Introdução ao Design

João Arthur e Guilherme Germoglio

Coordenação de Pós-graduação em Informática - COPIN

16/10/2008

Roteiro

Objetivos

- Introduzir os conceitos de design
- Apresentar design como processo e como produto
- Diferenciar design de baixo nível de design de alto nível
- Apresentar as principais técnicas de design

O que é design?

- Duas visões (Taylor e Van der Hoek [5]):
  - Nome
  - Verbo
- Stroustrup [4]: “Design is the end product of the design process.”

O que é design?

- Design surge de uma necessidade: ministrar uma aula sobre design
- Problema: Guiga está sobrecarregado
- Solução: Terceirizar
- O que o cliente (Guiga) quer: Uma aula bem dada sobre o que é design e o que é design de software.

O que é design?

- Design surge de uma necessidade: ministrar uma aula sobre design
- Problema: Guiga está sobrecarregado
- Solução: Terceirizar
- O que o cliente (Guiga) quer: Uma aula bem dada sobre o que é design e o que é design de software.

O que é design?

- Design surge de uma necessidade: ministrar uma aula sobre design
- Problema: Guiga está sobrecarregado
- Solução: Terceirizar
- O que o cliente (Guiga) quer: Uma aula bem dada sobre o que é design e o que é design de software.

O que é design?

- Design surge de uma necessidade: ministrar uma aula sobre design
- Problema: Guiga está sobrecarregado
- Solução: Terceirizar
- O que o cliente (Guiga) quer: Uma aula bem dada sobre o que é design e o que é design de software.

Por que projetar?

- Por que não fazer a aula diretamente?
- Risco
- Temos que projetar!
- Temos que avaliar o projeto!

Por que projetar?

- Por que não fazer a aula diretamente?
- Risco
- Temos que projetar!
- Temos que avaliar o projeto!

Por que projetar?

- Por que não fazer a aula diretamente?
- Risco
- Temos que projetar!
- Temos que avaliar o projeto!

Requisitos

- 40 alunos
- Mensagens a serem transmitidas (conteúdo)
- Duração de no máximo 2 horas

Requisitos

- 40 alunos
- Mensagens a serem transmitidas (conteúdo)
- Duração de no máximo 2 horas

Requisitos

- 40 alunos
- Mensagens a serem transmitidas (conteúdo)
- Duração de no máximo 2 horas

Requisitos

- 40 alunos
- Mensagens a serem transmitidas (conteúdo)
- Duração de no máximo 2 horas

Possíveis Soluções

- Alternativas para resolução do problema:
  - Aula ao ar livre
  - Aula prática
  - Aula virtual
  - Aula “tradicional”

Análise das possíveis soluções

- Analisar restrições
  - CD-105 ou Hattori
  - Tempo
  - Conhecimento interno sobre o assunto (viabilidade)
  - Custo
  - Nível dos alunos
- Escolher solução
- Escolher representação

Resultado

- Template Slides:
  - Seções
  - Relação entre as Seções
  - Fontes
  - Figuras
  - Mensagens
  - Exemplos
- Metodologia a ser utilizada

Benefícios

- Avaliação prévia
- Facilita comunicação
- Facilita implementação

Observações

- Embora seja mais barato que fazer o artefato, planejar não é barato.
- O resultado final não é a aula, mas a sua descrição (plano de aula)!

Software Design

- O foco é software
- Geração de modelos
- Diretrizes para implementação

Caraterísticas de Software

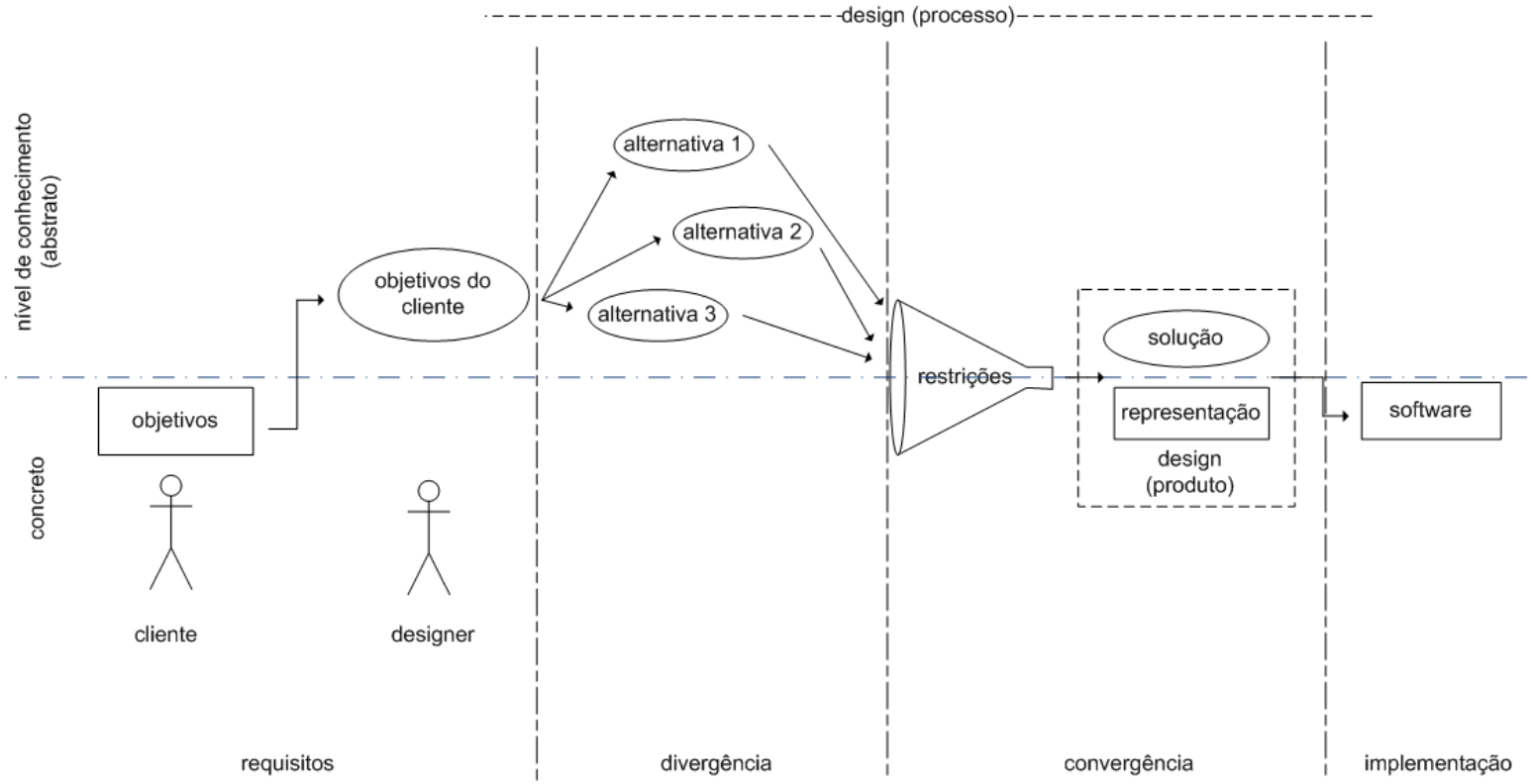
Brooks [1] identifica quatro propriedades:

- Complexidade: muitos estados durante a execução
- Conformidade: estar em conformidade com padrões, hardware, software e outros componentes
- *Changeability*: sofrer constantes mudanças
- Invisibilidade: não há representação visual

Como um nome

- Aspectos de representação do sistema [2]:
  - Estrutura do software (hierarquia, subprogramas etc)
  - Algoritmos
  - Estrutura de pacotes (organização das unidades de compilação)
  - Interação entre os módulos

Como uma atividade



Como uma atividade

- Belady [3]:
  - Diversificação: Geração das alternativas
  - Convergência: Seleção das alternativas que satisfazem objetivos e restrições

Fases

- Objetivos
- Restrições
- Alternativas
- Representações
- Soluções

Técnicas

- Diretrizes
- Guias
- Noções de boas práticas que podem resultar em um bom design
- *Design Patterns*

Abstração

- Eliminar complexidade
- Focar no que realmente importa
- Interfaces e camadas
- Requer habilidade e experiência. Quanto abstrair?
- Benefícios: facilita comunicação, análise e entedimento

Ocultação de Informação

- Encapsulamento
- Esconder detalhes de implementação
- Benefícios: redução de acoplamento

Modularização

- Decompor em módulos o sistema
- *Packaging*
- Benefícios: facilita implementação, entendimento, diminui acoplamento etc

Separação de interesses

- Modularizar interesses
- Exemplo clássico: *Log*

Acoplamento e Coesão

- Acoplamento: entre módulos
  - Evitar
  - Difícil de entender
  - Difícil de mudar
  - Indício para melhorar a modularização
- Coesão: Quão relacionada está a classe com sua responsabilidade?

Coincidental

```
class Angu {
  public static int acharPadrão(String texto, String padrão) {
    // ...
  }
  public static int média(Vector números) {
    // ...
  }
  public static outputStream abreArquivo(string nomeArquivo) {
    // ...
  }
}
```

Lógica

```
public void faça(int flag) {
  switch(flag) {
  case ON:
    // coisas para tratar de ON
    break;
  case OFF:
    // coisas para tratar de OFF
    break;
  case FECHAR:
    // coisas para tratar de FECHAR
    break;
  case COR:
    // coisas para tratar de COR
    break;
  }
}
```

Temporal

```
procedure inicializaDados() {
  font = "times";
  windowSize = "200,400";
  xpto.nome = "desligado";
  xpto.tamanho = 12;
  xpto.localização = "/usr/local/lib/java";
}
```

Outros tipos

- Procedural: Tarefas que ocorrem em ordem (sozinhas não fazem sentido)
- Comunicação: Compartilhamento de dados
- Sequencial: compartilham dados de entrada e saída (pipe)
- Funcional: A melhor! Objetos representam um único conceito

Separação entre política e implementação

- Um módulo deve lidar com política ou implementação
- Módulos de implementação somente devem ser responsáveis por executar algoritmos
- Benefícios: Facilita reuso e manutenção dos módulos de implementação
- Qual o padrão para esta técnica?

Separação de interface e Implementação

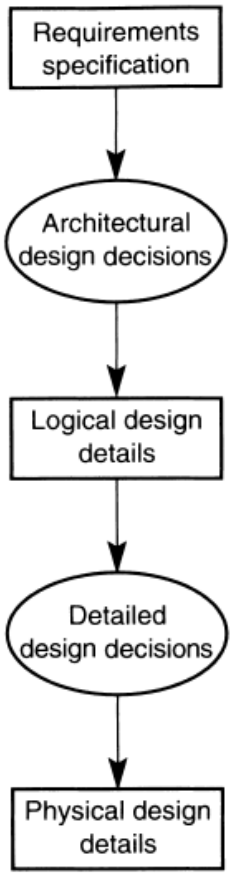
- Contratos
- O Que? X Como?
- Benefícios: redução de acoplamento entre clientes e módulos

Comparação

- Alto nível: Fase de refinamento da arquitetura
  - Definição de módulos
  - Interação entre os módulos
  - Bibliotecas a serem utilizadas
  - Paradigma de persistência
  - Atendimento a requisitos de qualidade
- Baixo nível
  - Definição dos objetos e suas responsibilidades
  - Questões de implementação: concorrência, tratamento de falhas, definição de esquema de BD etc

Quando acontece?

- David Budgen



Conclusões

- Design é processo
- Design é descrição
- Projetar não é barato
- Exige experiência (boas e ruins)
- Reduz riscos


F. Brooks.
No Silver Bullet: Essence and Accidents of Software Engineering.
IEEE Computer, 20(4):10–19, 1987.


D. Budgen.
Software Design.
Addison Wesley, 2003.


L. Peters.
In Forward to Software Design: Methods and Techniques, 1981.


B. Stroustrup and A. Publishing.
The C+ Programming Language.
IBM SYSTEMS JOURNAL, 31(4), 1992.


R. Taylor and A. van der Hoek.
Software design and architecture: The once and future focus of software engineering.

Future of Software Engineering, 2007.