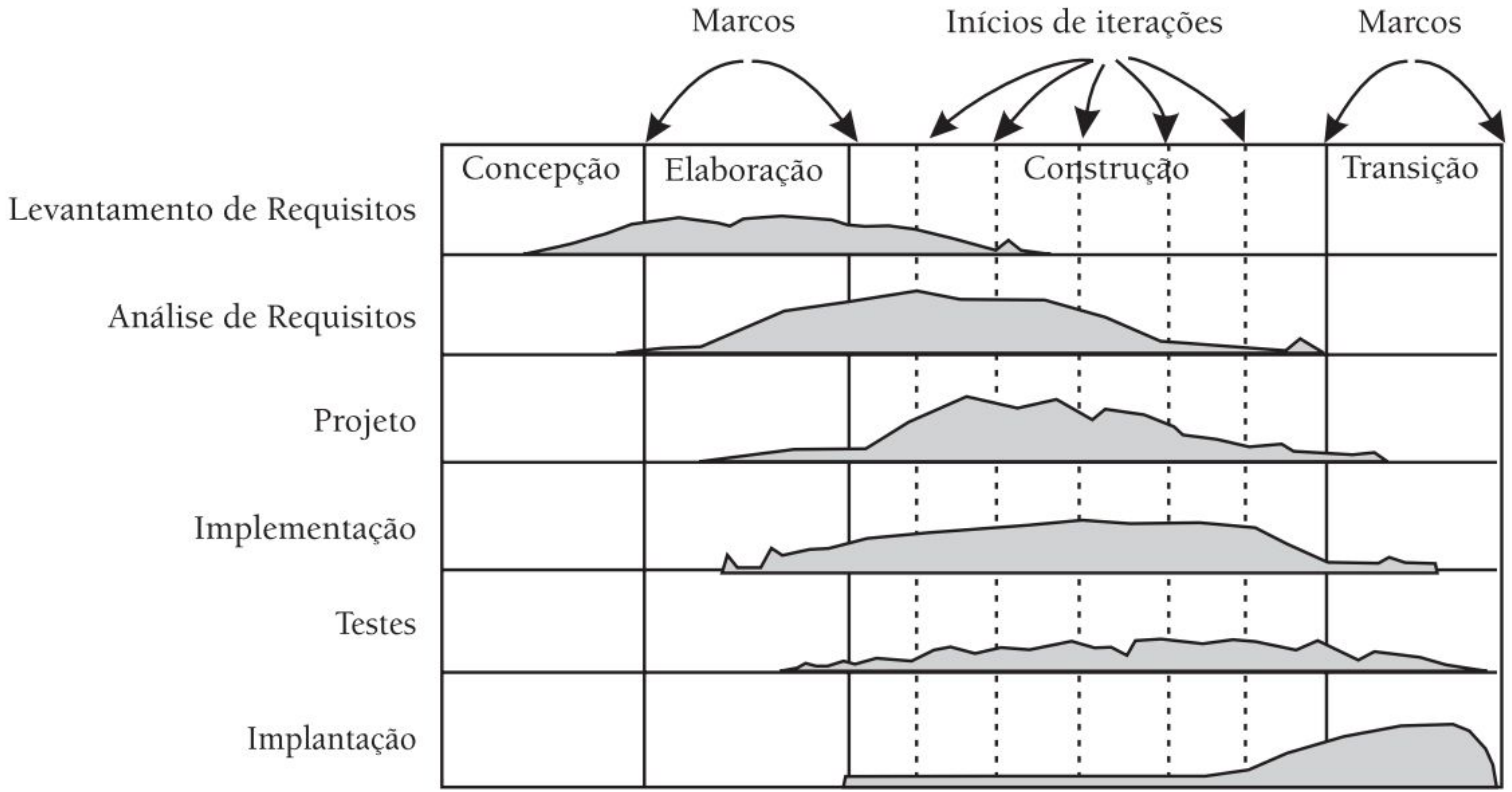


Figura 2-3: Estrutura geral de um processo de desenvolvimento incremental e iterativo.

1. Levantamento de Requisitos

- Participantes: Cliente e analistas
- Etapa de compreensão do problema
- Objetivo: Cliente e desenvolvedores tenham a mesma visão do problema a ser resolvido
- Desenvolvedores e cliente definem as necessidades dos futuros usuários
- Necessidades são chamadas de requisitos

1.1 Requisitos funcionais

Definem as funcionalidades do sistema, exemplos:

- “O sistema deve permitir que cada professor realize o lançamento de notas das turmas nas quais lecionou.”
- “O sistema deve permitir que um aluno realize a sua matrícula nas disciplinas oferecidas em um semestre letivo.”
- “Os coordenadores de escola devem poder obter o número de aprovações, reprovações e trancamentos em cada disciplina oferecida em um determinado período.”

1.2 Requisitos não funcionais

Declaram características de qualidade que o sistema deve possuir

- Confiabilidade: corresponde a medidas quantitativas da confiabilidade do sistema, tais como tempo médio entre falhas, recuperação de falhas ou quantidade de erros por milhares de linhas de código-fonte.
- Desempenho: requisitos que definem tempos de resposta esperados para as funcionalidades do sistema.
- Portabilidade: restrições sobre as plataformas de hardware e de software nas quais o sistema será implantado e sobre o grau de facilidade para transportar o sistema para outras plataformas.

1.2 Requisitos não funcionais (Cont.)

Declaram características de qualidade que o sistema deve possuir

- Segurança: limitações sobre a segurança do sistema em relação a acessos não autorizados.
- Usabilidade: requisitos que se relacionam ou afetam a usabilidade do sistema. Exemplos incluem requisitos sobre a facilidade de uso e a necessidade ou não de treinamento dos usuários.

1.3 Requisitos Normativos

Declaram as restrições impostas sobre o desenvolvimento do sistema

- Adequação a custos e prazos
- Plataforma tecnológica
- Aspectos legais
- Limitação sobre a interface com o usuário
- Componentes de hardware e software a serem adquiridos
- Eventuais necessidades de comunicação do sistema novo com o distema legado

2. Análise (de Requisitos)

- Participantes: Analistas
- Objetivo: Analistas compreenderem os requisitos e construírem modelo para representar o sistema.
- Entender o que fazer, e não como fazer
- Os modelos construídos devem ser validados e verificados
- Pode ser dividido em Análise do Domínio e Análise da Aplicação

2.1 Análise do Domínio

- Analisa os objetos do mundo real que serão modelados pelo sistema
- Exemplo: Um sistema de controle acadêmico deve tratar dos objetos Aluno, Professor, Disciplina, Matéria, Turma, etc...

2.2 Análise da Aplicação

- Analisa os objetos (outros que não são do mundo real) que serão modelados pelo sistema
- Exemplo: A tela que os alunos irão usar o software. Deve ser em um app ou na web?

3. Projeto

- Participantes: Projetista
- Objetivo: Complementando a Análise, o projeto se preocupa com “como fazer”
- Planejar a implementação do sistema
- Considera os aspectos físicos e dependentes de aplicações
- Esta fase produz uma descrição computacional do que o software deve fazer e deve ser coerente com a descrição feita na análise.
- O projeto consiste em duas atividades principais: projeto da arquitetura (também conhecido como projeto de alto nível) e projeto detalhado (também conhecido como projeto de baixo nível).

3.1 Projeto de Arquitetura

- Distribuir as classes de objetos relacionados dos sistema em subsistemas
- Distribuir esses componentes fisicamente pelos recursos de hardware disponíveis.
- Onde é feito o diagrama de implementação

3.1 Projeto Detalhado

- Distribuir as classes de objetos relacionados dos sistema em subsistemas
- Distribuir esses componentes fisicamente pelos recursos de hardware disponíveis.
- Onde é feito os diagramas
  - Diagrama de classes
  - Diagrama de caso de uso
  - Diagrama de interação
  - Diagrama de estados
  - Diagrama de atividades

4. Implementação

- Participantes: Desenvolvedores
- Objetivo: Implementar o sistema, fazer o código para o programa funcionar
- Tradução da descrição para a máquina em código executável

5. Testes

- Participantes: Testadores
- Objetivo: testar o sistema para saber se corresponde à especificação feita na fase de projetos
- Os componentes são integrados e o sistema final é testados

6. Implantação

- O sistema é empacotado, distribuído e instalado no ambiente do usuário.
- Os manuais do sistema são escritos, os arquivos são carregados, os dados são importados para o sistema, e os usuários treinados para utilizar o sistema corretamente.

Participantes do processo

- Gerentes de projeto
- Analistas
- Projetistas
- Arquitetos de software
- Programadores
- Clientes
- Avaliadores de qualidade

1. Gerente do Projeto

- Responsável pela gerência ou coordenação das atividades necessárias à construção do sistema.
- O responsável por fazer o orçamento do projeto de desenvolvimento, como, por exemplo, estimar o tempo necessário para o desenvolvimento do sistema, definir qual o processo de desenvolvimento, o cronograma de execução das atividades, a mão de obra especializada, os recursos de hardware e software etc.
- O acompanhamento das atividades realizadas
- verificar se os diversos recursos alocados estão sendo gastos na taxa esperada e, caso contrário, tomar providências para adequação dos gastos.

2. Analistas

- Deve entender os problemas do domínio do negócio para que possa definir os requisitos do sistema a ser desenvolvido.
- Aptos a se comunicar com especialistas do domínio para obter conhecimento acerca dos problemas e das necessidades envolvidas na organização empresarial.
- O analista não precisa ser um especialista. Contudo, ele deve ter suficiente domínio da área de conhecimento na qual o sistema será implantado para se comunicar com o especialista de domínio.
- Entender as necessidades dos clientes e repassar aos demais desenvolvedores do sistema
- A ponte de comunicação entre os profissionais de computação e os profissionais do negócio.

3. Projetista

- Avaliar as alternativas de solução (da definição) do problema resultante da análise
- Gerar a especificação de uma solução computacional detalhada
- Existem diversos tipos de projetistas
  - Projetistas de interface (especializados nos padrões de uma interface gráfica, como o Windows ou o MacOS),
  - Projetistas de redes (especializados no projeto de redes de comunicação),
  - Projetistas de bancos de dados

4. Arquitetos de software

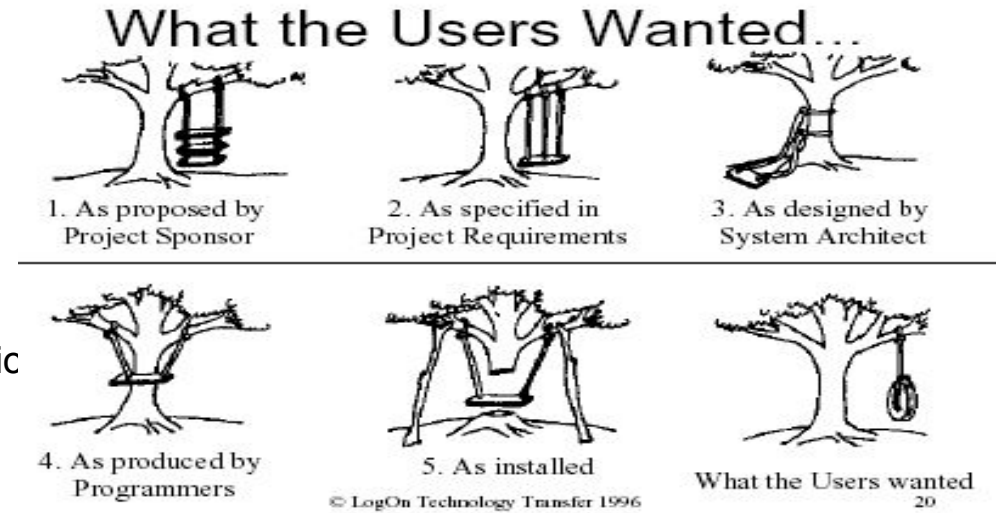
- Elaborar a arquitetura do sistema como um todo.
- Decisões sobre quais são os subsistemas que compõem o sistema como um todo e quais são as interfaces entre esses subsistemas.

5. Programadores

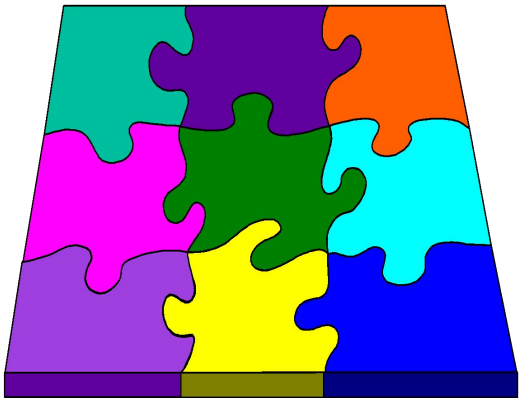
- Responsável pela implementação do sistema.
- É comum haver vários programadores em uma equipe de desenvolvimento.
- Um programador pode ser proficiente em uma ou mais linguagens de programação, além de ter conhecimento sobre bancos de dados e poder ler os modelos resultantes do trabalho do projetista.

Participação do usuário

- A participação do usuário durante o desenvolvimento de um sistema extremamente é importante.



Princípic



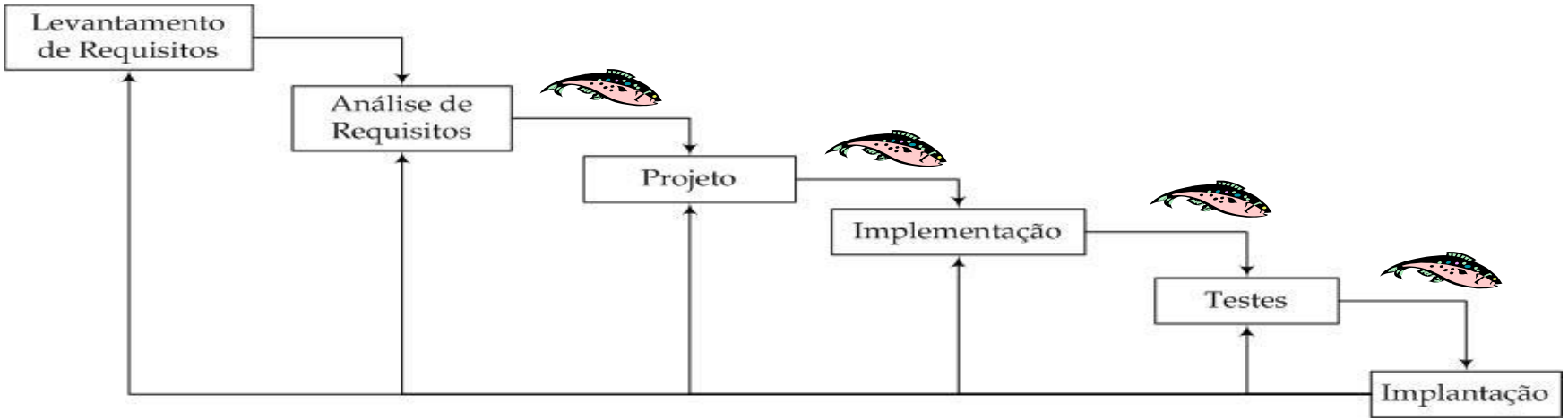
2.3 Modelos de ciclo de vida

Modelo de ciclo de vida

- Um ciclo de vida corresponde a um encadeamento específico das fases para construção de um sistema.
- Dois modelos de ciclo de vida:
  - *modelo em cascata*
  - *modelo iterativo e incremental*.

Modelo em cascata

- Esse modelo apresenta uma tendência para a progressão seqüencial entre uma fase e a seguinte.



Modelo em cascata

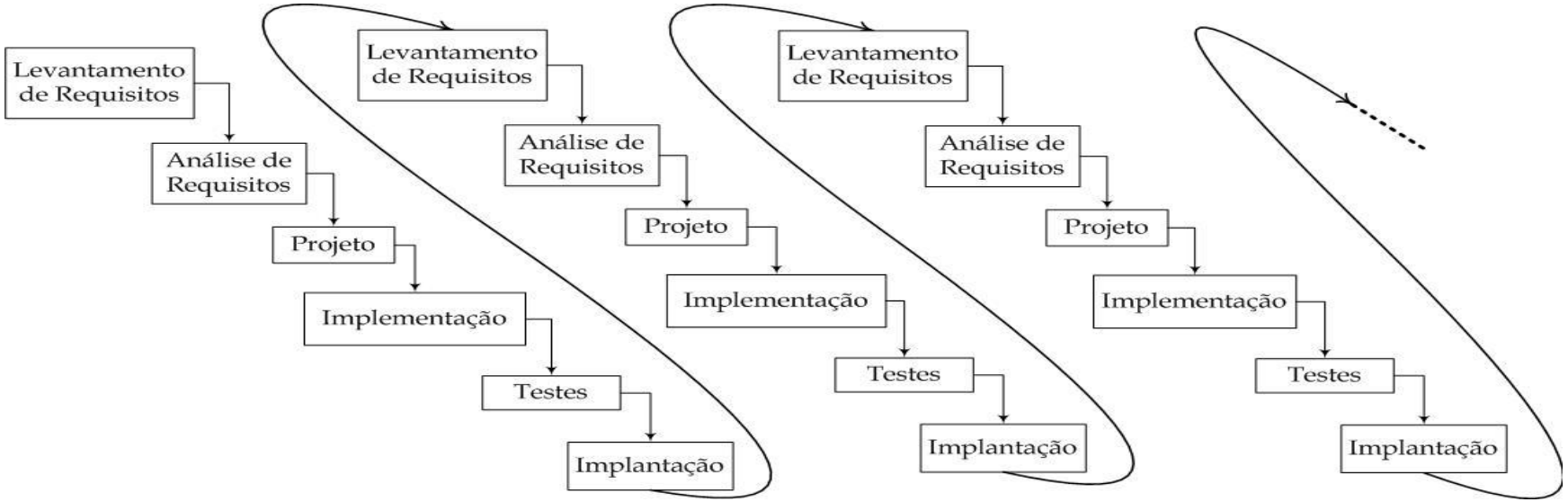
- Projetos reais raramente seguem um fluxo seqüencial.
- Assume que é possível declarar detalhadamente todos os requisitos antes do início das demais fases do desenvolvimento.
  - propagação de erros pelas as fases do processo.
- Uma versão de produção do sistema não estará pronta até que o ciclo do projeto de desenvolvimento chegue ao final.

Modelo de iterativo e incremental

- Divide o desenvolvimento de um produto de software em ***ciclos***.
- Em cada ciclo de desenvolvimento, podem ser identificadas as fases de análise, projeto, implementação e testes.
- Cada ciclo considera um subconjunto de requisitos.
- Esta característica contrasta com a abordagem clássica, na qual as fases são realizadas uma única vez.

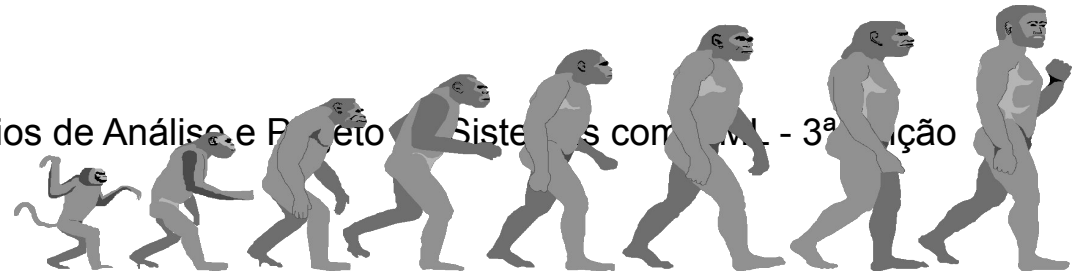
Modelo iterativo e incremental

- Desenvolvimento em “mini-cascatas”.



Modelo iterativo e incremental

- ***Iterativo**:* o sistema de software é desenvolvido em vários passos similares.
- ***Incremental**:* Em cada passo, o sistema é estendido com mais funcionalidades.



Modelo iterativo e incremental – vantagens e desvantagens

👍 Incentiva a participação do usuário.

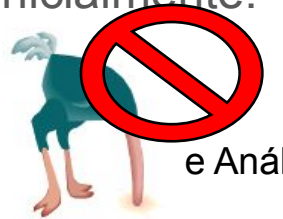
👍 R*iscos* do desenvolvimento podem ser mais bem gerenciados.

- Um ***risco de projeto*** é a possibilidade de ocorrência de algum evento que cause prejuízo ao processo de desenvolvimento, juntamente com as conseqüências desse prejuízo.
- Influências: custos do projeto,cronograma, qualidade do produto, satisfação do cliente, etc.

☹ Mais difícil de gerenciar

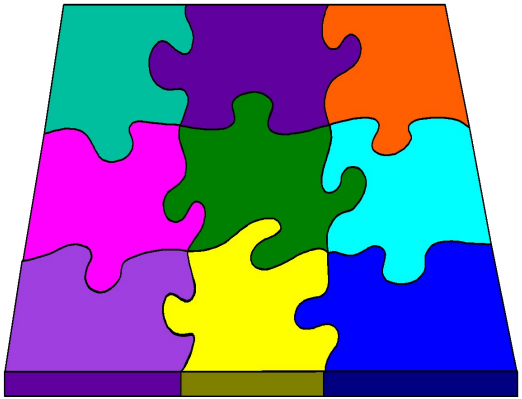
Ataque os riscos

- *“Se você não atacar os riscos [do projeto] ativamente, então estes irão ativamente atacar você.”* (Tom Gilb).
  - A maioria dos PDS que seguem o modelo iterativo e incremental aconselha que as partes mais arriscadas sejam consideradas inicialmente.



Riscos não gerenciados

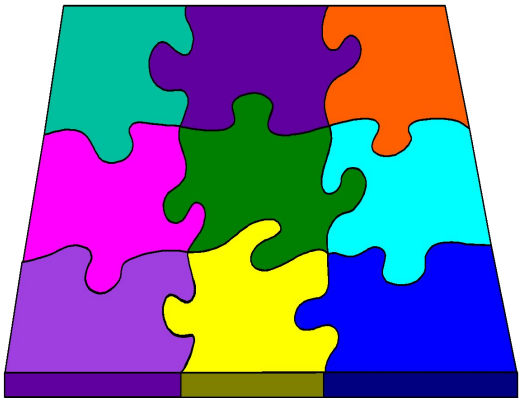
PDS



2.4 Utilização da UML no modelo iterativo e incremental

UML no modelo iterativo e incremental

- A UML é independente do processo de desenvolvimento.
  - Vários processos podem utilizar a UML para modelagem de um sistema OO.
- Os artefatos de software construídos através da UML evoluem à medida que o as iterações são realizadas.
  - A cada iteração, novos detalhes são adicionados a esses artefatos.
  - Além disso, a construção de um artefato fornece informações para adicionar detalhes a outros.



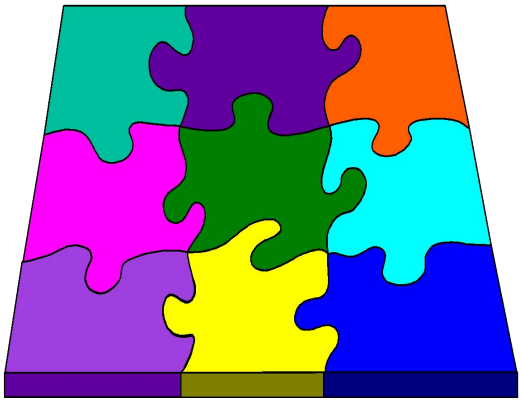
2.5 Prototipagem

Prototipagem

- A ***prototipagem*** é uma técnica aplicada quando:
  - há dificuldades no entendimento dos requisitos do sistema
  - há requisitos que precisam ser mais bem entendidos.
- A construção de ***protótipos*** utiliza ambientes com facilidades para a construção da interface gráfica.
- Procedimento geral da prototipagem:
  - Após o LR, um protótipo é construído para ser usado na *validação*.
  - Usuários fazem críticas...
  - O protótipo é então corrigido ou refinado
  - O processo de revisão e refinamento continua até que o protótipo seja aceito.
  - Após a aceitação, o protótipo é descartado ou utilizado como uma versão inicial do sistema.

Prototipagem

- Note que a prototipagem NÃO é um substituto à construção de modelos do sistema.
  - A prototipagem é uma técnica *complementar* à construção dos modelos do sistema.
  - Mesmo com o uso de protótipos, os modelos do sistema devem ser construídos.
  - Os erros detectados na validação do protótipo devem ser utilizados para modificar e refinar os modelos do sistema.



2.6 Ferramentas de suporte

Ferramentas de suporte

- O desenvolvimento de um software pode ser facilitado através do uso de ferramentas que auxiliam:
  - na construção de modelos,
  - na integração do trabalho de cada membro da equipe,
  - no gerenciamento do andamento do desenvolvimento, etc.

Ferramentas de suporte

- Há diversos sistemas de software que são utilizados para dar suporte ao desenvolvimento de outros sistemas.
- Um tipo bastante conhecido de ferramenta de suporte são as ***ferramentas CASE***.
  - CASE: *Computer Aided Software Engineering*
- Além das ferramentas CASE, outras ferramentas importantes são as que fornecem suporte ao ***gerenciamento***.
  - desenvolver cronogramas de tarefas,
  - definir alocações de verbas,
  - monitorar o progresso e os gastos,
  - gerar relatórios de gerenciamento, etc.

Ferramentas de suporte

- Criação e manutenção da consistência entre estes diagramas
- *Round-trip engineering*
- Depuração de código fonte
- Relatórios de testes
- Testes automáticos
- Gerenciamento de versões
- Verificação de desempenho
- Verificação de erros em tempo de execução
- Gerenciamento de mudanças nos requisitos
- Prototipagem