Engenharia de Software III

Aula 1
Visão Geral do Projeto de Software

l.bertholdo@ifsp.edu.br

Conteúdo

- Projeto de Software
- Processo de Projeto de Software
- Mapeamento dos Requisitos para o Projeto
- Conceitos de Projeto de Software

l.bertholdo@ifsp.edu.br

Projeto de Software

- O que é um **projeto** no contexto do ciclo de desenvolvimento de software?

> *Atividade que envolve a descrição da organização geral do software a ser implementado, dos modelos e estruturas de dados do sistema, das interfaces entre seus componentes e, às vezes, dos algoritmos usados.*

- O projeto de software cria uma representação ou modelo do software, mas, diferentemente do **modelo de requisitos**, que se concentra na descrição do **"que"** deve ser feito em relação às funções, aos dados e aos comportamento necessários.
- O **modelo de projeto** indica **"como"** fazer, fornecendo detalhes sobre a arquitetura de software, estruturas de dados, componentes e interfaces fundamentais para implementar o sistema.

l.bertholdo@ifsp.edu.br

Projeto de Software

- O projeto de software é uma **etapa iterativa** dentro do processo de desenvolvimento, visto que os projetistas dificilmente chegam a uma versão definitiva dele logo no primeiro ciclo deste processo.

- Pelo contrário, as descrições da estrutura do software são detalhadas, revisadas e corrigidas constantemente, a cada iteração durante o processo de desenvolvimento.

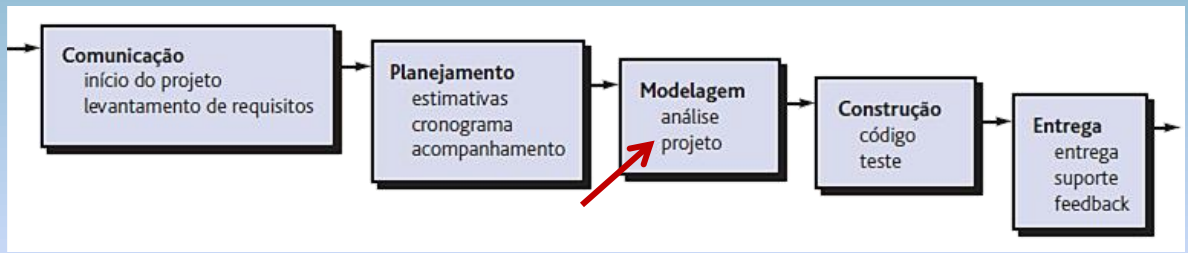
O projeto de software é importante porque as decisões tomadas nele afetam o sucesso da ***construção*** *do software e sua futura facilidade de* ***manutenção****. Além disso, o modelo gerado pode ser avaliado em termos de* ***qualidade*** *e aperfeiçoado antes de o código ser escrito e testado.* ***Grande parte da qualidade do software é determinada pelo seu projeto.***

l.bertholdo@ifsp.edu.br

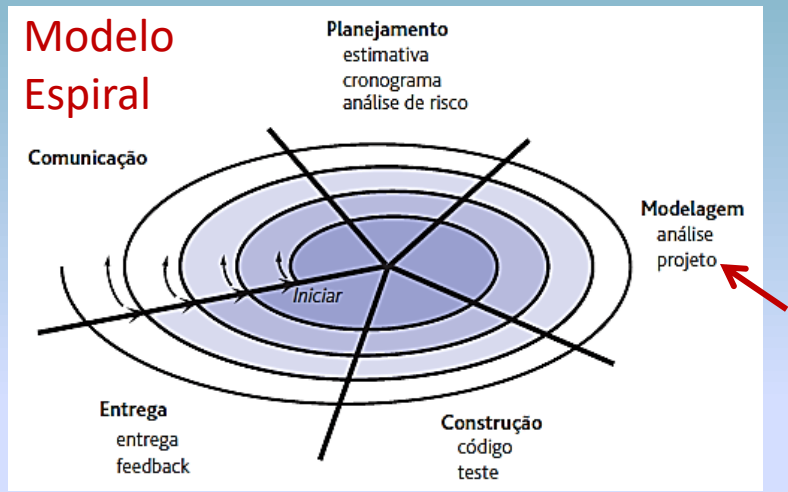
Projeto de Software

- O projeto de software está no núcleo técnico da engenharia de software e é necessário seja qual for o modelo de processo utilizado:

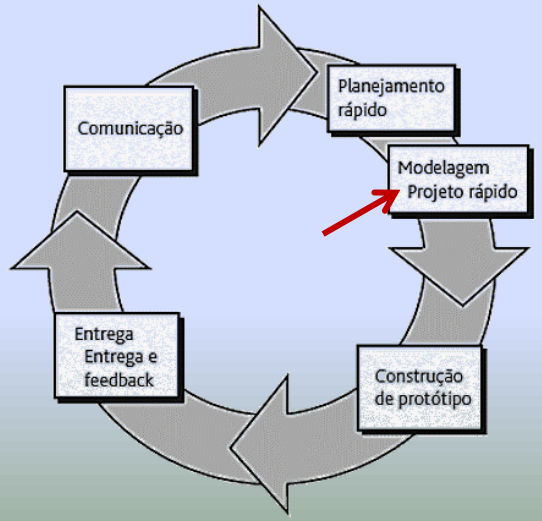
Modelo Cascata



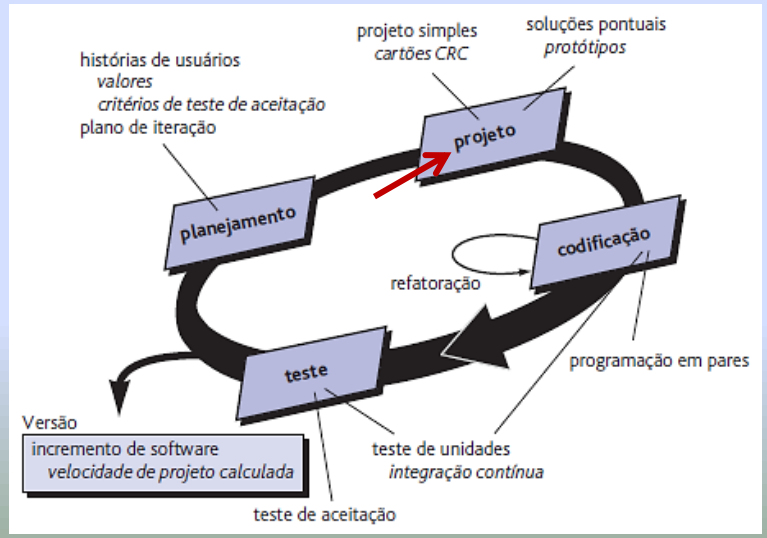
Modelo Espiral



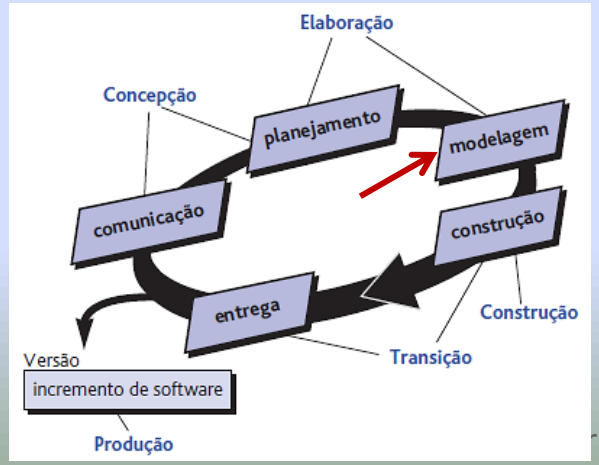
RAD – *Rapid Application Development*



XP – *Extreme Programming*

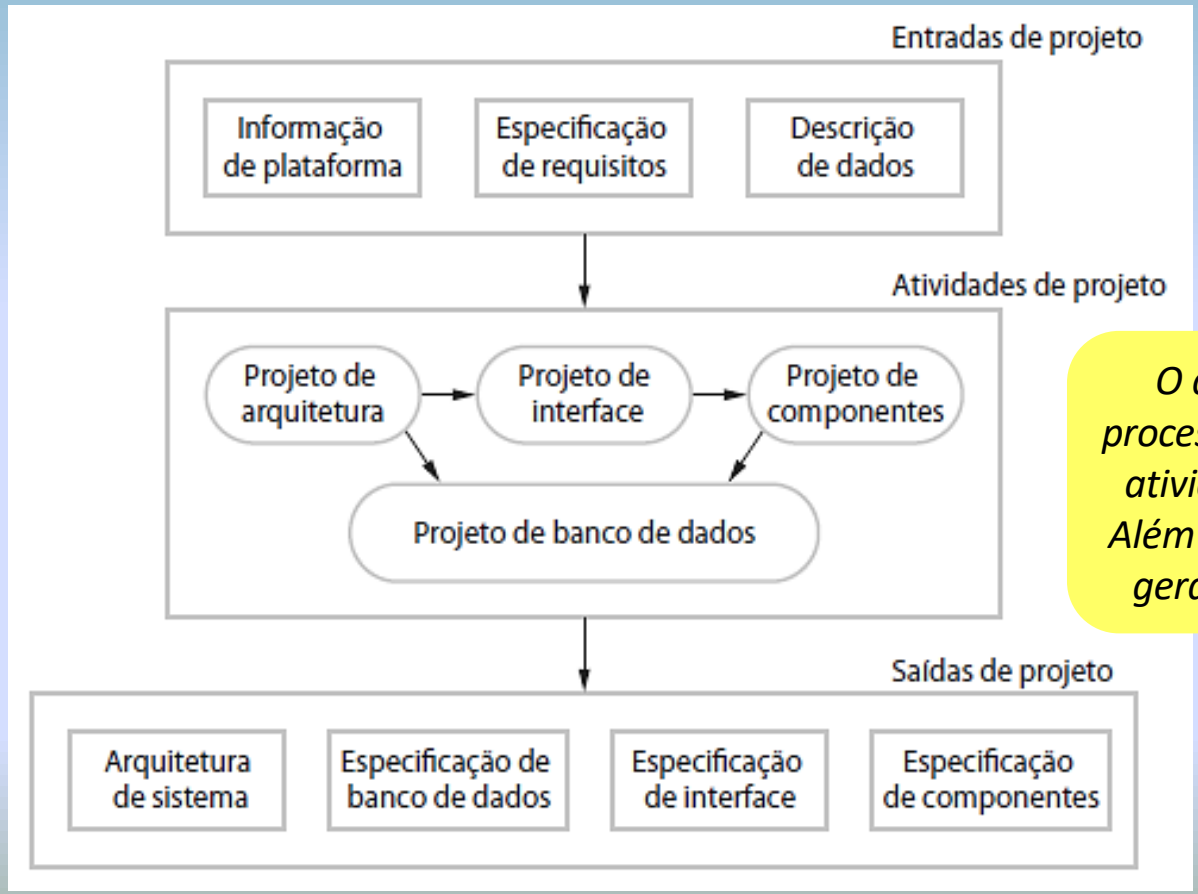


RUP – *Rational Unified Process*



Processo de Projeto de Software

- Modelo genérico que mostra as **entradas** para o processo de projeto de software, suas **atividades** e os documentos produzidos como **saídas**.



O diagrama sugere que os estágios do processo são sequenciais. Na realidade, suas atividades normalmente são intercaladas. Além disso, a conclusão de um estágio pode gerar retrabalho em um estágio anterior.

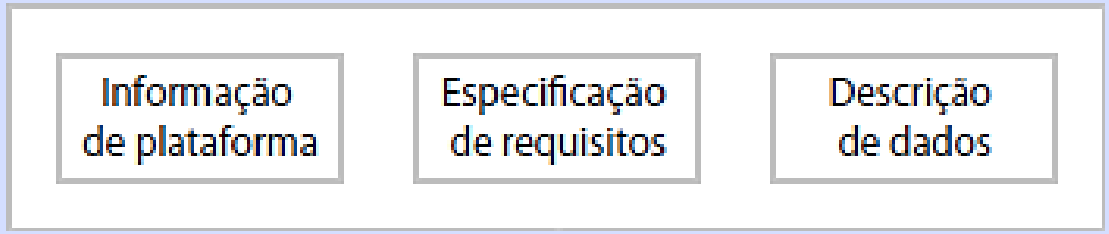
l.bertholdo@ifsp.edu.br

Processo de Projeto de Software

- Entradas do Projeto

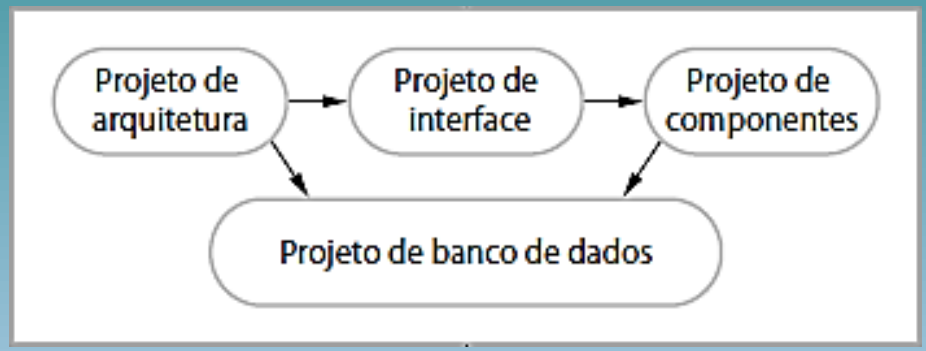
*A **plataforma** é o ambiente de software em que o sistema a ser desenvolvido será executado (sistema operacional, sistema de banco de dados, middleware etc). Essas informações são essenciais para projetar a melhor forma de integrar o software a este ambiente.*

*A **descrição dos dados** é importante para projetar como os dados do sistema devem ser estruturados e armazenados.*



*A **especificação de requisitos** descreve a funcionalidade que o software deve oferecer, bem como seus requisitos não funcionais, sendo uma das bases para projetar a arquitetura do software.*

Processo de Projeto de Software



- Atividades do Projeto

Projeto de arquitetura: *Identifica a estrutura geral do sistema, os componentes principais (algumas vezes, chamados subsistemas ou módulos), seus relacionamentos e como eles são distribuídos.*

Projeto de interface: *Define as interfaces entre os componentes do sistema. Com uma especificação de interface precisa e eficiente, um componente pode ser usado de maneira que outros componentes não precisam saber como ele é implementado.*

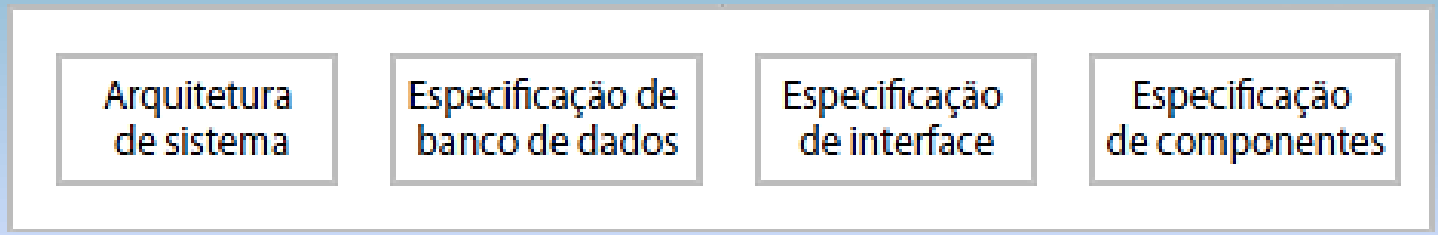
Projeto de componente: *Detalha o funcionamento de cada componente do sistema. Pode ser uma simples declaração da funcionalidade a ser implementada, uma lista de alterações a serem feitas em componentes ou uma descrição detalhada deles.*

Projeto de dados (ou banco de dados): *Estabelece os modelos e as estruturas de dados do sistema e como eles devem ser representados em seu repositório de dados. O projeto pode contemplar um repositório já existente ou um novo repositório a ser criado.*

l.bertholdo@ifsp.edu.br

Processo de Projeto de Software

- Saídas do Projeto



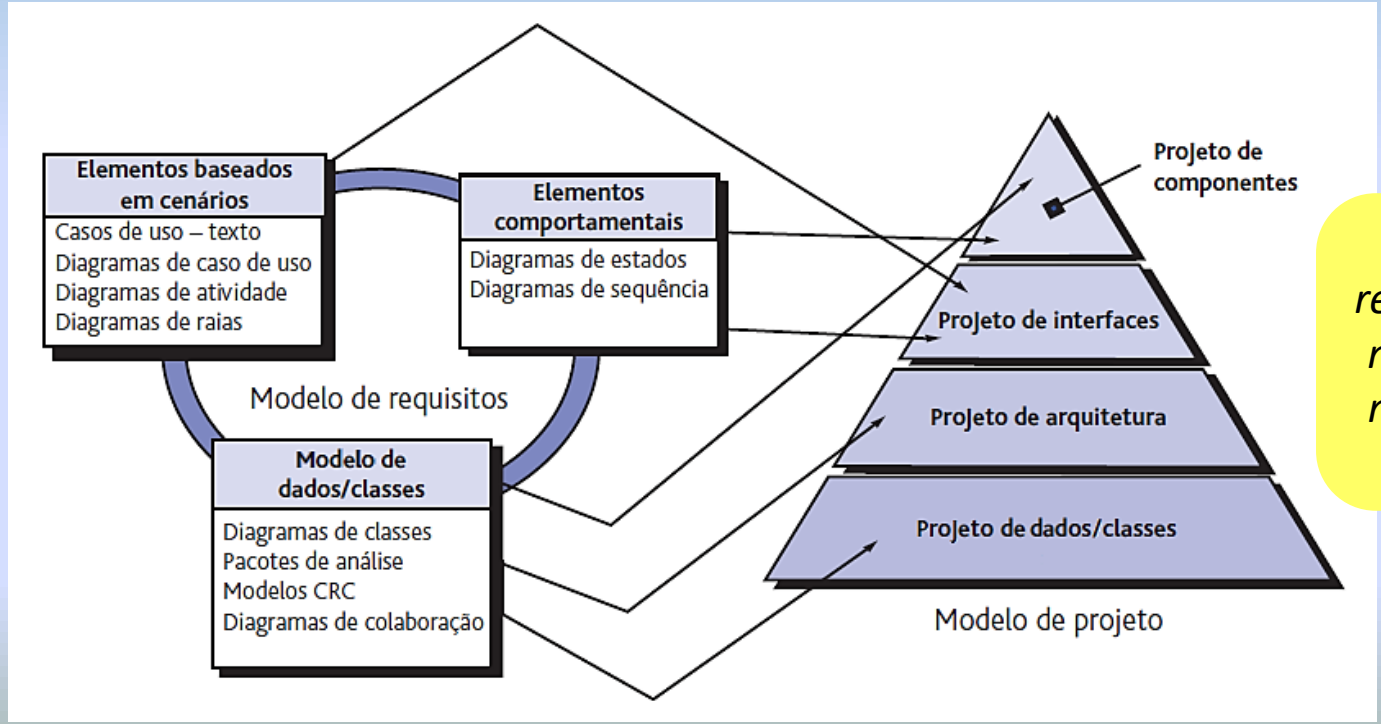
As atividades do projeto geram um conjunto de saídas, cujos detalhes e formas de apresentação podem variar bastante, dependendo do sistema a ser desenvolvido e do processo empregado.

Para sistemas críticos, devem ser produzidos documentos detalhados, com descrições completas e precisas do sistema. Em abordagens dirigidas a modelos, essas saídas podem ser majoritariamente diagramas. Já em métodos ágeis de desenvolvimento, as saídas podem não ser documentos, mas sim descrições de projeto associadas ao código do programa.

Mapeamento dos Requisitos para o Projeto

- O projeto de software se inicia assim que os requisitos tiverem sido analisados e modelados. Ele é a última atividade de modelagem e prepara o cenário para a construção (geração de código e testes).

Transformação do modelo de requisitos em modelo de projeto



Cada elemento do modelo de requisitos fornece as informações necessárias para criar os quatro modelos de projeto exigidos por uma especificação completa.

l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software

- Independentemente do método utilizado, existe um conjunto de **conceitos básicos** que devem ser considerados quando se elabora um projeto de software.

- Tais conceitos fornecem ao projetista de software uma base a partir da qual podem ser aplicados métodos de projeto mais sofisticados, que ajudarão a desenvolver software que não apenas funciona, mas funciona corretamente.

"O princípio da sabedoria para um engenheiro de software é reconhecer a diferença entre fazer um programa funcionar e fazer com que ele funcione corretamente".

***Michael A. Jackson**, cientista da computação britânico*

l.bertholdo@ifsp.edu.br

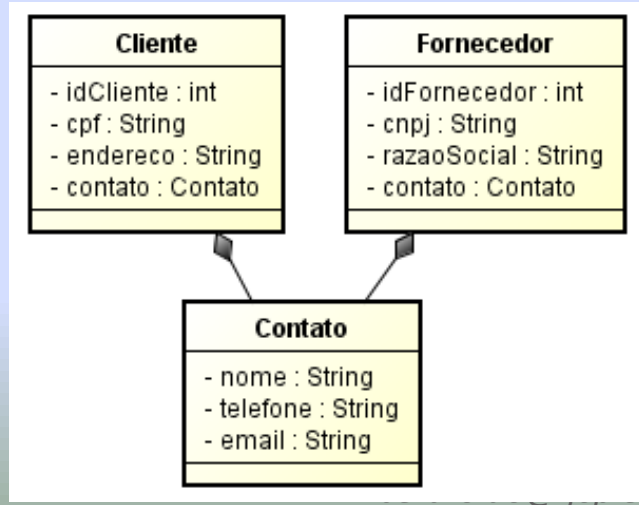
Conceitos de Projeto de Software

1. Abstração e Refinamento

- As soluções propostas em um projeto de software podem ser descritas em diferentes **níveis de abstração**. Essa característica é importante para lidar com a **complexidade de forma gradual**, por meio de refinamentos sequenciais à medida que o projeto avança.

*No nível de abstração **mais alto**, uma solução é expressa em termos abrangentes, usando a linguagem do domínio do problema. **Exemplo:** os dados de contato de clientes e fornecedores devem ser centralizados em uma única entidade.*

*Em níveis de abstração **mais baixos**, é fornecida uma descrição mais detalhada e técnica da solução, a qual pode ser implementada diretamente. **Exemplo:** A classe Contato deve ter um relacionamento de composição com as classes Cliente e Fornecedor, conforme o diagrama de classes.*



Conceitos de Projeto de Software

2. Arquitetura
   - A arquitetura é a estrutura ou a organização de componentes de software (módulos), a maneira como esses componentes interagem e a estrutura de dados que são usados pelos componentes.
   - Uma das principais metas do projeto de software é especificar a arquitetura do sistema. Para isso, existe um conjunto de **padrões de arquitetura** que podem ser reusados nos mais diferentes projetos.
3. Padrões de Projeto (*Design Patterns*)
   - Além de padrões de arquitetura, existem os **padrões de projeto**, que descrevem uma solução destinada a resolver tipos específicos de problema de projeto, que ocorrem em determinados contextos.

l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software

4. Separação por Interesses

- Este conceito sugere que qualquer problema complexo pode ser tratado mais facilmente se for subdividido em trechos a serem resolvidos e/ou otimizados independentemente.

*Nesse contexto, **interesse** refere-se a uma característica ou comportamento especificado no modelo de requisitos do software.*

- A percepção sobre a complexidade de dois problemas, quando estes são combinados, geralmente é maior do que soma da complexidade percebida quando cada um deles é analisado separadamente.

Problema A
Complexidade: **6**
+
Problema B
Complexidade: **8**
→ Complexidade: **14**

Problema A + Problema B → Complexidade: **18**

- Isso leva a uma estratégia **dividir-para-conquistar**, pois é mais fácil resolver um problema complexo quando o subdividimos.

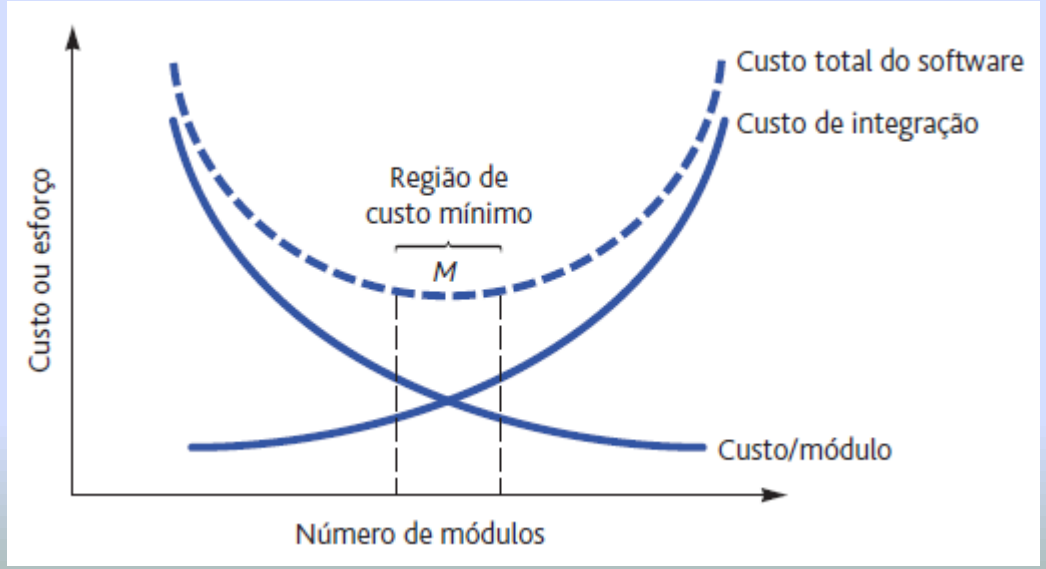
l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software

5. Modularidade

- Manifestação mais comum da separação por interesses. Em quase todos os casos, devemos dividir o projeto em vários módulos para facilitar a compreensão e, consequentemente, reduzir o custo necessário para construir o software.

Modularidade X Custos/Esforços



Subdividir o software indefinidamente não significa que o custo/esforço exigido para desenvolvê-lo será cada vez menor.

O custo para desenvolver os módulos individualmente de fato diminui à medida que o número de módulos cresce. Por outro lado, à medida que esse número aumenta, o custo para integração dos módulos também cresce.

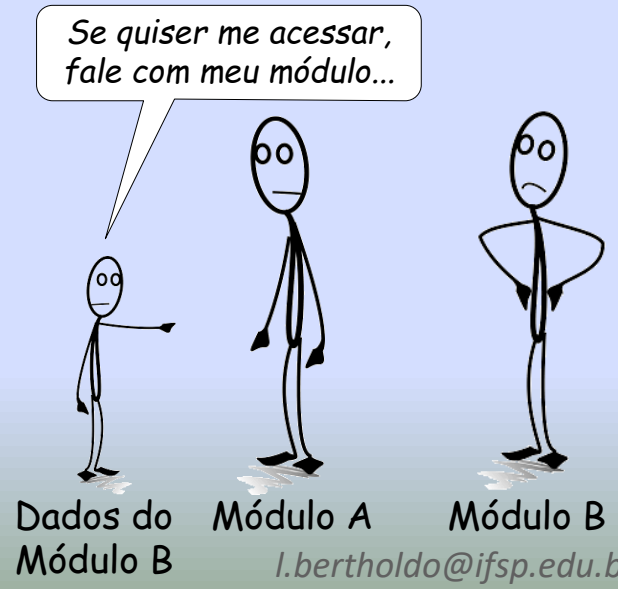
*Essas características levam a um **custo total**. Existe um número **M** de módulos que resultaria em um **custo total mínimo**. É nessa região **M** que a modularização deve se manter.*

Conceitos de Projeto de Software

6. Encapsulamento de Algoritmos e Dados

- Uma técnica que pode ajudar de forma significativa a obter um conjunto de módulos (modularidade) eficaz é o **encapsulamento**.
- Ele sugere que os módulos de software devem ser projetados de forma que **os algoritmos e os dados** contidos em um módulo sejam inacessíveis por outros módulos, disponibilizando apenas aquilo que é realmente essencial a estes módulos.

*O **encapsulamento** fornece benefícios quando são necessárias modificações durante a manutenção do software. Ao aplicá-lo, a maioria dos algoritmos e dos dados de um módulo **não são acessados diretamente** por outras partes do software. Com isso, eventuais erros introduzidos pelas modificações em um módulo têm menor chance de se propagarem para outros módulos.*



l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software

7. Independência Funcional

- Este conceito é baseada em dois critérios qualitativos: **coesão** e **acoplamento**.
- A **coesão** indica a robustez funcional de um módulo. Um módulo coeso realiza poucas tarefas, exigindo pouca interação com outros componentes do sistema.
- O **acoplamento** indica a interdependência entre os módulos e deve ser o menor possível. Módulos independentes resultam em software mais fácil de ser compreendido e menos sujeito à "reação em cadeia", em que erros pontuais que se propagam por todo o sistema.

*A **independência funcional** é a chave para um bom projeto de software, e um bom projeto é a chave para a qualidade de um software.*

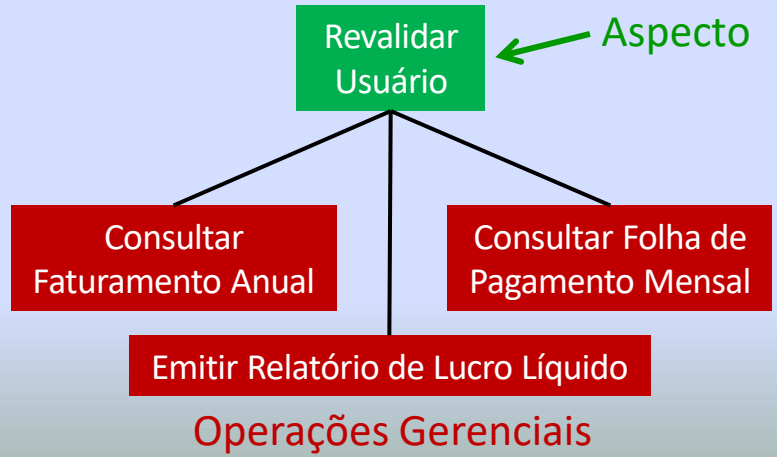
l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software

8. Aspectos

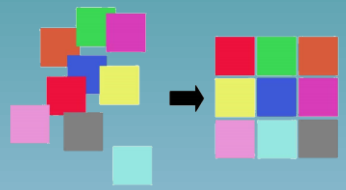
- Um modelo de requisitos deve ser organizado de modo a agrupar **interesses** (requisitos, casos de uso, características, estruturas de dados etc) para que possam ser considerados de forma conjunta.
- Porém, quando um interesse abrange grande parte do sistema, ele não deve fazer parte de vários grupos de interesse. Esse tipo de interesse é chamado de **aspecto** e deve ser tratado de forma isolada.

*É importante identificar **aspectos** de modo que o projeto possa acomodá-los apropriadamente à medida que ocorrem o **refinamento** e a **modularização**. Idealmente, um aspecto é implementado como um **módulo (componente)** separado, em vez de trechos de software "espalhados" ou "emaranhados" em vários componentes.*



l.bertholdo@ifsp.edu.br

Conceitos de Projeto de Software



9. Refatoração

- Atividade sugerida por diversos métodos ágeis que consiste em mudar um software de tal forma que não altere o comportamento externo do seu código, embora melhore sua estrutura interna.

Quando um software é refatorado, o projeto existente é examinado em termos de redundância, elementos não utilizados, algoritmos ineficientes, estruturas de dados mal construídas, entre outras falhas. Por exemplo, a primeira iteração de um projeto poderia gerar um componente com baixa coesão (realizar três funções com pouca relação entre si). Nesse caso, talvez seja interessante refatorá-lo em três componentes, cada um apresentando alta coesão.

10. Análise e Projeto Orientado a Objetos

- Conceito amplamente utilizado em projetos de software modernos, que envolvem elementos como classes, objetos, mensagens, composição, herança, polimorfismo, entre outros.

Conceitos de Projeto de Software

11. Classes de Projeto

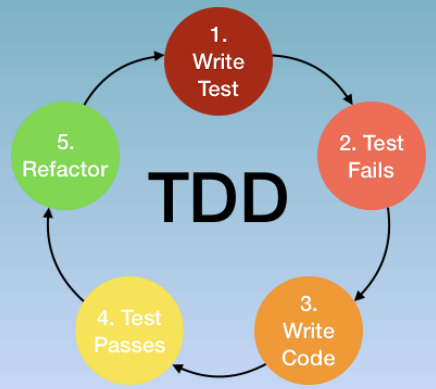
- O modelo de requisitos (ou análise) define um conjunto de **classes de análise**. Cada uma delas descreve, com um nível de abstração relativamente alto, algum elemento do domínio do problema.
- À medida que o projeto evolui, essa abstração diminui, e cada classe de análise é transformada em um conjunto de **classes de projeto**, as quais já têm os detalhes necessários para serem implementadas.

Tipos de classes de projeto que podem ser gerados, cada um retratando uma camada da arquitetura:

- ***Classes de interfaces do usuário*** *definem todas abstrações para a interação humano-computador.*
- ***Classes de domínio de negócio*** *identificam os atributos e serviços (métodos) necessários para implementar algum elemento do domínio de negócio, definido por uma ou mais classes de análise.*
- ***Classes de processos*** *implementam as abstrações de baixo nível necessárias para a completa gestão das classes de domínio de negócio.*
- ***Classes persistentes*** *representam repositórios cujos dados persistirão depois da execução do código.*
- ***Classes de sistema*** *implementam funções de gerenciamento e controle de software que permitem ao sistema operar e comunicar em seu ambiente computacional e com o mundo exterior.*

Conceitos de Projeto de Software

12. Projeto para Teste



- Atualmente, com o amplo uso do **desenvolvimento orientado a testes (*Test-Driven Development*, TDD)**, muitos dos testes de desenvolvimento são feitos antes de implementar o código.
- Considerando o emprego do TDD, como o projeto de software ocorre antes dos testes e da implementação em si, o projeto deve levar em conta locais adequados onde seja possível inserir código de teste que investigue o software em execução.

Esses "locais para código de teste" devem ser projetadas no nível dos componentes. Para isso, o projetista deve pensar nos testes que serão realizados para cada componente. Em resumo, é preciso projetar o software de modo que o código de teste seja contemplado na estrutura do projeto.

Referências

- PRESSMAN, Roger S.; MAXIM, Bruce R. **Engenharia de software:** uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

- SOMMERVILLE, Ian. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011.