Técnicas de Programação 1

Rodrigo Bonifácio

2017/2

Qual o nosso objetivo?

Introduzir a preocupação com o *desenho* de um software, utilizando as construções presentes na orientação a objetos (alinhadas com construções de programação funcional).

Qual o nosso objetivo?

Introduzir a preocupação com o *desenho* de um software, utilizando as construções presentes na orientação a objetos (alinhadas com construções de programação funcional).

De forma geral, programar é uma atividade que envolve um processo criativo relativamente simples e, em muitas situações, podemos conceber uma solução de forma trivial. Por outro lado, conceber boas abstrações, que levam a um software de qualidade, é uma atividade difícil e que requer experiência. O que é um software de qualidade?

- corretude e usabilidade
- tempo de resposta adequado
- fácil de entender, manter e evoluir

De forma geral, programar é uma atividade que envolve um processo criativo relativamente simples e, em muitas situações, podemos conceber uma solução de forma trivial. Por outro lado, conceber boas abstrações, que levam a um software de qualidade, é uma atividade difícil e que requer experiência. O que é um software de qualidade?

- corretude e usabilidade
- tempo de resposta adequado
- fácil de entender, manter e evoluir

De forma geral, programar é uma atividade que envolve um processo criativo relativamente simples e, em muitas situações, podemos conceber uma solução de forma trivial. Por outro lado, conceber boas abstrações, que levam a um software de qualidade, é uma atividade difícil e que requer experiência. O que é um software de qualidade?

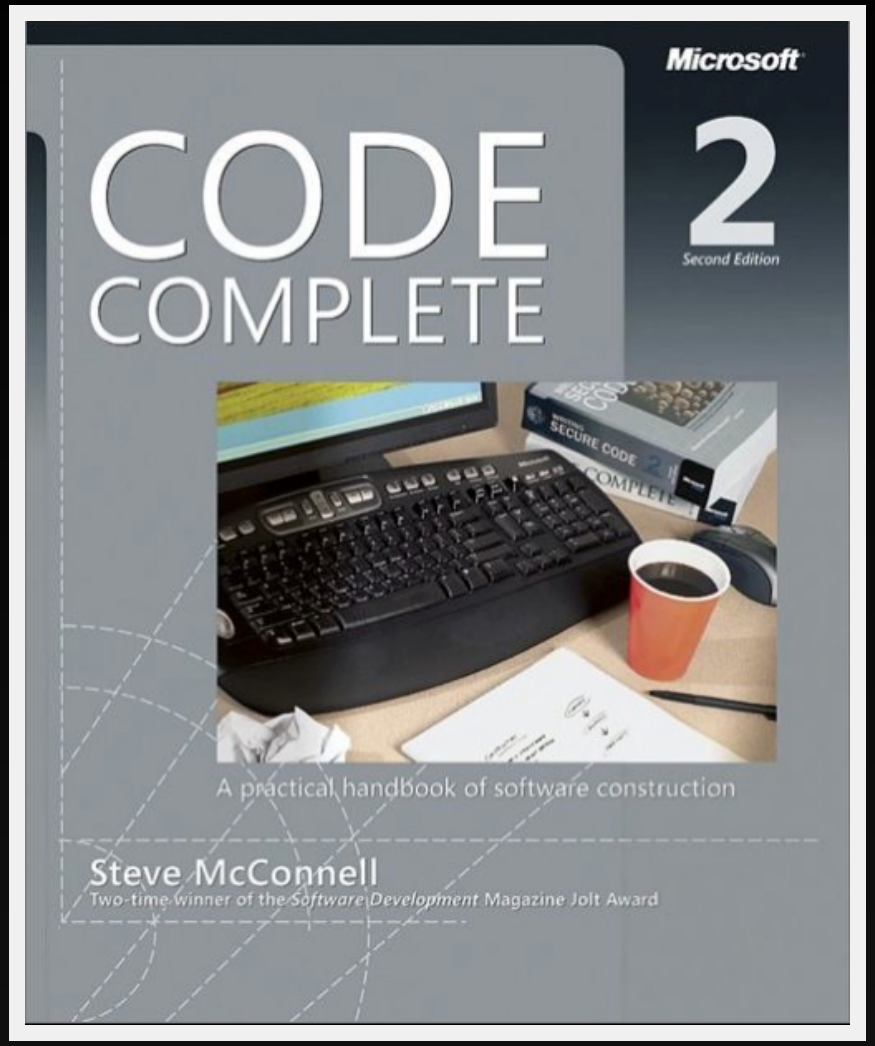
- corretude e usabilidade
- tempo de resposta adequado
- fácil de entender, manter e evoluir

Durante a formação de um desenvolvedor, o mais importante é compartilhar técnicas de resolução de problemas e de decomposição de software de forma adequada. Isso é o que traz diferença competitiva em relação à formação.

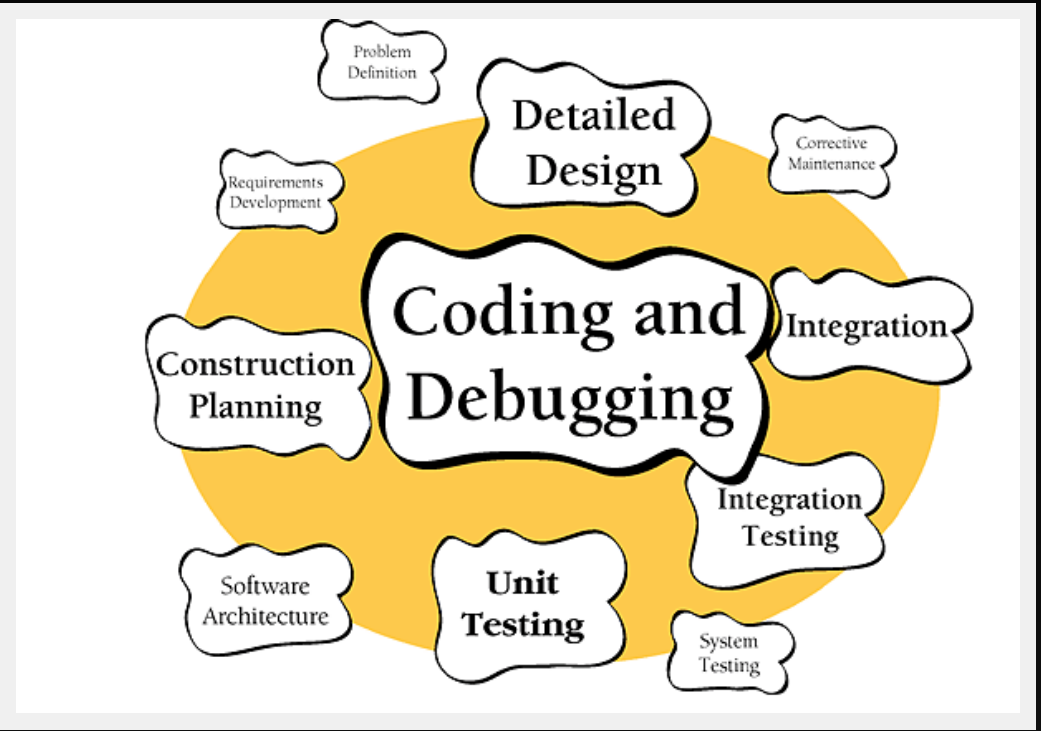
O custo de um desenvolvedor de software na Índia equivale a um terço do custo de um desenvolvedor de software no Brasil (Geraldo Gomes, Dell Computadores do Brasil, CSBC 2014).

Precisamos formar pessoas que produzam software com mais produtividade e mais qualidade. Não basta estar correto, o software tem que ser fácil de entender, manter e evoluir.

Precisamos formar pessoas que produzam software com mais produtividade e mais qualidade. Não basta estar correto, o software tem que ser fácil de entender, manter e evoluir.


Microsoft
CODE
COMPLETE
2
Second Edition
SECURE CODE
A practical handbook of software construction
Steve McConnell
Two-time winner of the Software Development Magazine Jolt Award

Construção de software



Construção de software envolve o *desenho detalhado*, que objetiva identificar as abstrações de (parte de) um software. As abstrações (pacotes, classes, classes abstratas em Java, Scala, C++ etc.) devem favorecer a modularidade de um software e as possibilidades de reuso.

Ferramentas

- esboço em quadro branco
- desenho dirigido a testes
- diagramas UML
- linguagens de especificação formais

On small, informal projects, a lot of design is done while the programmer sits at the keyboard. "Design" might be just writing a class interface in pseudocode before writing the details. It might be drawing diagrams of a few class relationships before coding them. It might be asking another programmer which design pattern seems like a better choice. Regardless of how it's done, small projects benefit from careful design just as larger projects do, and recognizing design as an explicit activity maximizes the benefit you will receive from it.

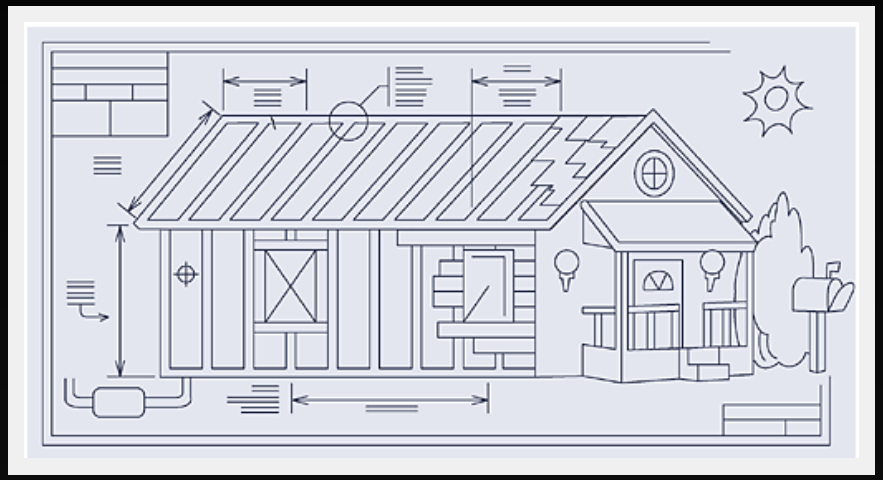
Design is the activity that links requirements to coding and debuging.

If you build a software using the wrong abstractions, it would be hard to code, hard to test, hard to debug, . . .

Daniel Jackson

O esforço relacionado com o desenho do software depende da aplicação, da mesma forma que o esforço com o desenho da *construção* de uma casa depende da *complexidade* da casa.


?
FIDO
FIDO



O objetivo do desenho detalhado é evitar o retrabalho (identificando boas abstrações), aumentar a produtividade (reuso) e favorecer as atividades de manutenção e evolução de software (flexibilidade).

O que isso tem a ver com TP1 e POO?

As unidades de modularização e as estratégias de (de)composição de módulos na programação orientada a objetos favorecem a identificação de boas abstrações de software, favorecem o reuso do software e favorecem a flexibilidade do software. Por outro lado, as construções do paradigma imperativo, do qual POO faz parte, não favorecem a construção de programas concorrentes— que se beneficiam de algumas características presentes tipicamente em linguagens funcionais (como Haskell).

O que isso tem a ver com TP1 e POO?

As unidades de modularização e as estratégias de (de)composição de módulos na programação orientada a objetos favorecem a identificação de boas abstrações de software, favorecem o reuso do software e favorecem a flexibilidade do software. Por outro lado, as construções do paradigma imperativo, do qual POO faz parte, não favorecem a construção de programas concorrentes— que se beneficiam de algumas características presentes tipicamente em linguagens funcionais (como Haskell).

Desafios relacionados com o desenho de um software

- design is a wicked problem
- design is about tradeoffs and priorities
- design is a heuristic process

Como lidar com esses desafios?

- gerenciar a complexidade do software
  - buscar maior grau de abstração
  - focar no domínio, em vez dos detalhes de implementação
  - usar técnicas como dividir para conquistar e separação de preocupações
- identificar objetos do mundo real (Bertand Meyer)
- utilizar abstrações consistentes
- encapsular detalhes de implementação
- identificar preocupações que podem ser alteradas
- pesquisar boas soluções (padrões de projeto)
- formalizar o contrato das classes

Como lidar com esses desafios?

- gerenciar a complexidade do software
  - buscar maior grau de abstração
  - focar no domínio, em vez dos detalhes de implementação
  - usar técnicas como dividir para conquistar e separação de preocupações
- identificar objetos do mundo real (Bertand Meyer)
- utilizar abstrações consistentes
- encapsular detalhes de implementação
- identificar preocupações que podem ser alteradas
- pesquisar boas soluções (padrões de projeto)
- formalizar o contrato das classes

Como lidar com esses desafios?

- gerenciar a complexidade do software
  - buscar maior grau de abstração
  - focar no domínio, em vez dos detalhes de implementação
  - usar técnicas como dividir para conquistar e separação de preocupações
- identificar objetos do mundo real (Bertand Meyer)
- utilizar abstrações consistentes
- encapsular detalhes de implementação
- identificar preocupações que podem ser alteradas
- pesquisar boas soluções (padrões de projeto)
- formalizar o contrato das classes

Existem duas formas para conceber o desenho de um software: uma estratégia é fazer com que o software seja tão simples que obviamente não existem deficiências; a outra é fazer com que o software seja tão complexo que não existem deficiências óbvias (fáceis de serem identificadas).

C R Hoare

Elegance is not a dispensable luxury but a factor that decides between success and failure.

Edsger Dijkstra

When software-project surveys report causes of project failure, they rarely identify technical reasons as the primary causes of project failure. Projects fail most often because of poor requirements, poor planning, or poor management. But when projects do fail for reasons that are primarily technical, the reason is often uncontrolled complexity. The software is allowed to grow so complex that no one really knows what it does. When a project reaches the point at which no one completely understands the impact that code changes in one area will have on other areas, progress grinds to a halt.

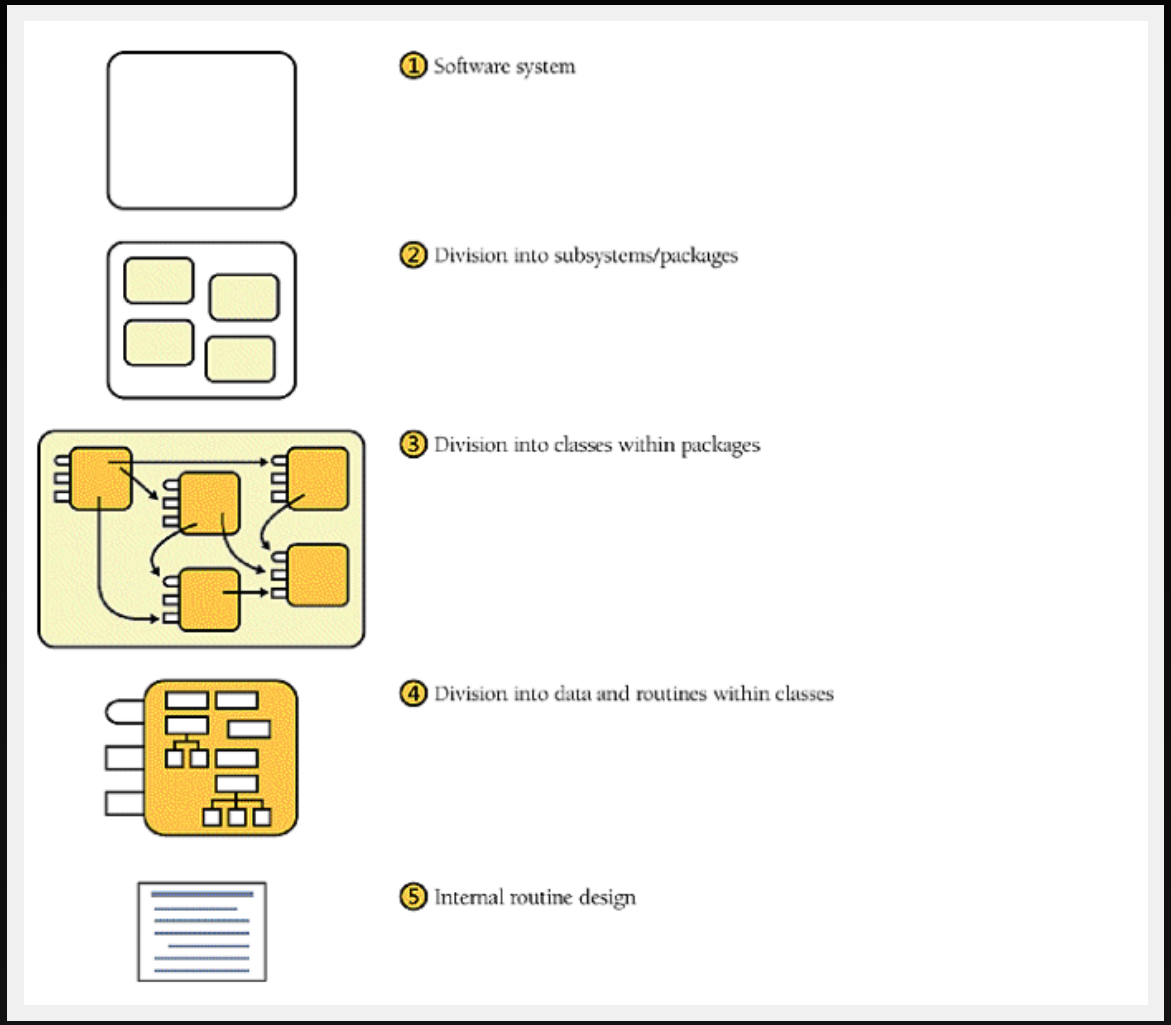
Características de um bom design

- simplicidade
- manutenibilidade
- extensibilidade
- reusabilidade
- portabilidade
- baixo acoplamento

When I am working on a problem, I never think about beauty but when I have finished, if the solution is not beautiful, I know it is wrong.

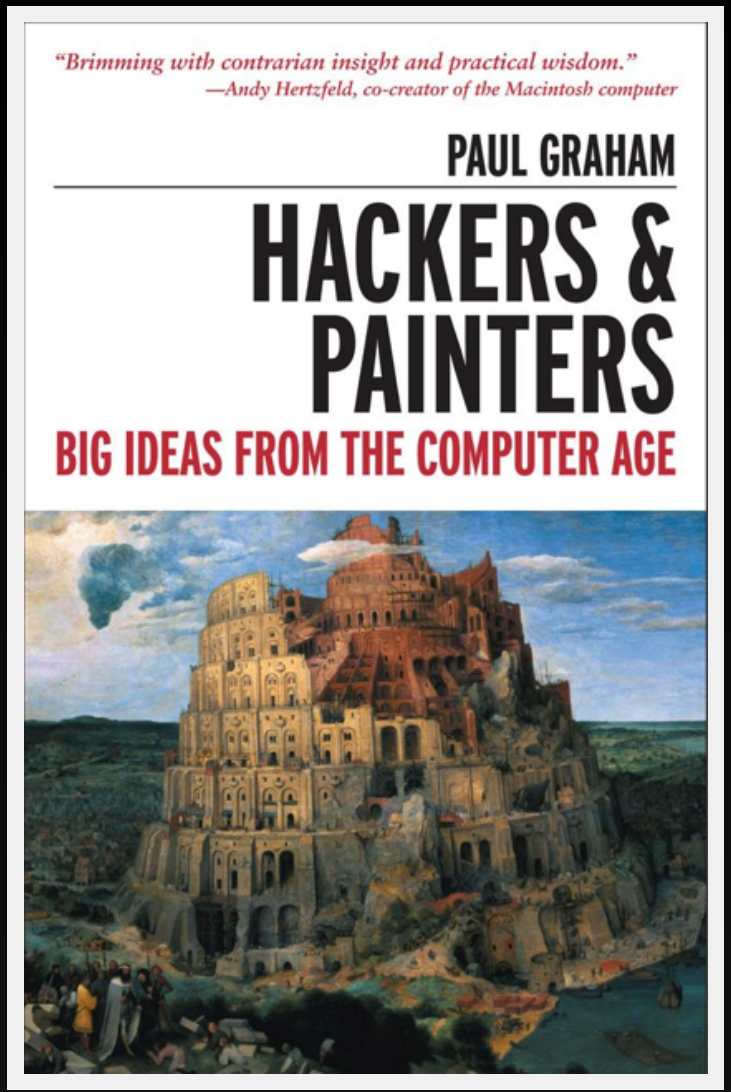
R. Buckminster Fuller

Diferentes níveis de design



Mas esse curso é de programação. . . ,
então precisamos formar hackers.

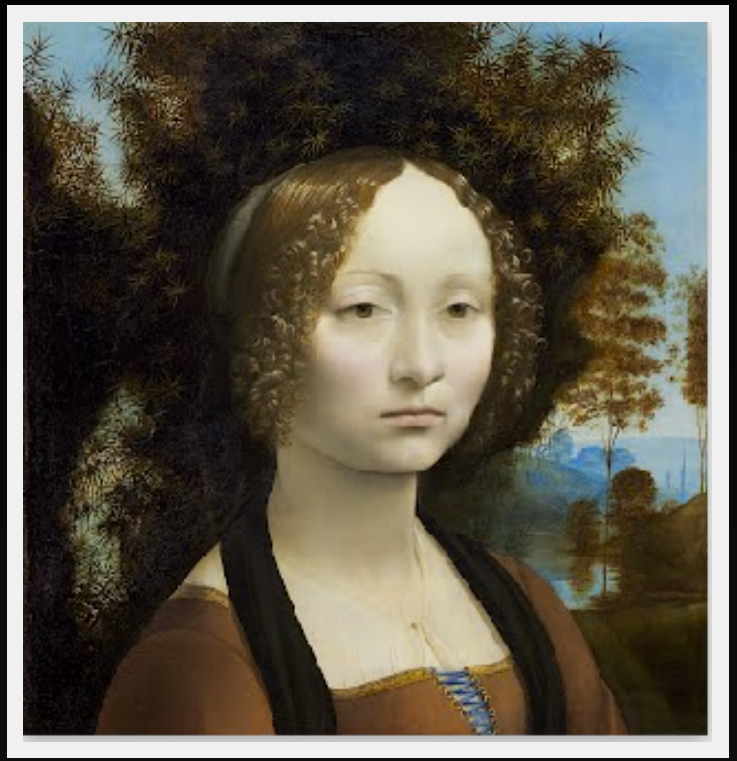
Mas esse curso é de programação. . . ,
então precisamos formar hackers.


"Brimming with contrarian insight and practical wisdom."
—Andy Hertzfeld, co-creator of the Macintosh computer
PAUL GRAHAM
HACKERS & PAINTERS
BIG IDEAS FROM THE COMPUTER AGE

What hackers and painters have in common is that they're both makers. Along with composers, architects, and writers, what hackers and painters are trying to do is make good things.

Paul Graham

. . . ou seja, temos que nos preocupar com os detalhes, da mesma forma que os grandes artistas, arquitetos e engenheiros.



Ginevra de' Benci (Leonardo da Vinci)

Aprender ~~C++~~, ~~Java~~, Scala (ou qualquer outra linguagem OO) não é o mais importante nessa disciplina. O mais importante é saber projetar um software utilizando boas abstrações e ter a preocupação com as características de um bom desenho OO + funcional. Isso requer o balanceamento entre a completude do projeto da disciplina com a qualidade da solução implementada. O desenho deve ser refletido em uma implementação.

Aprender ~~C++~~, ~~Java~~, Scala (ou qualquer outra linguagem OO) não é o mais importante nessa disciplina. O mais importante é saber projetar um software utilizando boas abstrações e ter a preocupação com as características de um bom desenho OO + funcional. Isso requer o balanceamento entre a completude do projeto da disciplina com a qualidade da solução implementada. O desenho deve ser refletido em uma implementação.

Plano de Ensino

Técnicas de Programação 1

Rodrigo Bonifácio

2017/2