Projeto de software

Bruna Diirr

brunadiirr@ic.uff.br

Desenvolvimento de software

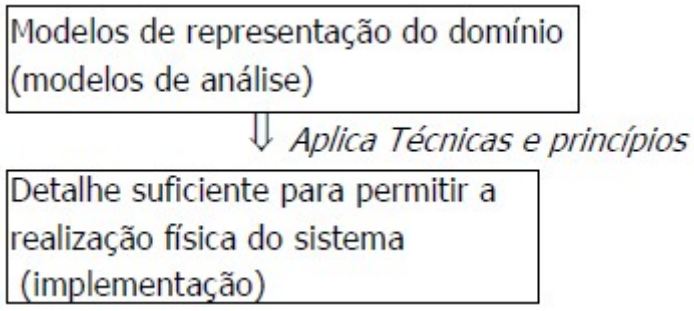
 Levantamento de requisitos	• Compreensão do problema
 Análise	• O que o software deve fazer • Para quem devem ser construídas as funcionalidades
 Projeto	• Definição da solução computacional para implementação dos requisitos (como?)
 Implementação	• Tradução do projeto em algo passível de execução por máquinas (codificação)
 Testes	• Verificação se o código atende ao projeto e implementa corretamente os requisitos
 Implantação	• Sistema é empacotado, distribuído e instalado no ambiente do usuário (software em produção)
 Manutenção	• Correção de erros presentes no software • Evolução do software

Projeto de software

"Fase que visa projetar uma estrutura de software que realiza a especificação" (Sommerville)

Conjunto de atividades e artefatos que visa conceber uma solução lógica para o problema proposto pelos requisitos do sistema
- Adaptar resultados da análise a restrições do ambiente de implementação
- Refinar modelos para adequarem-se a requisitos de desempenho

Ainda envolve modelagem do sistema!
- Ênfase em resolução de problemas
- Pensar antes para evitar dificuldades durante a implementação
- De acordo com princípios bem estabelecidos



Projeto x Análise

Análise

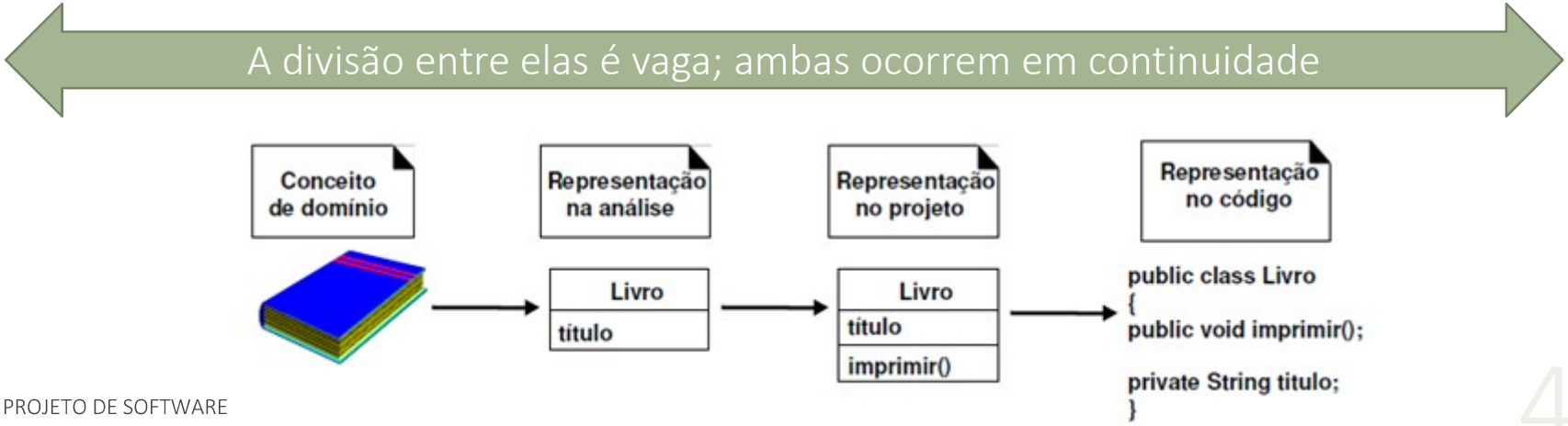
Construção do modelo do mundo real, com objetivo de entender o problema

Transformação do conhecimento dos especialistas do domínio e do cliente em um modelo não ambíguo dos requisitos do sistema

Projeto

Construção do modelo de solução computacional para o problema

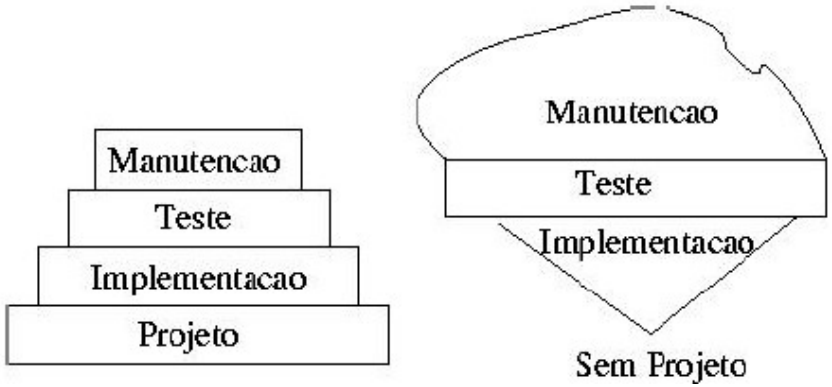
Refinamento progressivo dos requisitos do sistema até o nível de detalhamento do código



Importante...

Fase na qual a qualidade é criada e poderá ser efetivamente avaliada

Fundamental para fases de implementação, teste e manutenção



"tentamos resolver o problema passando rapidamente pelo processo de projeto para que sobre tempo suficiente para detectar e corrigir defeitos que foram introduzidos porque dedicamos pouco tempo ao projeto."

... mas nem sempre fácil!

Projetar
- Envolve criatividade
- Exige avaliação de diferentes opções
- Requer conhecimento sobre contexto da aplicação

→ Para cada sistema, um projeto diferente pode ser necessário

Mas existem algumas soluções e princípios recorrentes
- Quase universalmente aceitos ou
- Adequados para certas situações

Queremos evitar projetos ruins!

Características

Rigidez: Alteração em um módulo gera cascata de alterações em módulos dependentes

Fragilidade: Tendência de sistema falhar em diversos pontos quando uma simples alteração é realizada

Imobilidade: Projetos contendo partes úteis para outros sistemas, mas esforço e risco de separar tais partes é muito grande

Opacidade: Tendência de um módulo permanecer difícil de entender

Complexidade exagerada: Projeto contendo elementos que não são úteis no presente momento, devido a antecipação exagerada de mudanças

Viscosidade: Mais fácil “ajustar” projeto do que fazer a alteração correta quando uma mudança é desejada

Repetição desnecessária: Código repetido deveria ser isolado e representado em apenas um lugar

Divisão da fase de Projeto

Projeto geral ou preliminar

Tradução da especificação do sistema em termos da arquitetura a partir do conhecimento adquirido com requisitos

- Definição de pacotes (módulos) e interfaces entre eles
- Decisão sobre uso/criação de bibliotecas e/ou componentes

Projeto detalhado

Refinamento progressivo e adição de detalhes à arquitetura visando a codificação

- Atribuição de responsabilidades entre objetos
- Construção de diagramas de classes
- Construção de diagramas de interação (sequência e colaboração)
- Levantamento de necessidades de concorrência
- Considerações de tratamento de defeitos
- Detalhamento do formato de saída (interface com usuário, relatórios, transações enviadas para outros sistemas...)
- Definição do esquema do BD (mapeamento objetos → tabelas)

Divisão da fase de Projeto

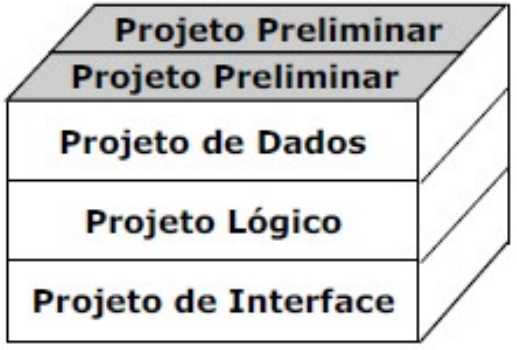
Também pode ser dividida em:

Projeto de interface: Características da interface do sistema com meio externo

Projeto de dados: Organização, armazenamento e recuperação das informações manipuladas pelo sistema

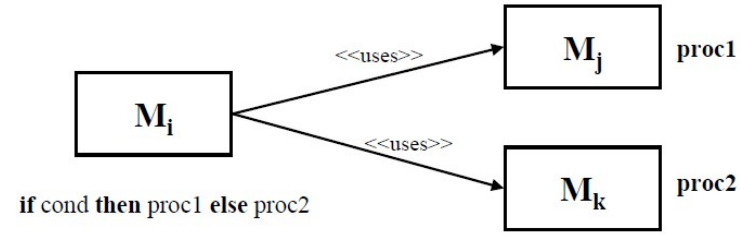
Projeto lógico: Características de processamento das informações manipuladas pelo sistema

Visões ortogonais das atividades de projeto



Projeto preliminar

Mais detalhes (arquitetura de SW) na próxima aula!



Módulo

- Componente bem definido de um sistema
  - Podem conter rotinas, dados e definições de tipos
  - Devem ser compreensíveis sem análise detalhada dos demais módulos

- Relação entre dois módulos ocorre quando um módulo depende do outro
  - É direcionada e definida em termos estáticos
  - Tipos
    - Uso: Módulo “cliente” usa rotina, variável ou tipo definido no módulo “fornecedor”
    - Composição: Indica que um módulo é composto por conjunto de outros módulos
    - Generalização-Especialização: Indica que um módulo representa uma versão mais genérica ou específica de elemento definido em outro módulo

Projeto preliminar

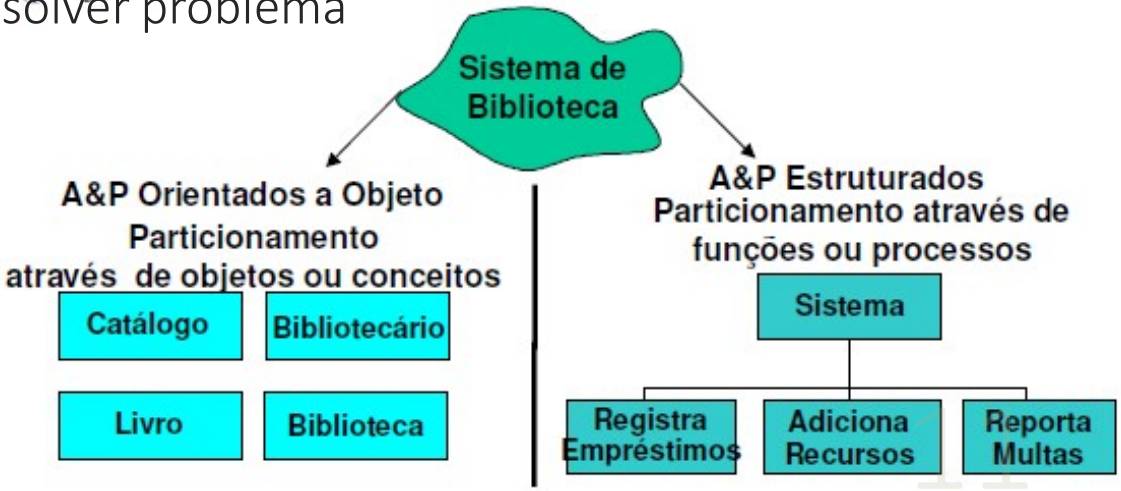
Modelo de decomposição em módulos

Especifica a decomposição de cada subsistema em módulos

Podem ser

Orientados a funções: Subsistemas decompostos em módulos funcionais que recebem dados e transformam estes dados em saída

Orientado a objetos: Subsistemas decompostos em conjunto de objetos que se comunicam para resolver problema



Projeto preliminar

Representações

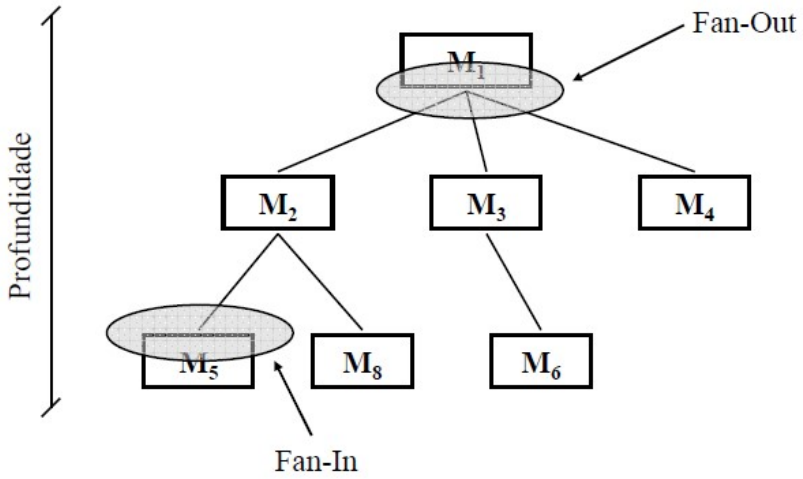
1) Árvore de dependência

Diagrama apresentando relações entre módulos do sistema

Não representa ordem de uso entre módulos, apenas suas dependências

Módulos representados como retângulos e dependências como linhas

Projeto preliminar



Representações

1) Árvore de dependência

Alguns conceitos:

Módulo superior: Módulo que controla um módulo

Módulo subordinado: Módulo controlado por outro módulo

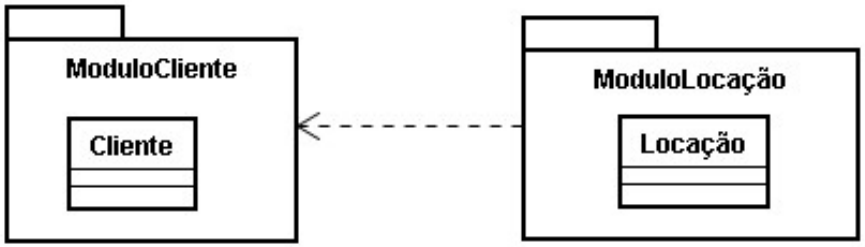
Largura: Amplitude da estrutura

Profundidade: Número de níveis de controle

Fan-out: Número de módulos controlados por dado módulo

Fan-in: Número de módulos que controlam dado módulo

Projeto preliminar



Representações

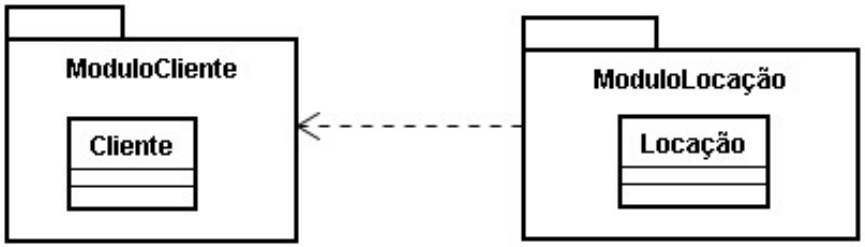
2) Diagrama de pacotes (UML)

Útil na modelagem de sistemas extensos, permitindo representar limites de cada subsistema e como eles se relacionam

Pode conter qualquer diagrama da UML, inclusive outros pacotes

- Geralmente, pacotes agregam um mesmo tipo de item
- Mais comuns para diagramas de casos de uso e de classes

Projeto preliminar



Representações

2) Diagrama de pacotes (UML)

Relacionamentos da UML também são aplicados a pacotes

Dependência, Associação, Generalização, Composição etc.

Estereótipos úteis em pacotes:

<<system>>

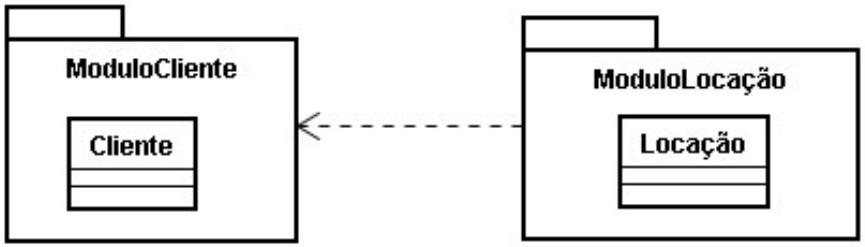
<<subsystem>> - corte "vertical"

<<framework>> - conjunto de classes abstratas e concretas concebido para ser estendido para um domínio específico

<<stub>> - serve como proxy para conteúdo público de outro pacote

<<layer>> - corte "horizontal"

Projeto preliminar



Representações

2) Diagrama de pacotes (UML)

Como selecionar classes para um pacote?

- Classes que estejam na mesma árvore de herança
- Classes que estejam em um mesmo jogo de agregação e composição
- Classes que apareçam em um mesmo diagrama de sequência com muitas colaborações (alto acoplamento)
- Um pacote utilitário contém classes sem afinidade direta com o domínio do problema, porém necessárias

Projeto de software

Bruna Diirr

brunadiirr@ic.uff.br