Nome: Davi Augusto Neves Leite

RA: 191027383

Atividade 2 – Conceitos de Projeto de Software

a) A abstração funcional está relacionada com o uso de subprogramas parametrizados, ou seja, constituir o programa principal por meio de uma coleção de grupos menores (rotinas visíveis e escondidas). Isso é relacionado, na orientação a objetos, com a utilização de classes **generalizadas** que estendem as subclasses constituintes e menores do sistema. Por exemplo, em um sistema ecológico, tem-se a classe de *Animais* como sendo uma abstração das classes *Anfíbios, Répteis* e *Mamíferos*, haja visto que estas três possuem **rotinas** comuns (como *respirar* ou se *alimentar*).

Desta forma, tendo em vista o sistema de controle de doações abordado no exercício, pode-se generalizar (ou seja, abstrair) as seguintes classes: *Voluntário*/*Doador*; *Donatário*; *Item Doado*. Por fim, pode-se pensar em uma abstração funcional relacionada a **rotinas** que salvem os dados dos donatários ou dos itens doados neste projeto.

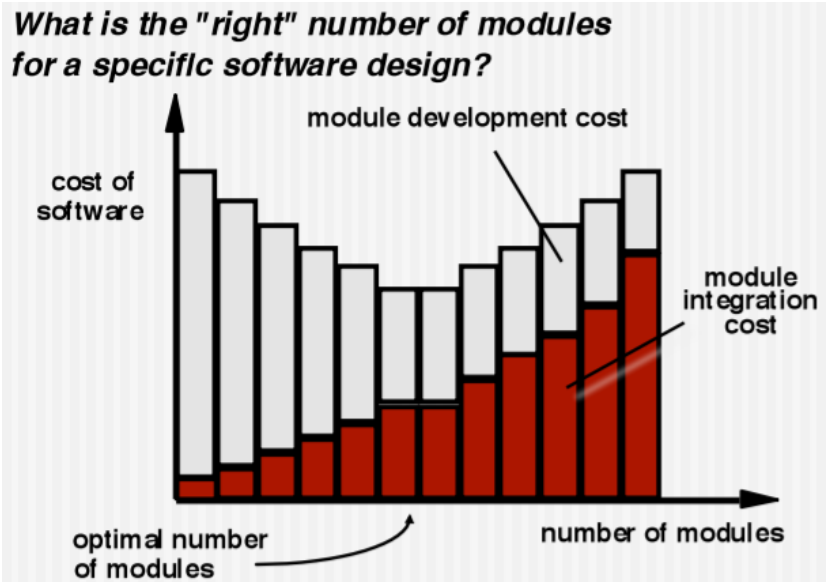
b) A abstração de dados refere-se à descrição das classes por meio de estrutura de dados, ou seja, aos **atributos** generalizados que a classe possui. Dessa forma, tendo em base o sistema do exercício, pode-se pensar em uma estrutura contendo as informações pessoais do Donatário (como *nome*, *endereço*, *CPF*, *telefone*) e a respeito do Item Doado (como *tipo de item*, *CPF do voluntário, CPF do donatário*).

c) A abstração de controle está relacionada a definição de controle por meio de especificações de ordens de instrução separadamente da implementação dessa ordem, ou seja, representa **como** realizar, de maneira ordenada, as ações de uma classe. Aplicando no sistema do exercício, pode-se pensar em **como** salvar os dados dos donatários sendo, por exemplo, utilizado um formulário em que há campos para os atributos e um botão de enviar essas informações a um banco de dados.

d) O conceito de modularidade, como o próprio nome sugere, está relacionado na utilização de **módulos** no projeto de software, ou seja, constituir e separar o desenvolvimento do software em partes menores (de tal forma em que se possa produzir umas as outras de forma paralela), **independentes** e de fácil interconexão. No ambiente do exercício, pode-se pensar em um módulo para a criação de um formulário de inscrição do *Doador* e um módulo para a criação de um formulário de inscrição do *Donatário*.

e) A arquitetura, em termos de projeto de software, está relacionada na definição dos componentes de software, suas propriedades e seus relacionamentos entre si. Ou seja, compreende em **como** um sistema deve ser **organizado** e **estruturado**, sendo uma etapa crítica do projeto pois identifica os principais componentes estruturais do projeto. Dessa forma, relacionando ao sistema do exercício, pode-se pensar em **estruturar** o sistema na interligação dos módulos dos formulários (descritos no item d) com um banco de dados.

f) Os chamados *tradeoffs* estão relacionados na **escolha** de uma decisão **em detrimento** de outra como, por exemplo, em um sistema baseado em módulos, como responder à pergunta de “Quantos módulos utilizar?” haja visto que quanto mais módulos, maior o custo do software e da integração em contrapartida em que é facilitado o trabalho individualmente. Para tanto, essa ideia é demonstrada no gráfico abaixo (retirado do livro *Software Engineering: A Practitioner’s Approach* de *Roger Pressman*, edição de número 7).



Portanto, relacionando ao sistema do exercício, pode-se pensar no *tradeoff* de “Qual linguagem de programação utilizar?” já que, por exemplo, enquanto que Java é uma linguagem multiplataforma, a linguagem C++ possui maior velocidade de execução em contrapartida a multiplataforma. Outro *tradeoff* pode ser relacionado a própria modularização do sistema, no que diz respeito a “Quantos módulos o sistema deve possuir?”, como mostra a imagem acima.