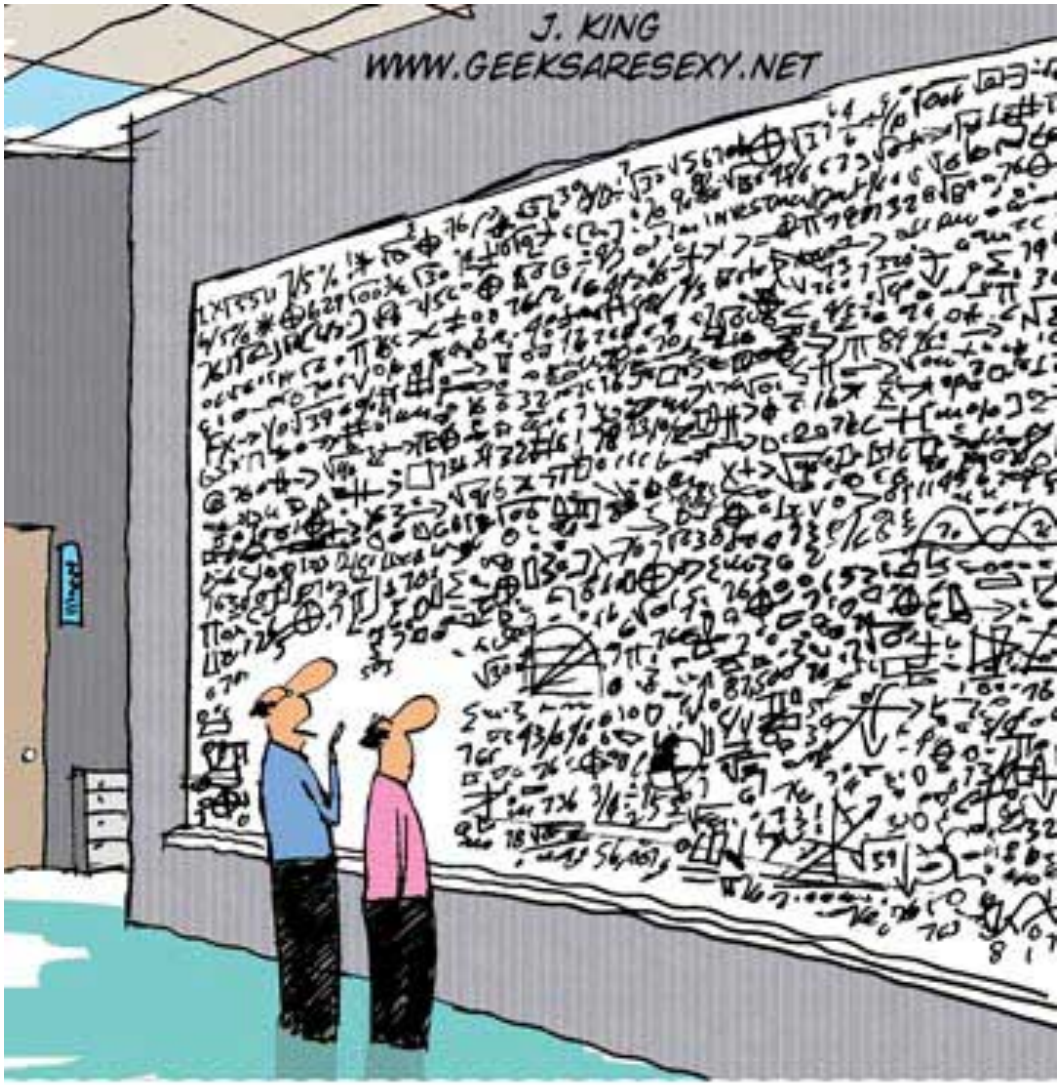
ARQUITETURAS E PADRÕES DE SOFTWARE

HUGO PAREDES

engenharia de software

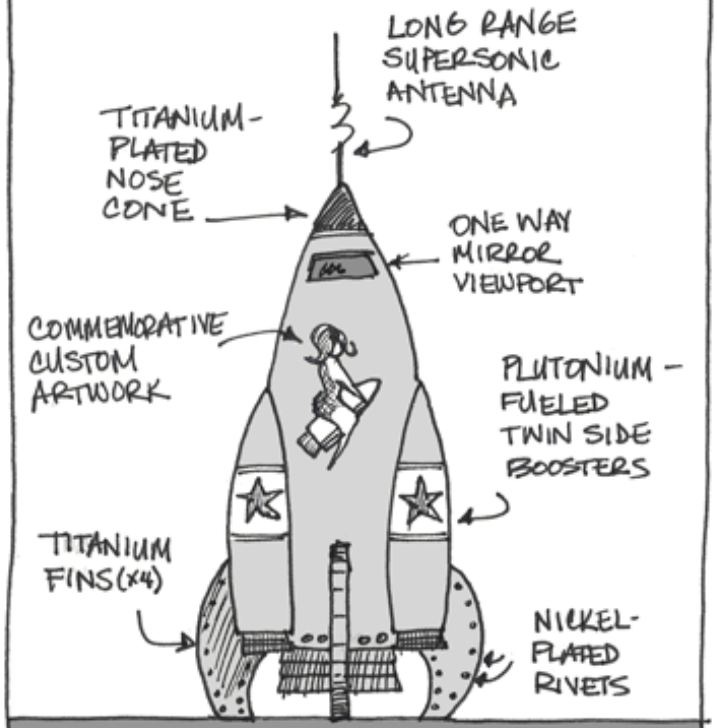
motivação



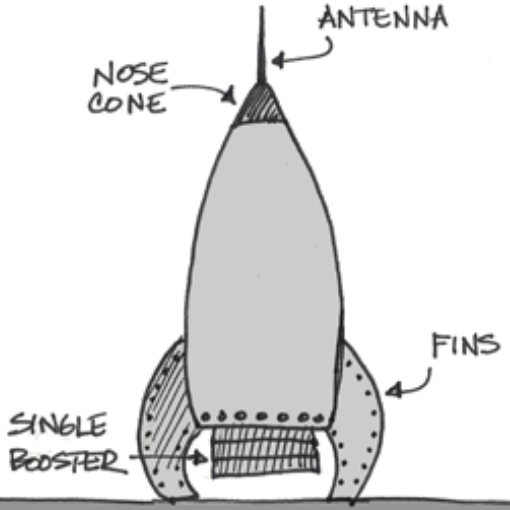
"...And that, in simple terms, is what's wrong with your software design."

software design

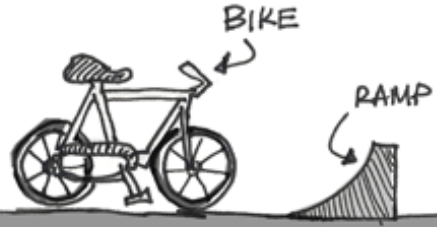
WHAT WE DREAM UP AT KICKOFF



WHAT WE SETTLE FOR AT LAUNCH



WHAT THE USER NEEDS



BONUS 2015

arquiteturas de software

software design patterns

arquiteturas e padrões de software

objetivos e programa

objetivos

- Identificar a necessidade e a oportunidade de reutilização de soluções padronizadas para problemas típicos no desenvolvimento de software
- Compreender os conceitos associados a arquitecturas de software, estilos arquitecturais e padrões de software
- Associar o processo de desenvolvimento de software à sua arquitectura e as respectivas correlações
- Estudar os padrões de software mais conhecidos
- Perante um problema específico, identificar os padrões que se adequam à sua resolução e implementá-los

programa (1)

- Desenho de software
  - Conceitos gerais
  - Princípios de desenho
  - Abordagens fundamentais
- Arquitecturas de Software
  - Conceitos
  - Estilos arquitecturais
  - Modelos de referência
  - A arquitectura no desenvolvimento de software
- Introdução aos Design Patterns
  - Conceitos gerais
  - Catalogação de Design Patterns
  - Metodologia de projecto usando Design Patterns

programa (2)

- Padrões de criação
  - Abstract Factory
  - Builder
  - Factory Method
  - Prototype
  - Singleton
- Padrões de estrutura
  - Adapter
  - Bridge
  - Composite
  - Decorator
  - Facade
  - Flyweight
  - Proxy

programa (3)

- Padrões de comportamento
  - Chain of responsibility
  - Command
  - Interpreter
  - Iterator
  - Mediator
  - Memento
  - Observer
  - State
  - Strategy
  - Template Method
  - Visitor

programa (4)

- Anti-Patterns
  - Conceitos gerais
  - Principais anti-patterns
  - Patterns vs Anti-patterns
- Refactoring
  - Conceitos
  - Refactoring to pattern

avaliação

avaliação contínua

avaliação teórica:

- avaliações conceitos (AC)
  Escolha múltipla, realizada no Kahoot durante a aula teórica. Serão realizadas pelo menos 4 avaliações de conceitos.
- 1 apresentação padrões (AP)
  Apresentação de 20 minutos, na aula teórica, apresentando um Padrão de Software do catalogo GoF.
  A apresentação será suportada por um ensaio de 6-8 páginas, formato IEEE, duas colunas.
- 1 frequência (F)
  Cenários de aplicação de conceitos abordados nas aulas, duração 60 minutos.

avaliação prática

- 2 mini-testes práticos (MT)
  Abrangendo as três tipos de padrões de software, de acordo com o catálogo GoF.

Nota Final = 15%xAC + 15%xAP + 20%xF + 50%xMT

avaliação contínua - datas

avaliação teórica

- avaliações conceitos (AC):

  abril, maio

- apresentação padrões (AP)

  28 abril - 12 maio

- frequência (F)

  2 junho

avaliação prática

- mini-testes práticos (MT)

  25 abril; 2 junho

nota mínima

<u>As notas mínimas são aplicáveis para aprovação e admissão a exame:</u>

- avaliações conceitos (AC): 6 valores
- apresentação padrões (AP): 6 valores
- frequência (F): 6 valores
- mini-testes práticos (MT): 8,5 valores

exame final

A avaliação em exame final apenas compreende a componente teórica, sendo a nota da componente prática obtida em avaliação continua ou complementar.

A componente teórica (T) inclui uma prova escrita, com duração de 120 minutos, abrangendo conceitos e cenários de aplicação.

$$\text{Nota final exame} = 50\% \times T + 50\% \times MT$$

bibliografia

“**Software architecture in practice**”. Bass, Len, Paul Clements, and Rick Kazman. Addison-Wesley, 2012.

“**Design Patterns: Elements of Reusable Object-Oriented Software**”. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides. Addison-Wesley. 1994.

“**Documenting software architectures: views and beyond**“. Clements, Paul, et al. Pearson Education, 2010.

“**Design Patterns Explained: A New Perspective on Object-Oriented Design**”. Alan Shalloway, James R. Trott. Net Objectives. 2004.

“**Design Patterns for Dummies**”. Steve Holzner. Wiley. 2006