Arquitetura de Software !

Introdução

Prof. Galdir Reges

Tópicos

- O que é Arquitetura de Software?
- Como se desenha bons softwares?
- O que é decomposição de sistema?
- O que é Subsistema?
- O que são serviços?
- O que é acoplamento?
- O que é coesão?
- O que são camadas?
- O que são partições?

Pesquisa diagnóstica

- Brainstorming: Arquitetura de software?

Desenho de software

- “Existem duas maneiras de construir um desenho de software: Uma maneira é torná-lo tão simples que obviamente não há deficiências, e a outra maneira é torná-lo tão complicado que não há deficiências óbvias.”

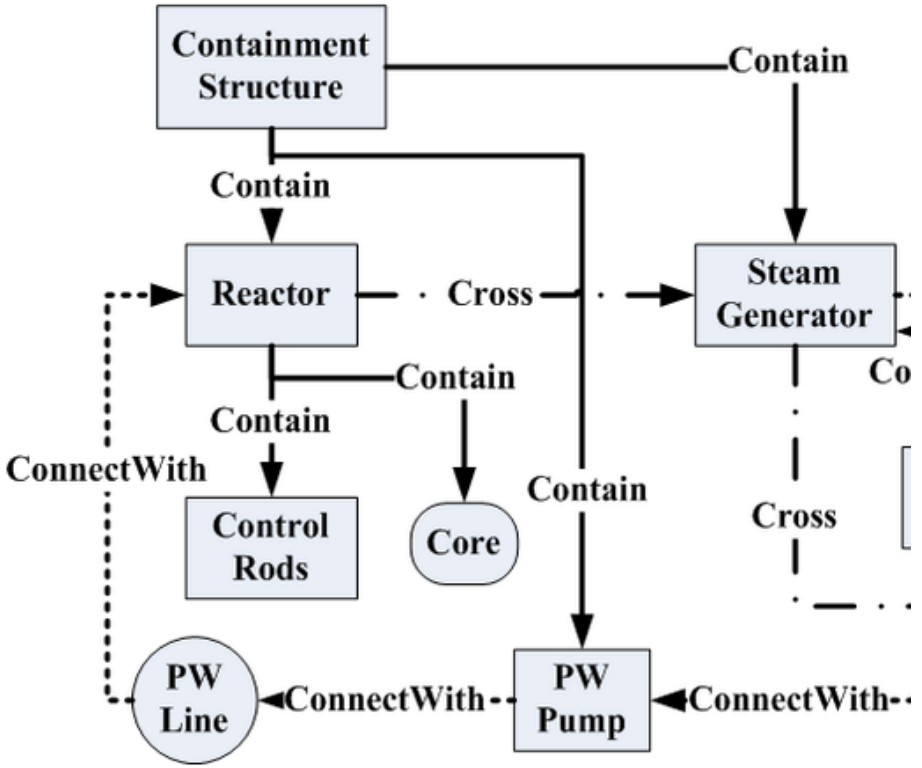
  C.A.R. Hoare, in The Emperor’s Old Clothes

Desenho do sistema

- ✧ É a transformação de um modelo de análise de requisitos em um modelo do sistema.
- ✧ Os desenvolvedores definem:
  1. os objetivos do desenho do projeto e decompõem o sistema em subsistemas menores que podem ser realizados por equipes individuais.
  2. a estratégia de hardware / software
  3. a estratégia de gerenciamento de dados persistentes
  4. o fluxo de controle global
  5. a política de controle de acesso
  6. o tratamento de condições límitrofes
- ✧ O resultado é um modelo que inclui isso tudo

Decompondo o sistema

- ✧ O desenho do sistema não é algorítmico.
  - Muitos trade-offs entre objetivos.
- ✧ O desenho do sistema é decomposto em várias atividades, cada uma abordando parte do problema geral:
  - Identificar os objetivos do desenho.
  - Desenhar a decomposição inicial em subsistemas.
  - Refinar a decomposição em subsistemas para atender às metas do desenho.



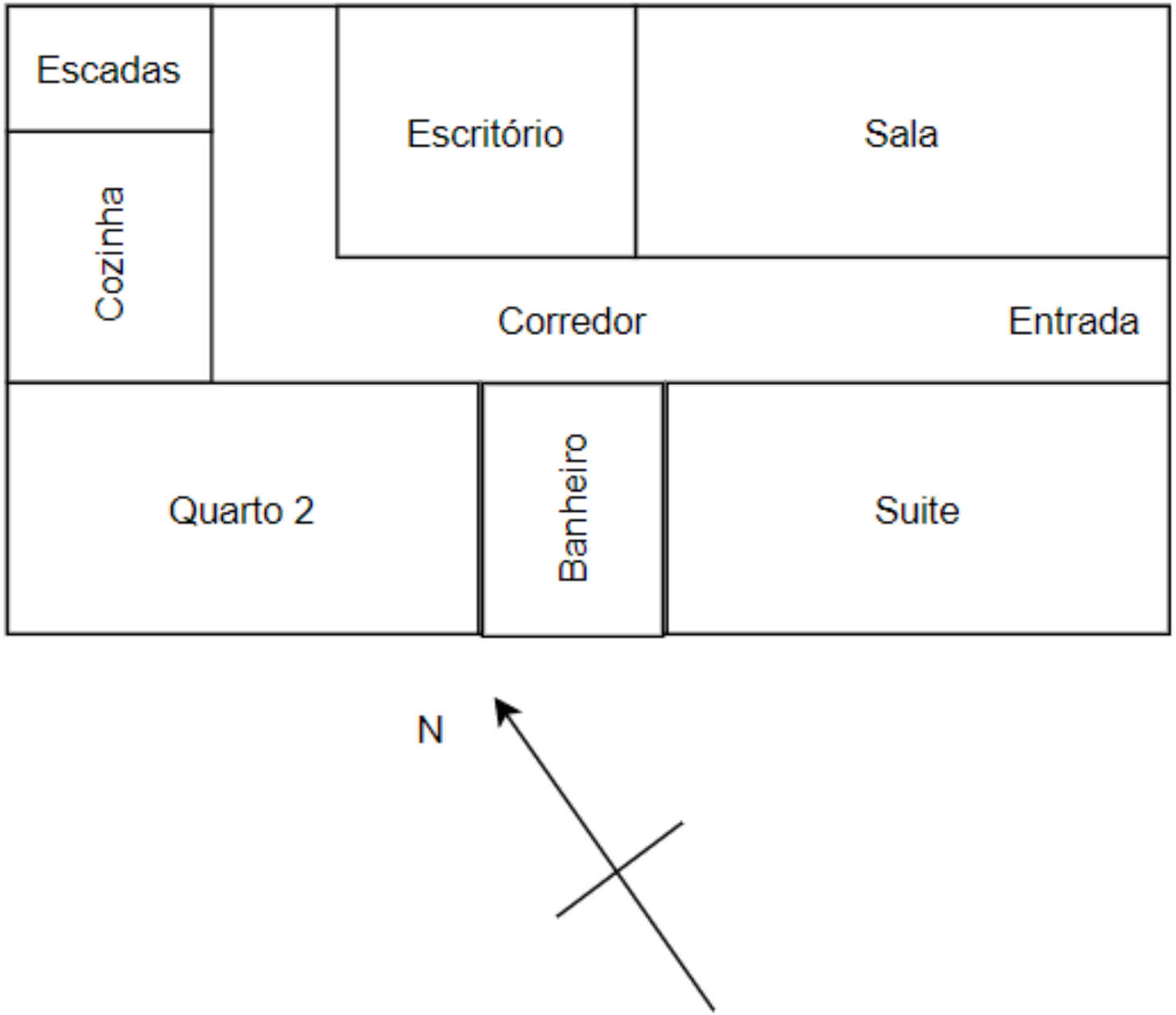
Um exemplo de planta baixa

- ✧ Depois de concordar com o cliente sobre o número de cômodos e pisos, o tamanho da área de estar e a localização da casa, **o arquiteto deve projetar a planta, ou seja, onde as paredes, portas e janelas devem estar localizadas**.
  - **de acordo com vários requisitos funcionais**: a cozinha deve estar perto da sala de jantar e da garagem, o banheiro deve estar perto dos quartos e assim por diante.
- ✧ **O arquiteto também pode confiar em vários padrões** ao estabelecer as dimensões de cada quarto e a localização da porta
  - os armários de cozinha são fornecidos em incrementos fixos e as camas em tamanhos padrão.
- ✧ **O arquiteto não precisa conhecer o conteúdo exato de cada sala** e o layout dos móveis; pelo contrário, essas decisões devem ser adiadas e deixadas ao cliente.

Primeira versão

1. Esta casa deve ter dois quartos, um escritório, uma cozinha e uma sala de estar.
2. A distância total que os ocupantes percorrem todos os dias deve ser minimizada.
3. O uso da luz do dia deve ser maximizado.

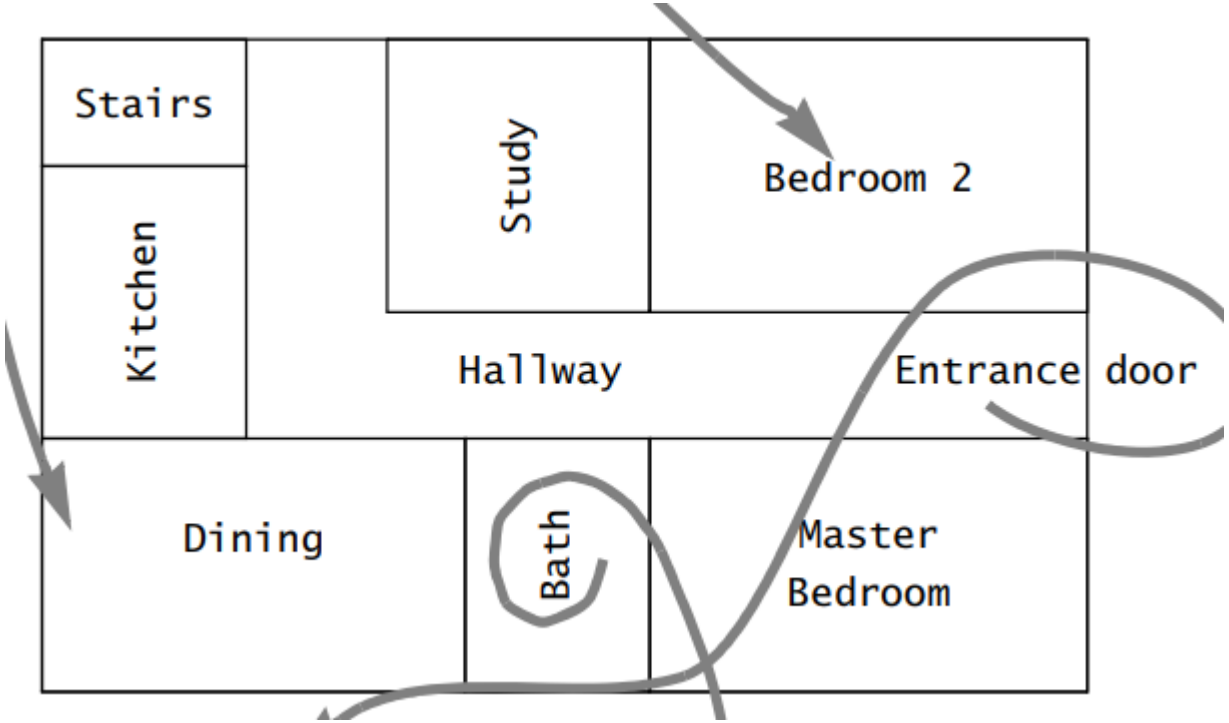
✧ Problemas? Sugestões?



Segunda versão

1. Esta casa deve ter dois quartos, um escritório, uma cozinha e uma sala de estar.
2. A distância total que os ocupantes percorrem todos os dias deve ser minimizada.
3. O uso da luz do dia deve ser maximizado.

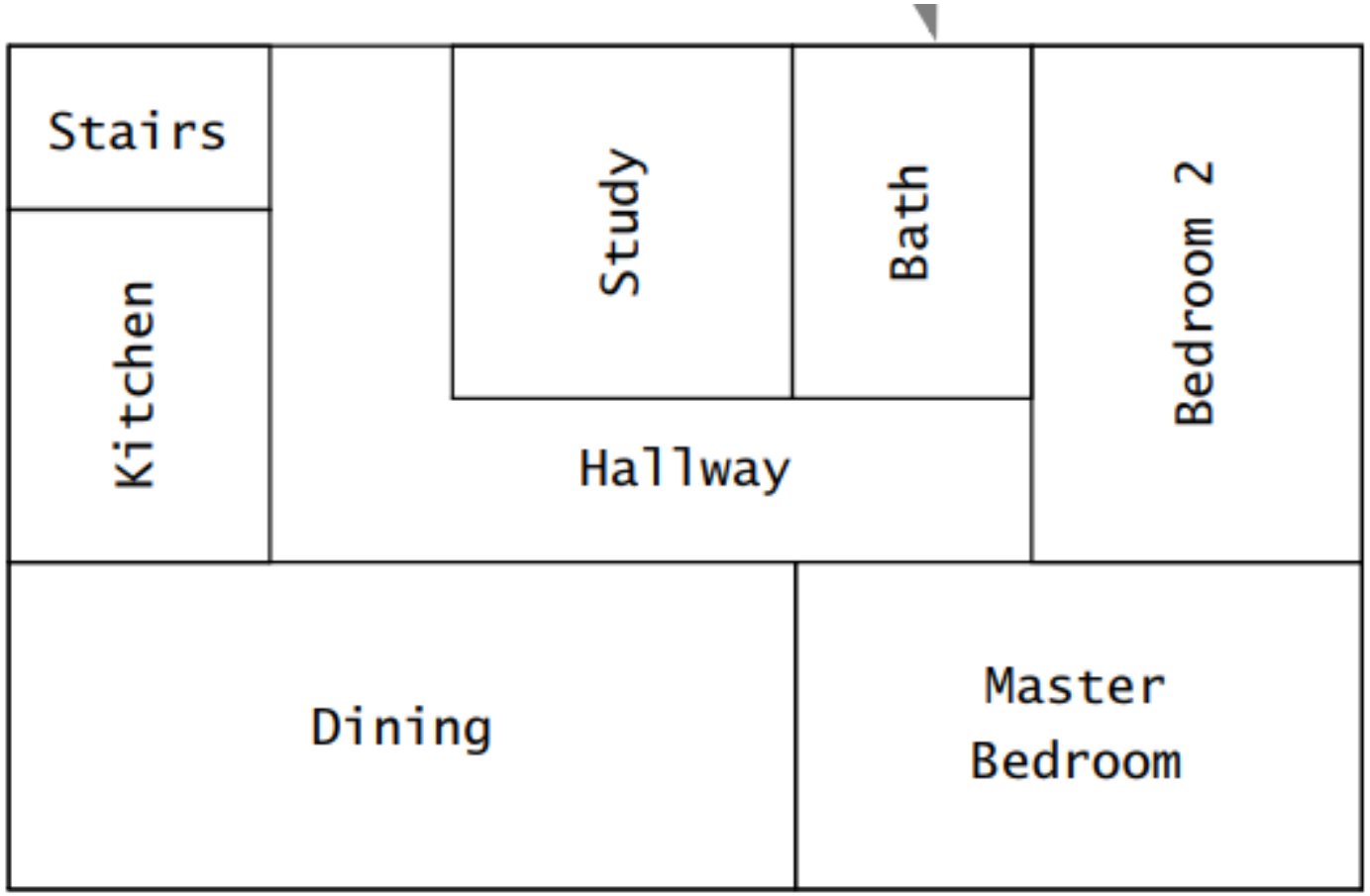
✧ Problemas? Sugestões?



Terceira versão

1. Esta casa deve ter dois quartos, um escritório, uma cozinha e uma sala de estar.
2. A distância total que os ocupantes percorrem todos os dias deve ser minimizada.
3. O uso da luz do dia deve ser maximizado.

✧ Agora podemos posicionar portas e janelas e atender requisitos de cada componente.



Semelhanças entre arquitetura civil e de software

- ✧ O conjunto é dividido em componentes e interfaces mais simples, levando em consideração requisitos funcionais e não funcionais.
- ✧ O design do sistema afeta as atividades de implementação e resulta em retrabalho caro, se alterado posteriormente. O design de componentes individuais é adiado até mais tarde.

	Conceito de arquitetura civil	**Conceito de engenharia de software**
Componentes	Quartos	Subsistemas
Interfaces	Portas	Serviços
Requisitos não funcionais	Menor distancia entre cômodos	Tempo de resposta
Requisitos funcionais	Casa residencial	Casos de uso
Custo de retrabalho	Mover paredes	Mudanças de interfaces

Atividade prática

- Uma casa sem paredes internas tem quantos componentes? Como podem ser rearranjados?

- Vamos rodar o código ao lado e refatorar ele para que tenha 2 componentes, um para as funções de interface com o usuário e outro para calcular IMC.

```
package calculaIMC;

import javax.swing.JOptionPane;

//declaracao de classe igual nome do arquivo
public class CalculaIMC {
//metodo principal
public static void main(String[] args) {
// obtem entrada de usuario de um dialogo JOPtionPane
String primeirNumero =
JOptionPane.showInputDialog("Informe a altura");
String segundoInteiro =
JOptionPane.showInputDialog("Informe o peso");

// convert String inputs to int values for use in a calculation
double altura = Double.parseDouble(primeirNumero);
double peso = Double.parseDouble(segundoInteiro);

double imc = peso/(altura*altura); // add numbers

// display result in a JOptionPane message dialog
JOptionPane.showMessageDialog(null, "O IMC é " + imc,
"Cálculo de IMC", JOptionPane.WARNING_MESSAGE);
}
}
```

Desenho de sistemas

- Conceitos

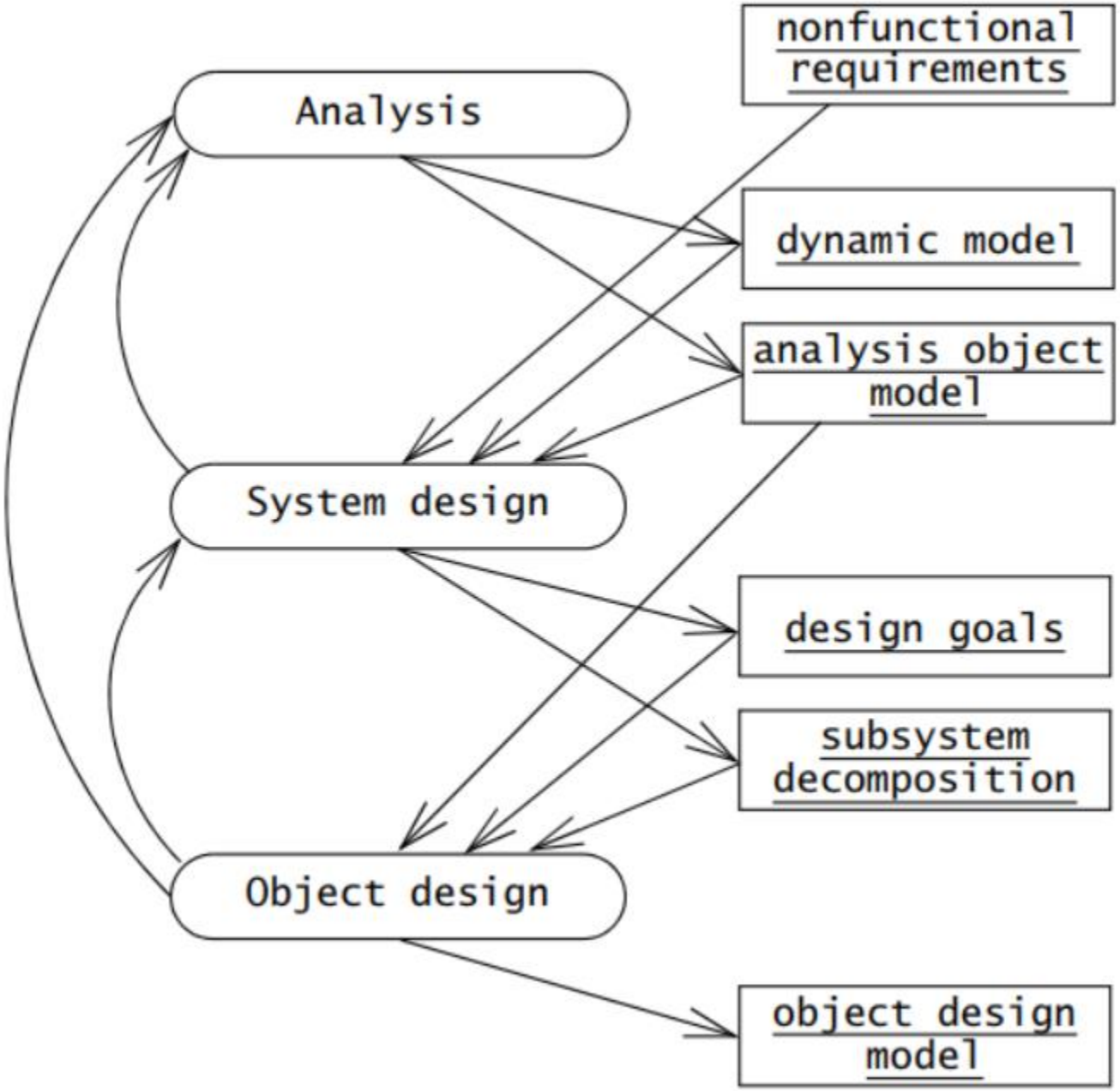
Conceitos para projeto ou para gambiarra?

Desenho de sistemas (1)

- A **análise dos requisitos** de um sistema resulta em modelo de **análise de sistema**:
  - Um conjunto de requisitos não funcionais e restrições
  - Modelo de casos de uso
  - Modelo de objetos
  - Diagramas de sequencia para cada caso de uso
- **Não contém informação sobre a estrutura interna do sistema**, configuração de hardware, nem como o sistema deve ser montado.

Desenho de sistemas (2)

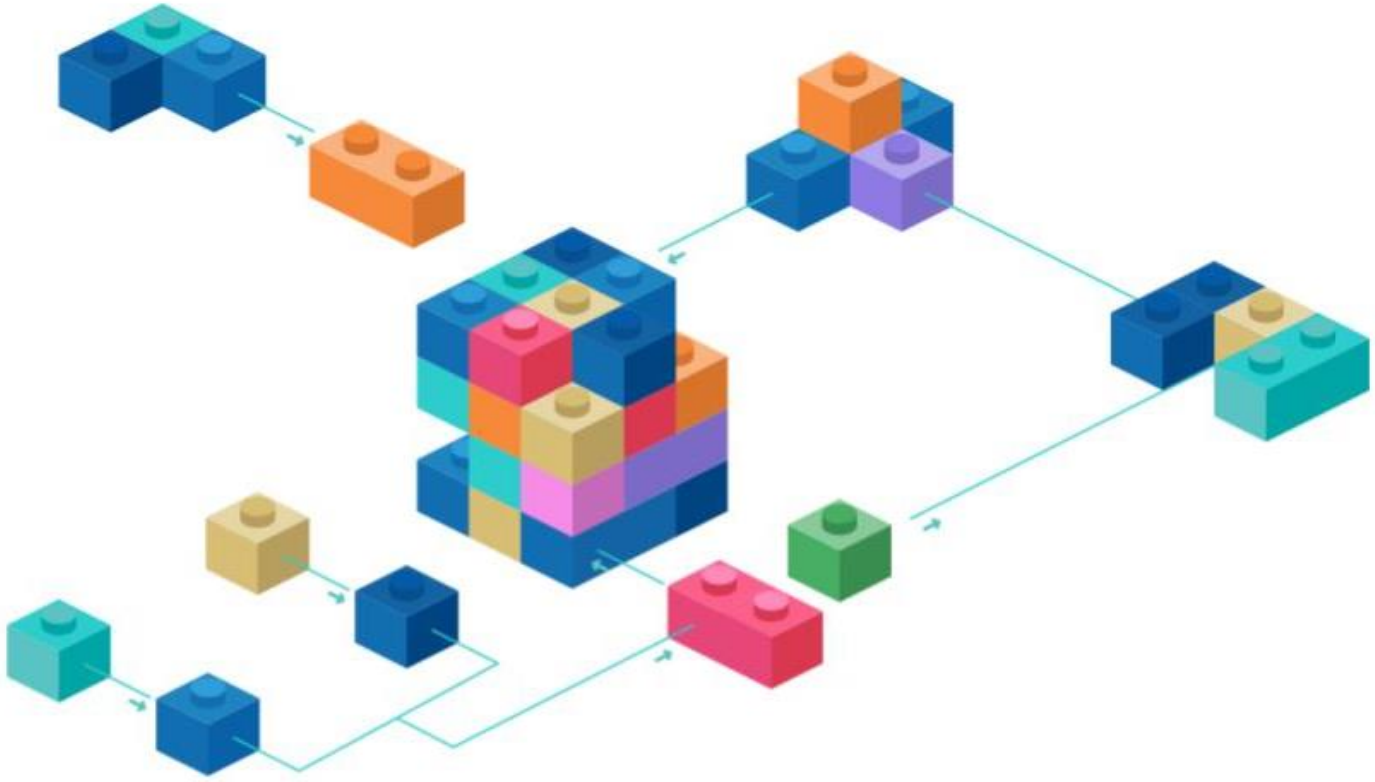
- O processo de **desenho (design) do sistema** usa o modelo de requisitos e o modelo de análise para gerar:
  - Metas do desenho
  - Arquitetura de software
  - Casos de uso limítrofes



Conceitos para o desenho de sistemas

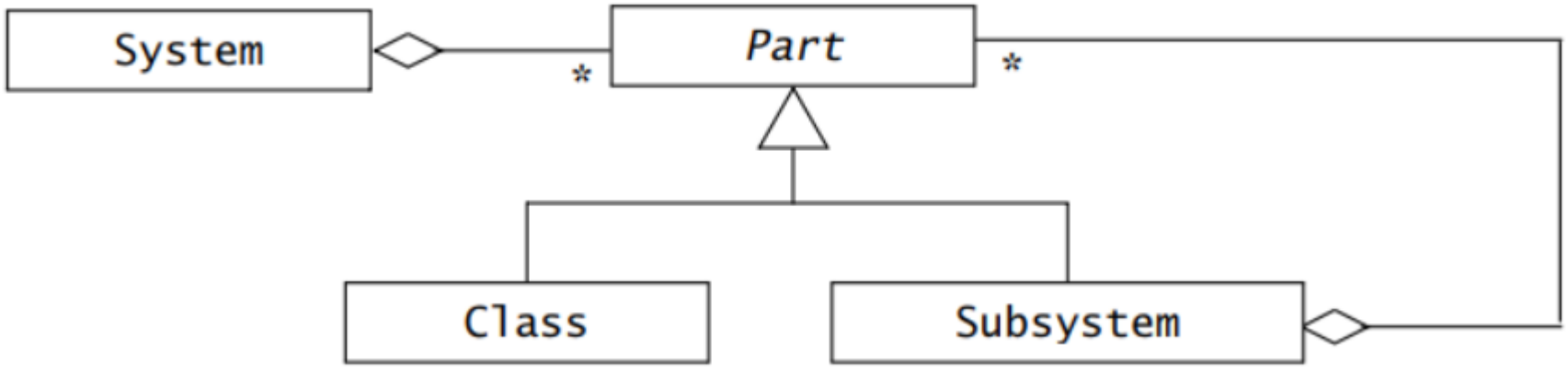
Conceitos que precisamos conhecer:

- ✧ Subsistemas
- ✧ Serviços
- ✧ Acoplamento
- ✧ Coesão
- ✧ Camadas
- ✧ Partições



Subsistemas

- Parte substituível do sistema com interfaces bem definidas que **encapsula o estado e o comportamento** de suas classes contidas.
  - Normalmente corresponde à quantidade de trabalho que um único desenvolvedor ou uma única equipe de desenvolvimento pode realizar.
  - Decompor sistema em subsistemas e repetir recursivamente.

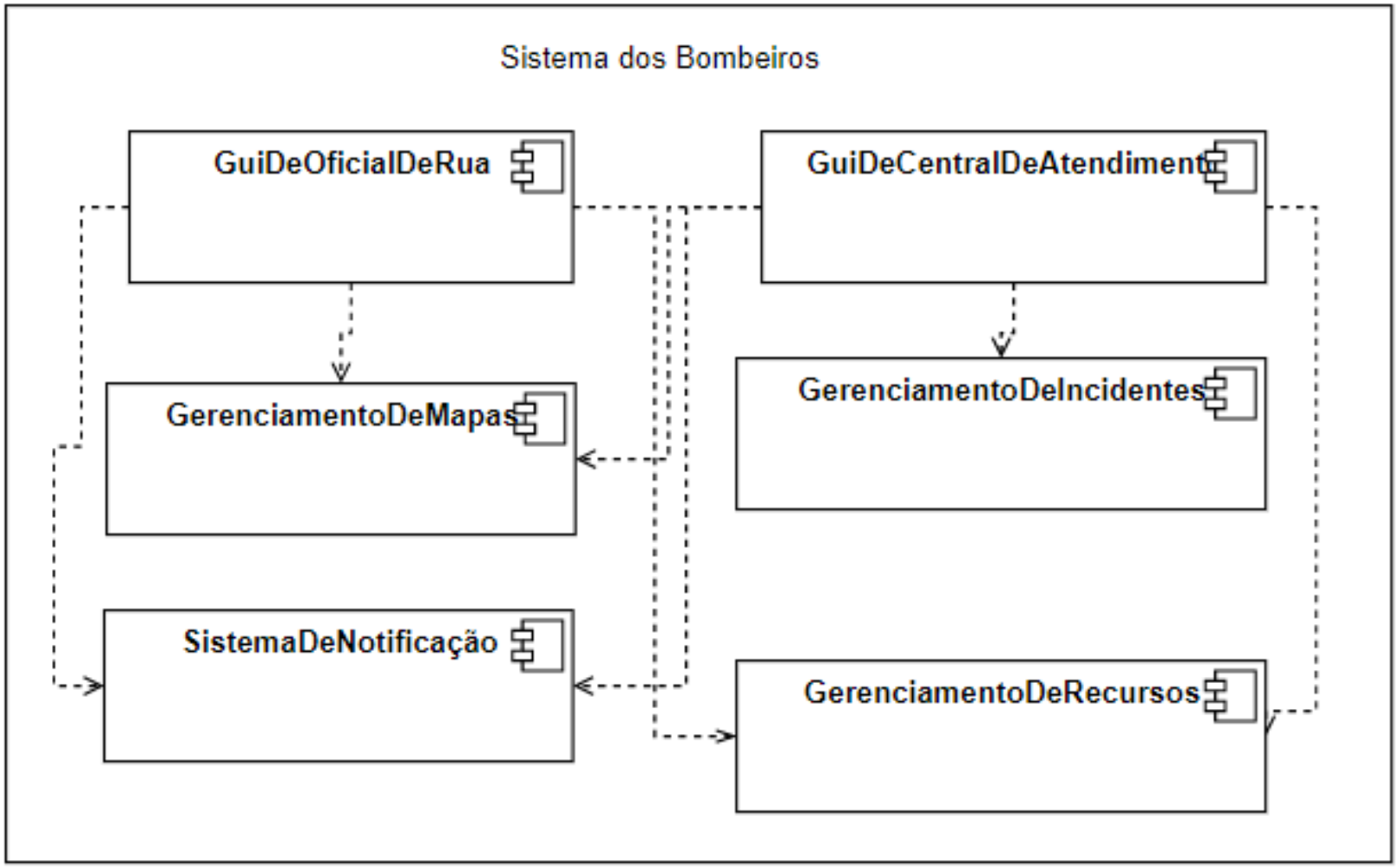


Exemplo de sistema de gerenciamento de acidentes

- Componentes?
- Ícones de componentes?
- Componente lógico ou físico?
- Dependências?

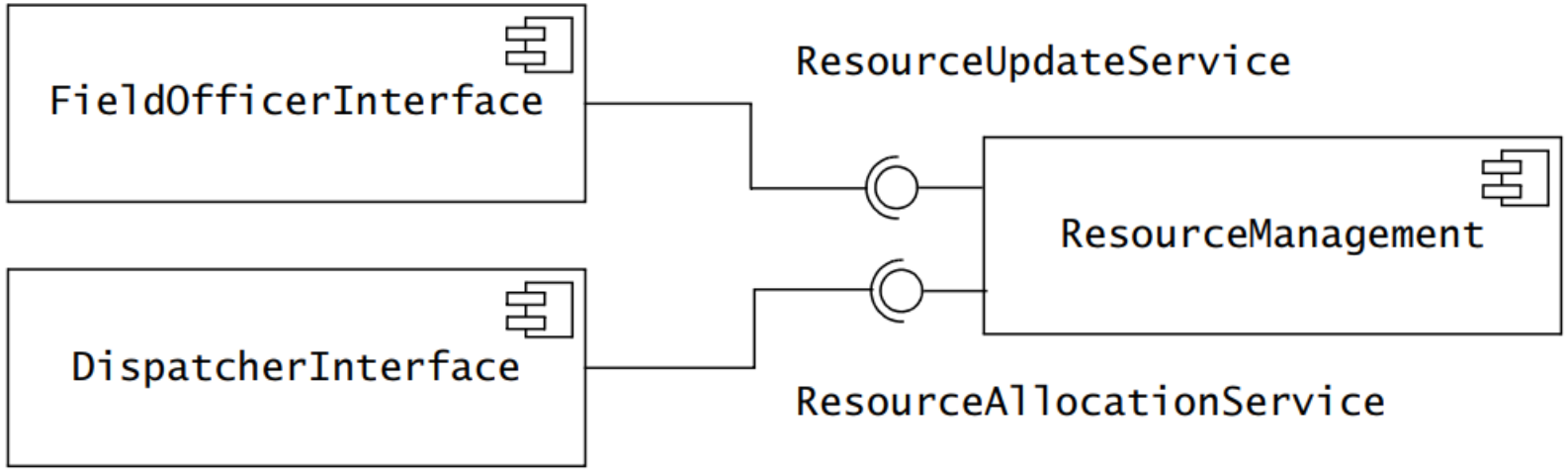
*GUI = interface gráfica de usuário

Subsistemas:
- GUI de oficial de rua
- GUI de despacho de equipe
- Gerenciamento de Mapas
- Gerenciamento de Incidentes
- Sistema de notificação
- Gerenciamento de Recursos



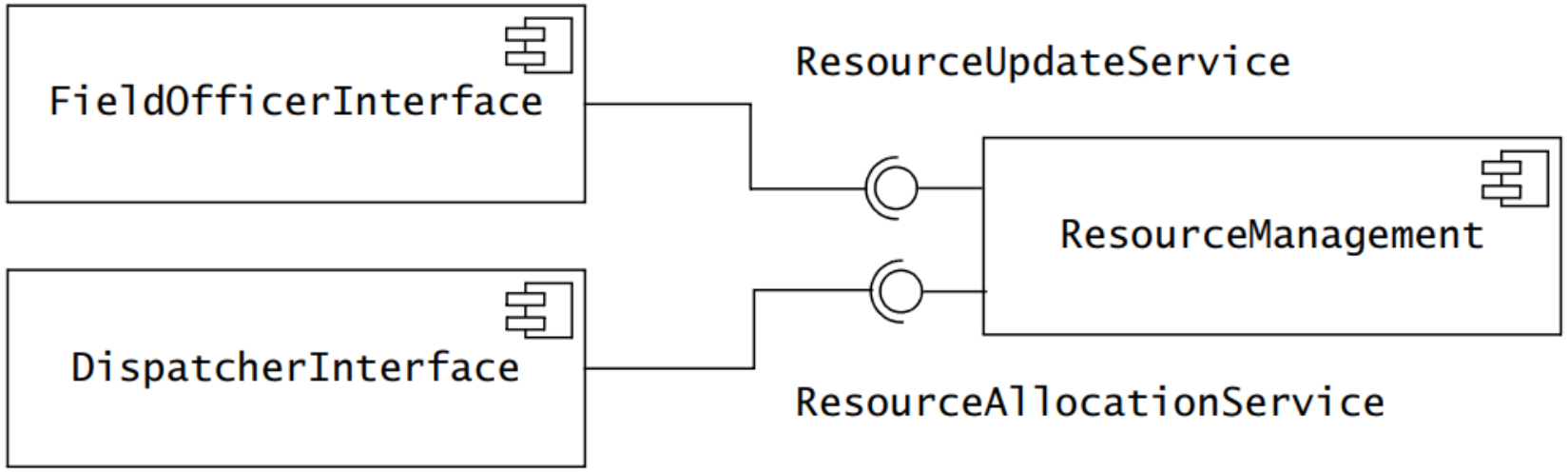
Serviços

- Um subsistema é caracterizado pelos serviços que fornece a outros subsistemas.
- Um serviço é um conjunto de operações relacionadas que compartilham um objetivo comum.
- O conjunto de operações de um subsistema que está disponível para outros subsistemas formam a interface do subsistema.



Serviços (2)

- A interface fornecida é mostrada como um ícone de bola (também chamado de pirulito) com o nome ao lado.
- Uma interface necessária é mostrada como um ícone de soquete.
- A dependência entre dois subsistemas é mostrada conectando a esfera e o soquete correspondentes no diagrama de componentes.

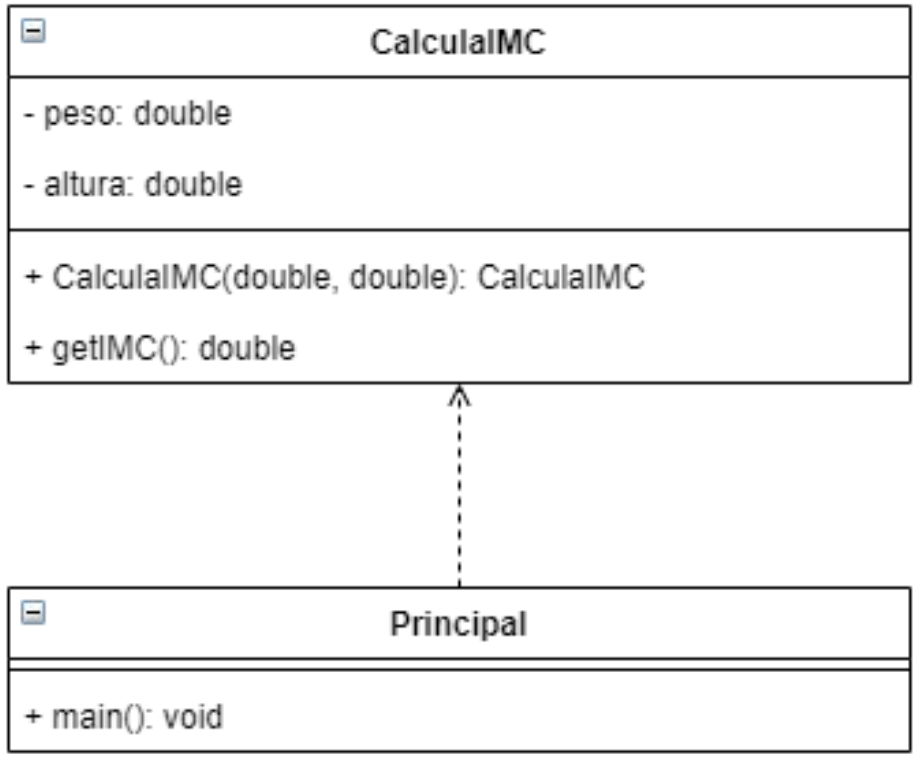


Atividade prática

- ✧ Desenhar diagrama de serviço do projeto CalculaIMC-OO

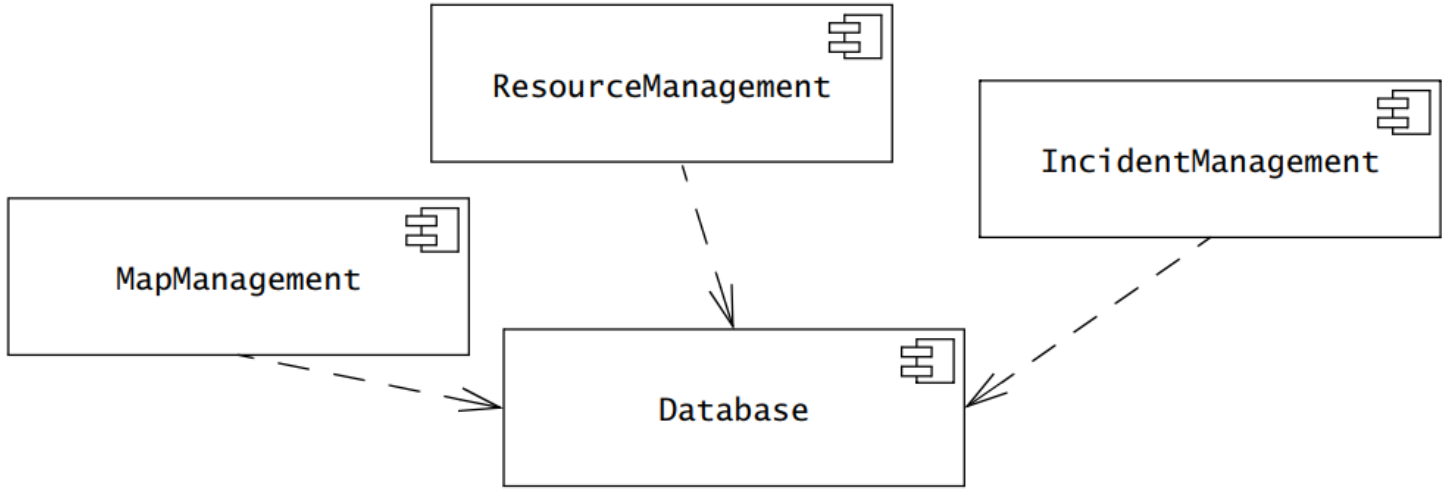
Atividade prática (resultado)

- ✧ Desenhar diagrama de serviço do projeto CalculaIMC-OO



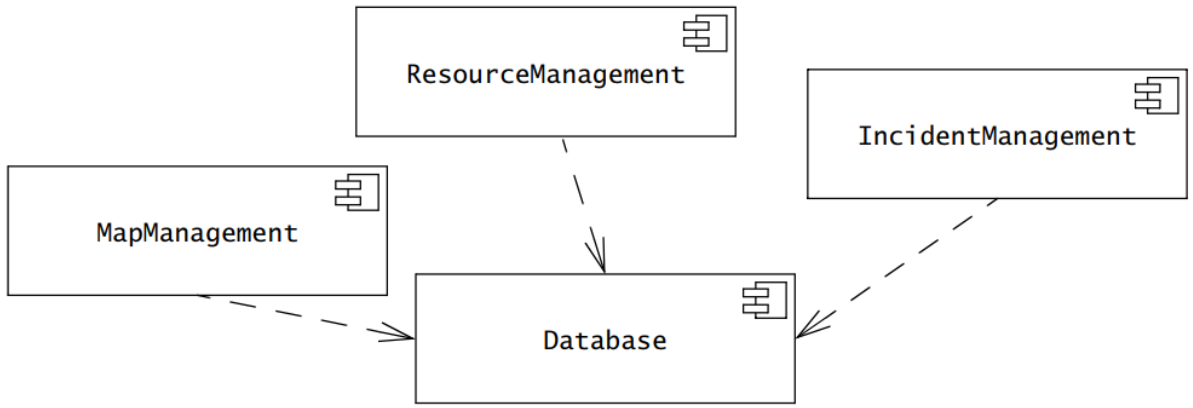
Acoplamento e coesão

- ✧ Acoplamento é o nível de dependências entre subsistemas.
  - fracamente acoplados > relativamente independentes > as modificações em um dos subsistemas terão pouco impacto no outro.
  - fortemente acoplados > as modificações em um subsistema provavelmente terão impacto no outro.
- ✧ desejável decomposição até com acoplamento mínimo possível

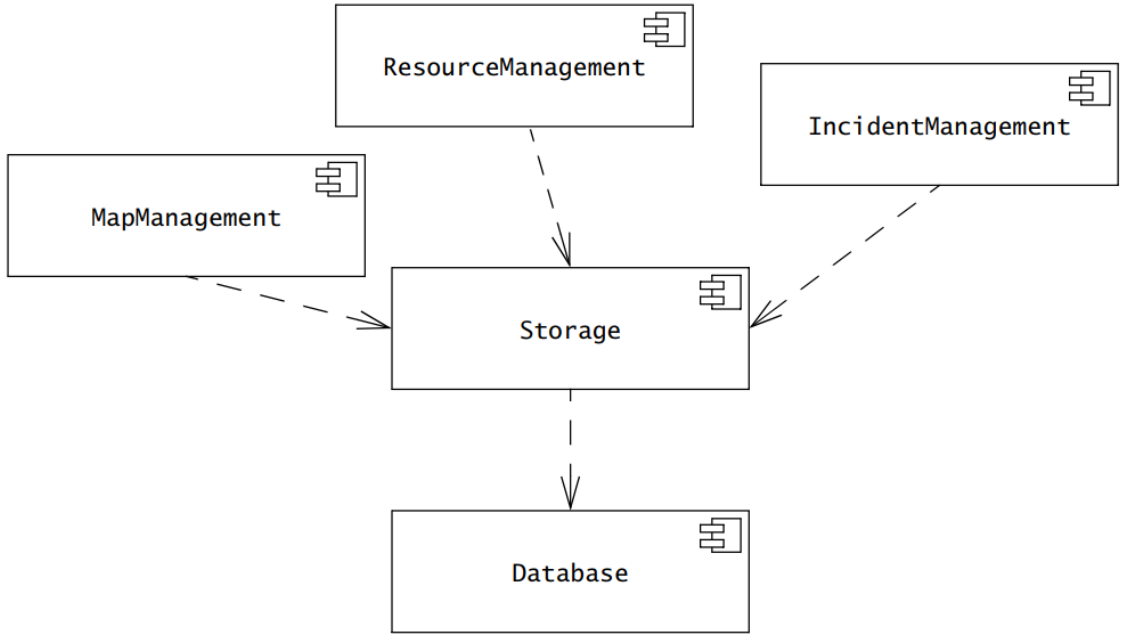


Acoplamento e coesão (2)

Alternative 1: Direct access to the Database subsystem



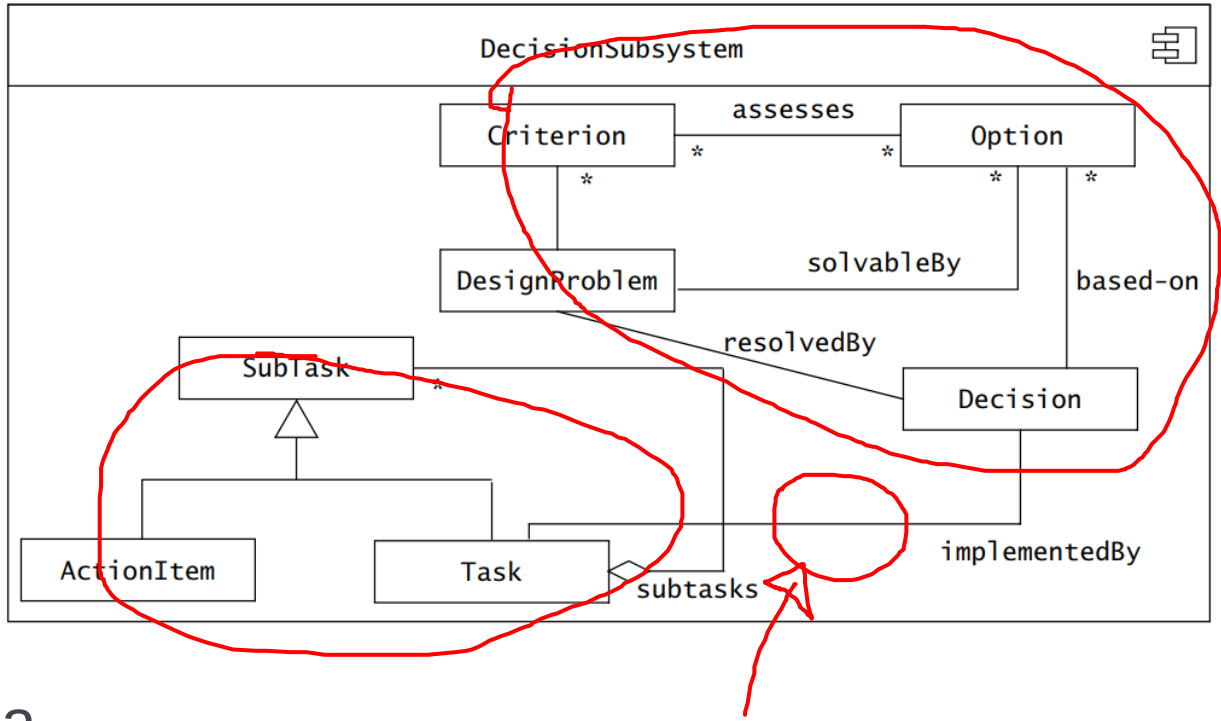
Alternative 2: Indirect access to the Database through a Storage subsystem



Menos acoplamento
Mais complexidade

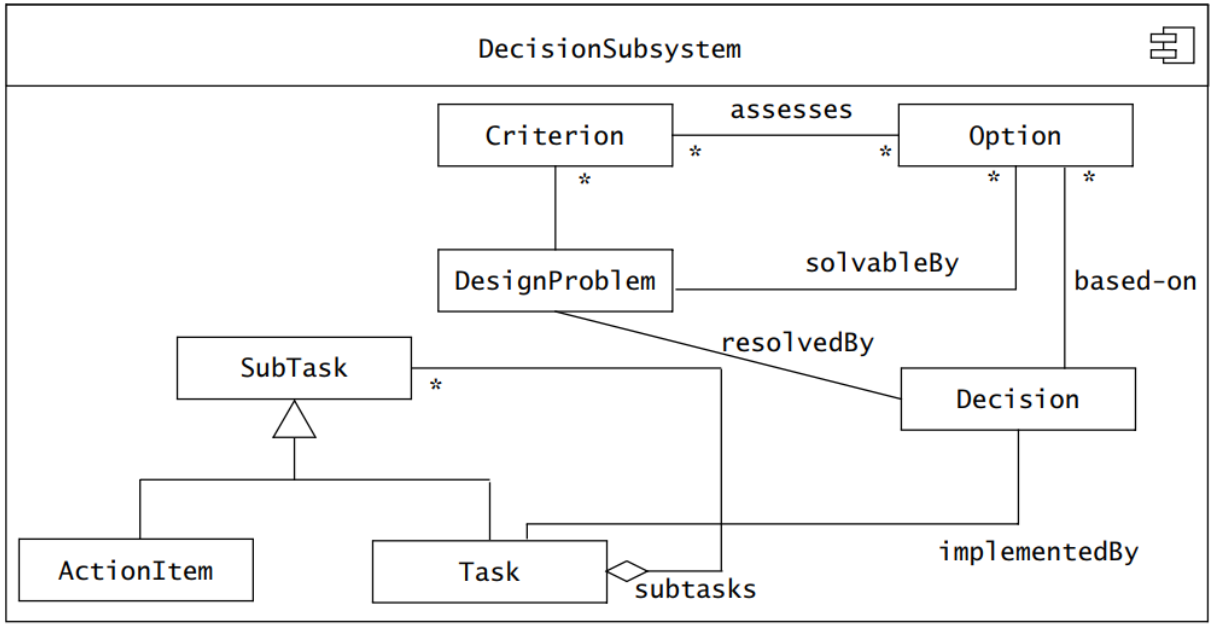
Acoplamento e coesão (3)

- Coesão diz respeito às associações dentro de um subsistema.
  - Se um subsistema **contiver muitos objetos relacionados** entre si e executar tarefas semelhantes, sua **coesão é alta**.
  - Se um subsistema **contiver vários objetos não relacionados**, sua **coesão será baixa**.
  - Uma propriedade desejável de uma decomposição de subsistema é que ela leva a subsistemas com alta coesão.



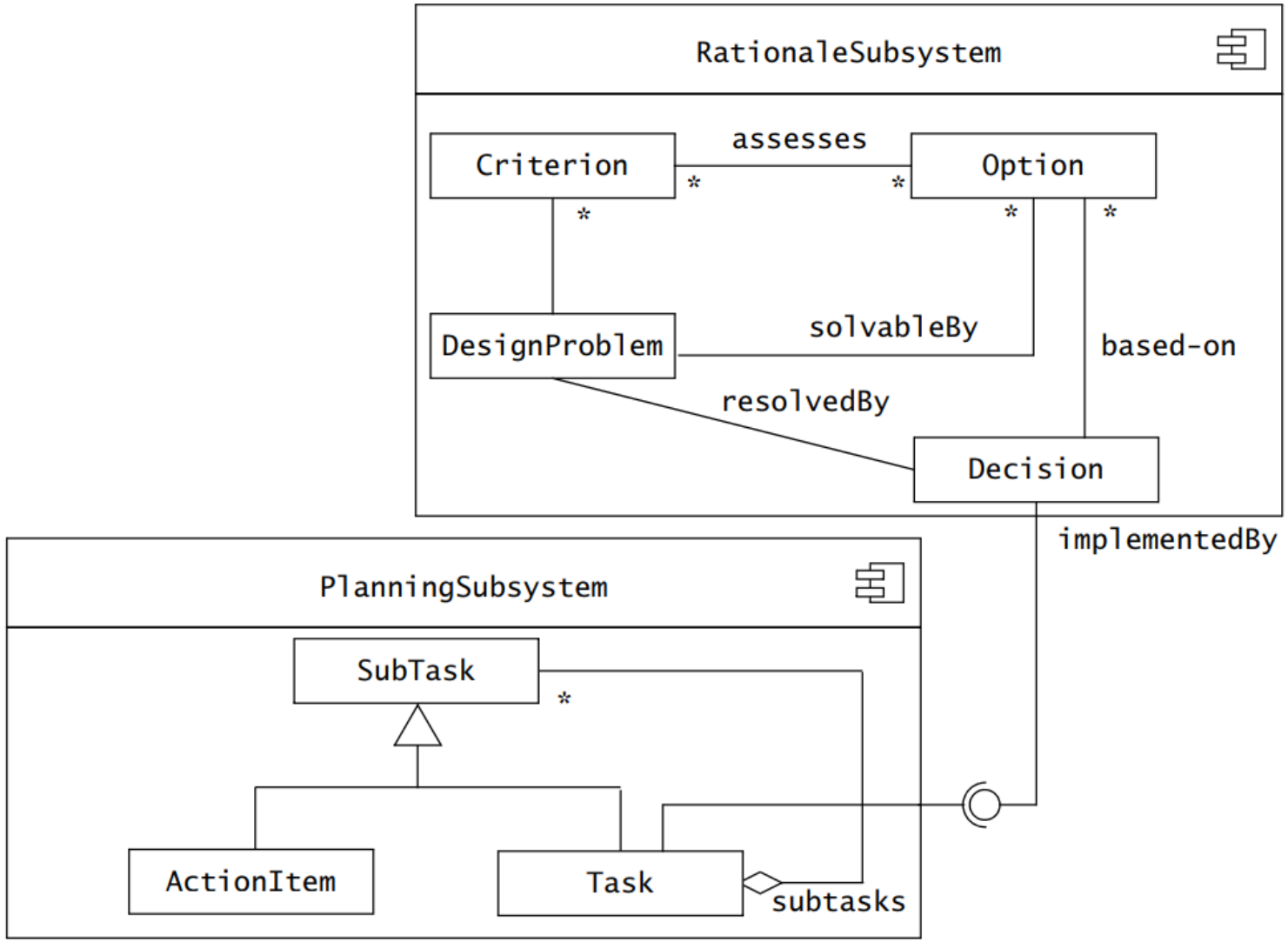
Acoplamento e coesão (3)

- Professor decompõe o sistema em dois subsistemas.



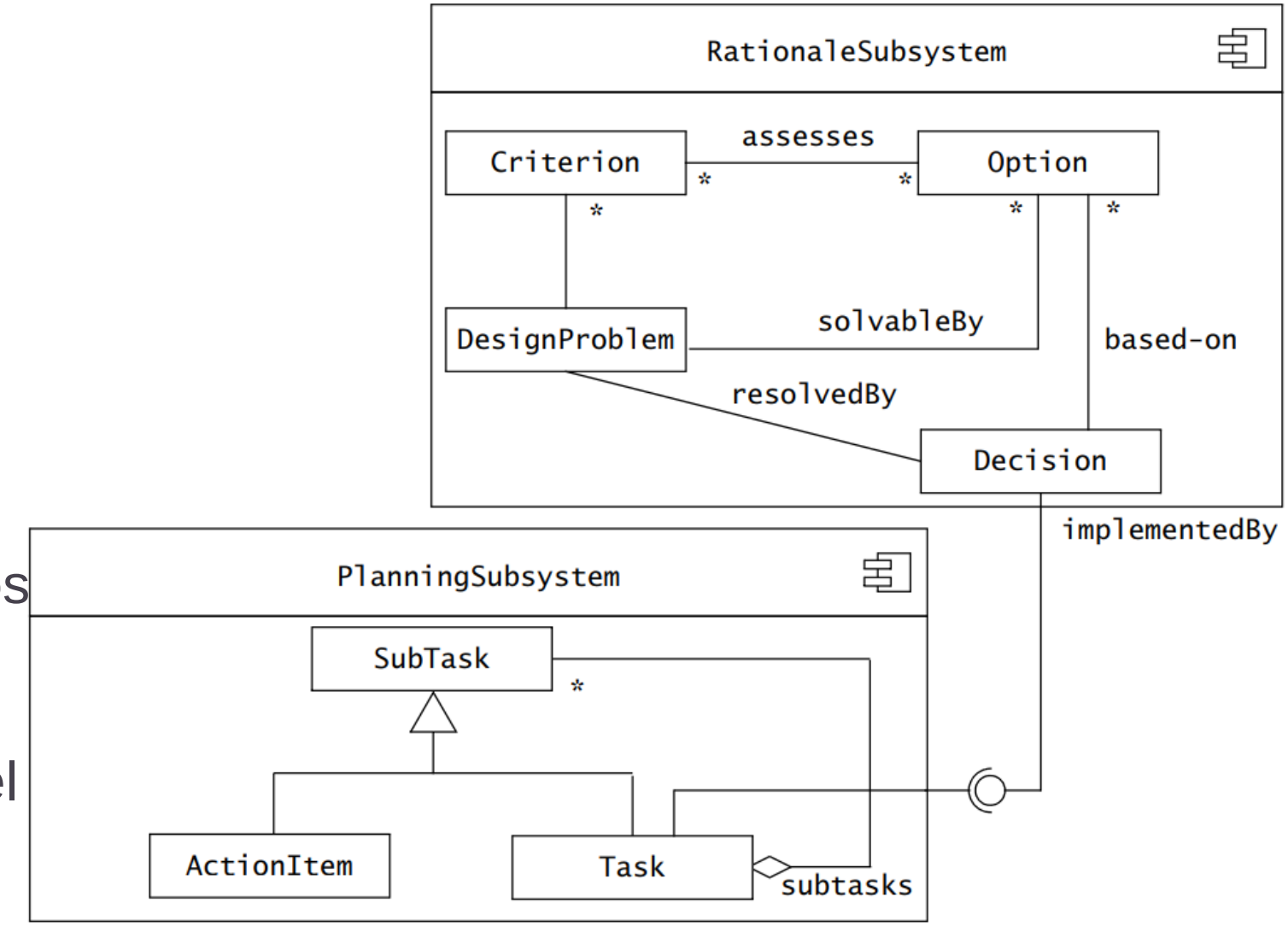
Acoplamento e coesão (4)

- ✧ Ambos os subsistemas têm uma coesão mais alta que o DecisionSubsystem original.
  - permite reutilizar cada parte
  - subsistemas resultantes são menores
- ✧ O acoplamento entre os subsistemas é relativamente baixo
- ✧ Em geral há um balanço entre coesão e acoplamento.



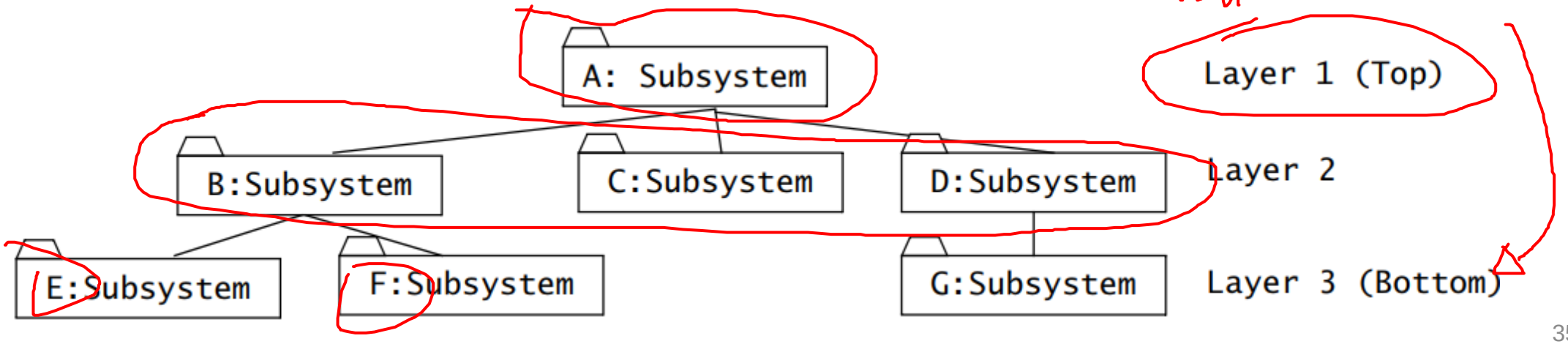
Acoplamento e coesão (5)

- podemos **aumentar a coesão decompondo o sistema em subsistemas menores**. No entanto, isso também pode aumentar o acoplamento à medida que o número de interfaces aumenta.
- Uma boa heurística é que os desenvolvedores podem lidar com 7 ± 2 conceitos (serviços) em qualquer nível de abstração



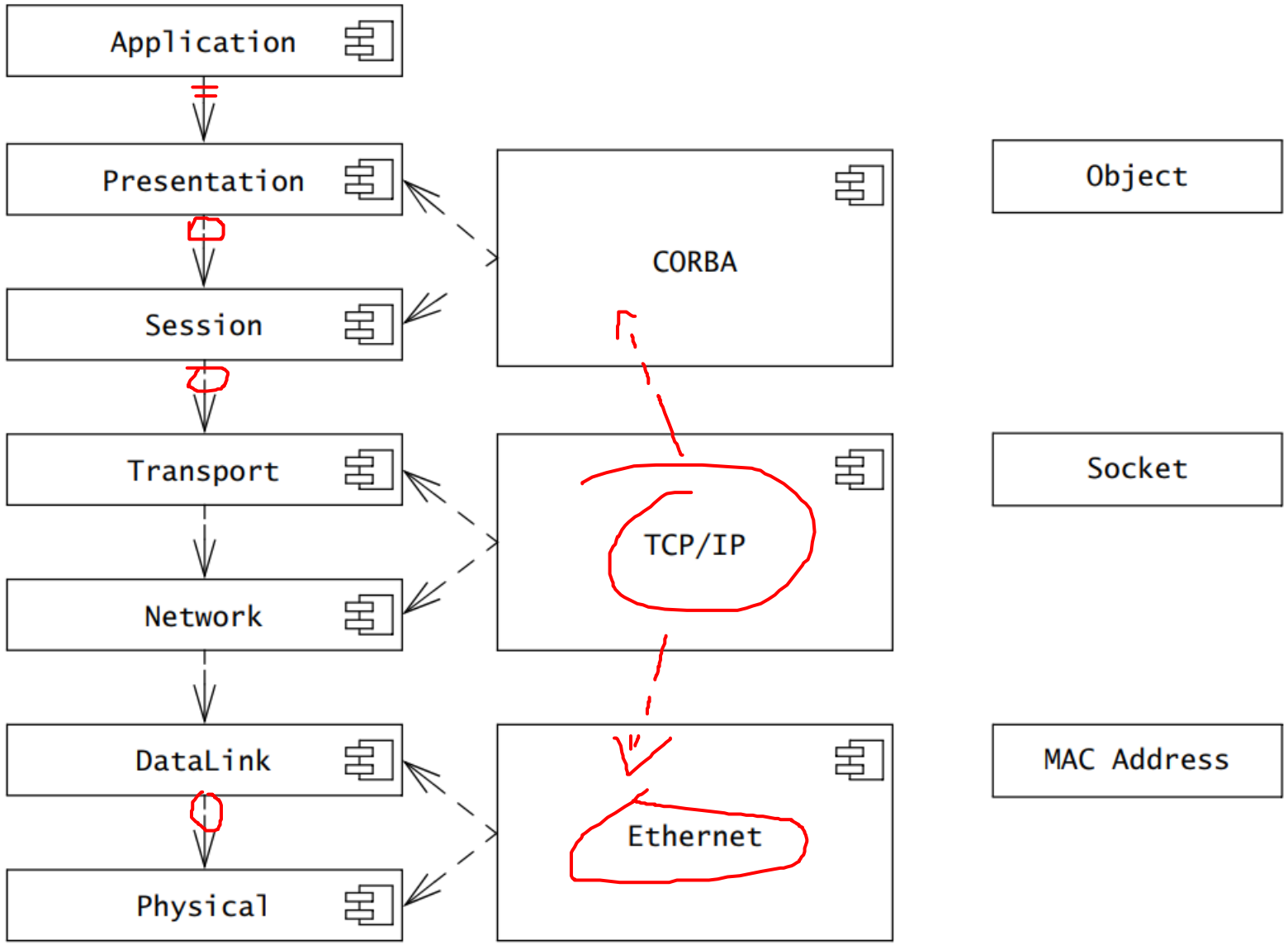
Camadas e partições

- ✧ Uma decomposição hierárquica de um sistema produz um conjunto ordenado de camadas.
- ✧ Uma camada é um agrupamento de subsistemas que fornecem serviços relacionados, possivelmente realizados usando serviços de outra camada
- ✧ Em uma arquitetura fechada, cada camada pode acessar apenas a camada imediatamente abaixo dela. Em uma arquitetura aberta, uma camada também pode acessar camadas em níveis mais profundos.



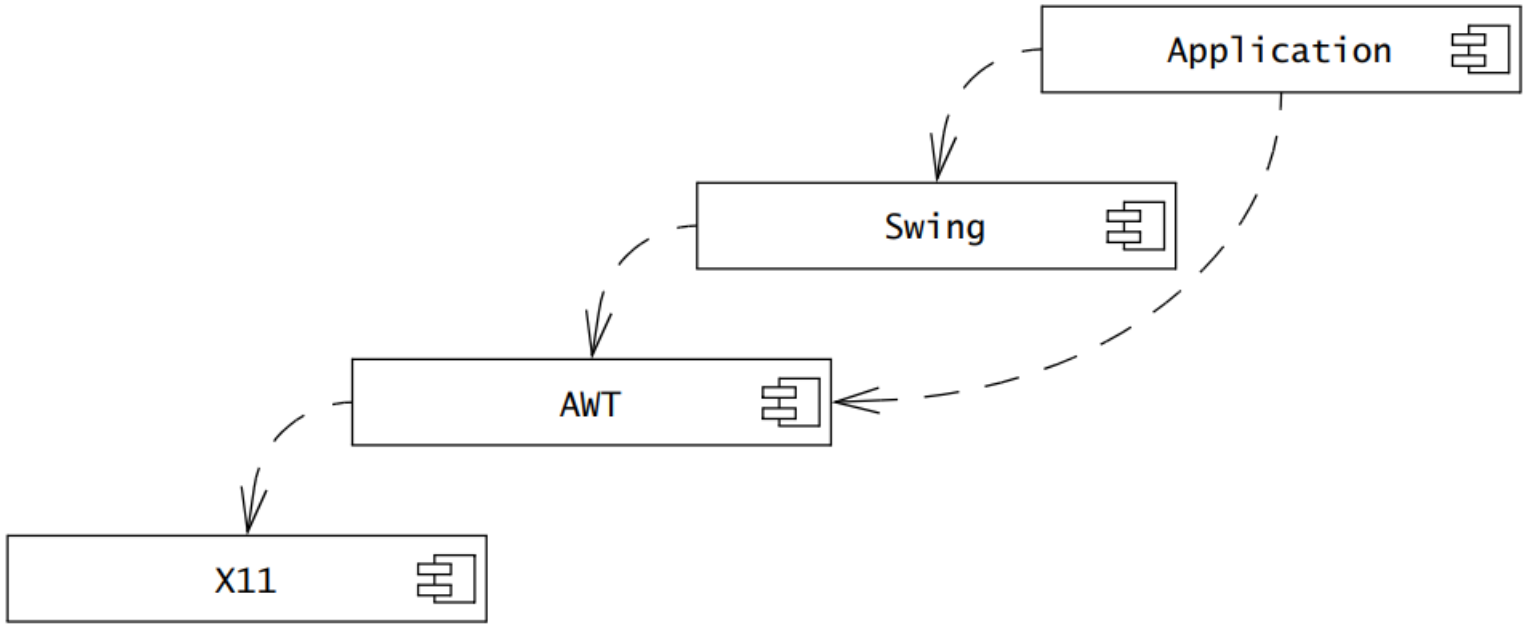
Camadas e partições (2)

- Um exemplo de arquitetura fechada é o Modelo de Referência de Interconexão de Sistemas Abertos (em resumo, o modelo OSI)



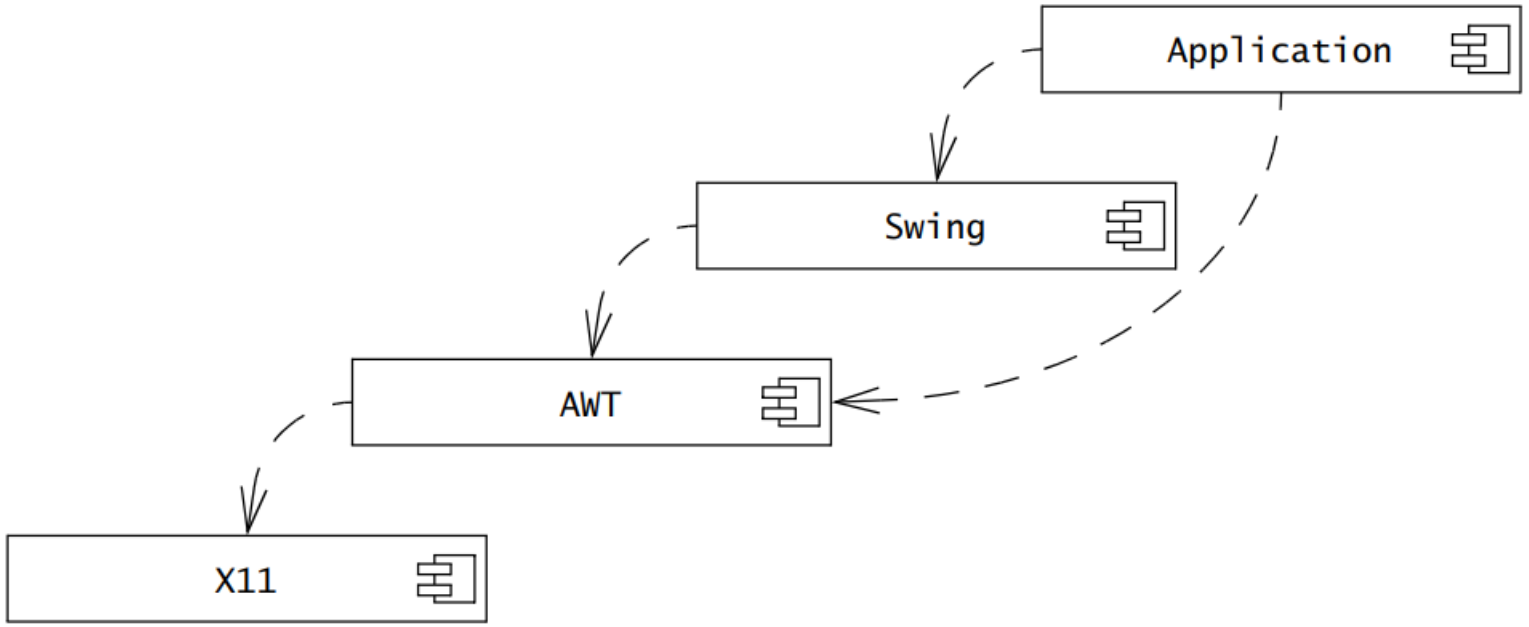
Camadas e partições (3)

- ✧ Um exemplo de arquitetura aberta é o kit de ferramentas da interface do usuário Swing para Java [JFC, 2009].
- ✧ A camada mais baixa é fornecida pelo sistema operacional ou por um sistema de janelas, como o X11, e fornece gerenciamento básico de janelas. AWT é uma interface de janela abstrata fornecida pelo Java para proteger aplicativos de plataformas de janela específicas.



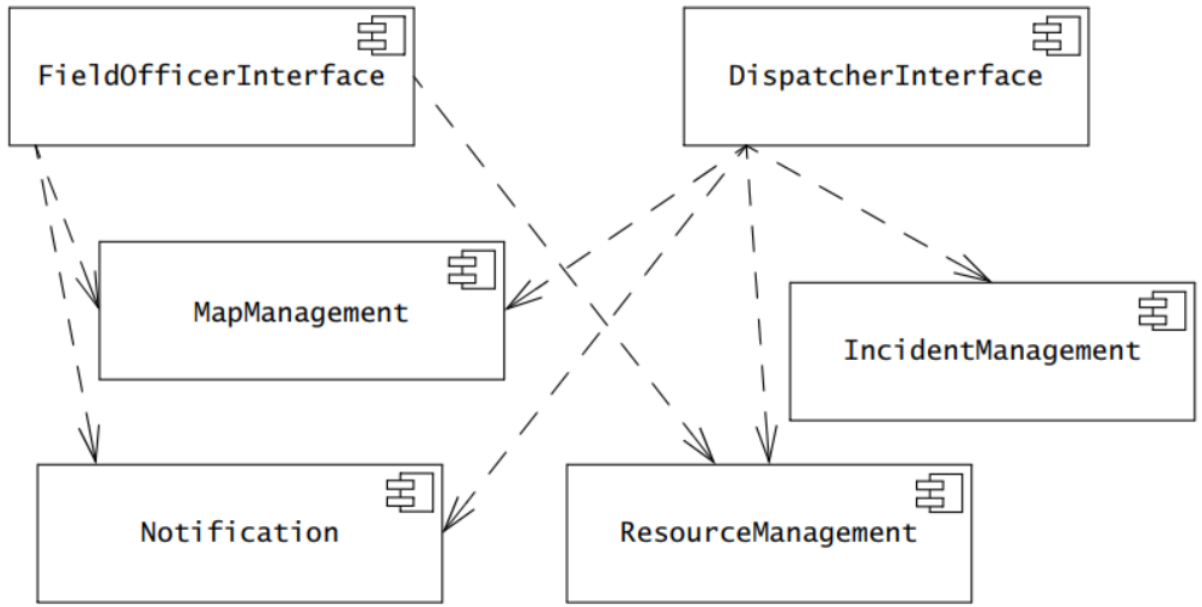
Camadas e partições (4)

- ✧ Arquiteturas fechadas e em camadas têm propriedades desejáveis: levam a um baixo acoplamento entre subsistemas, e os subsistemas podem ser integrados e testados de forma incremental.
- ✧ Cada nível, no entanto, introduz uma sobrecarga de velocidade e armazenamento que pode dificultar o atendimento de requisitos não funcionais



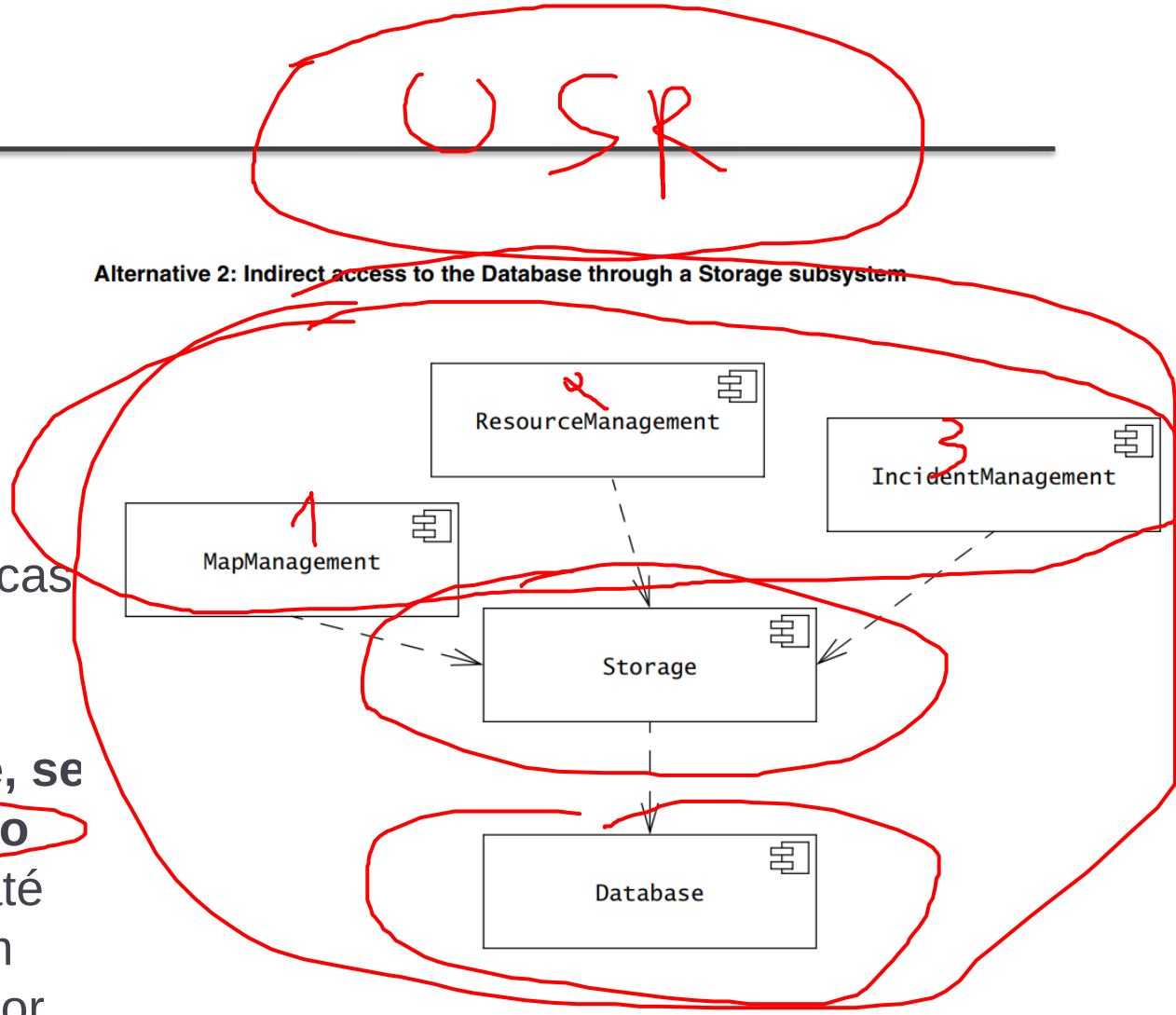
Camadas e partições (5)

- Outra abordagem para lidar com a complexidade **é particionar o sistema em subsistemas de mesmo nível**, cada um responsável por uma categoria de serviços diferente.
- Cada subsistema depende livremente dos outros, mas geralmente pode operar isoladamente.



Camadas e partições (5)

- **Em geral, uma decomposição de subsistema é o resultado de particionamento e camadas**.
  - Primeiro, **particionamos o sistema em subsistemas de nível superior**, responsáveis por funcionalidades específicas ou executados em um nó de hardware específico.
  - Cada um dos **subsistemas resultantes é, se a complexidade o justificar, decomposto em camadas de nível inferior** e inferior até que sejam simples o suficiente para serem implementadas por um único desenvolvedor.
  - **Cada subsistema adiciona uma certa sobrecarga** de processamento devido à sua interface com outros sistemas.



Questões (1)

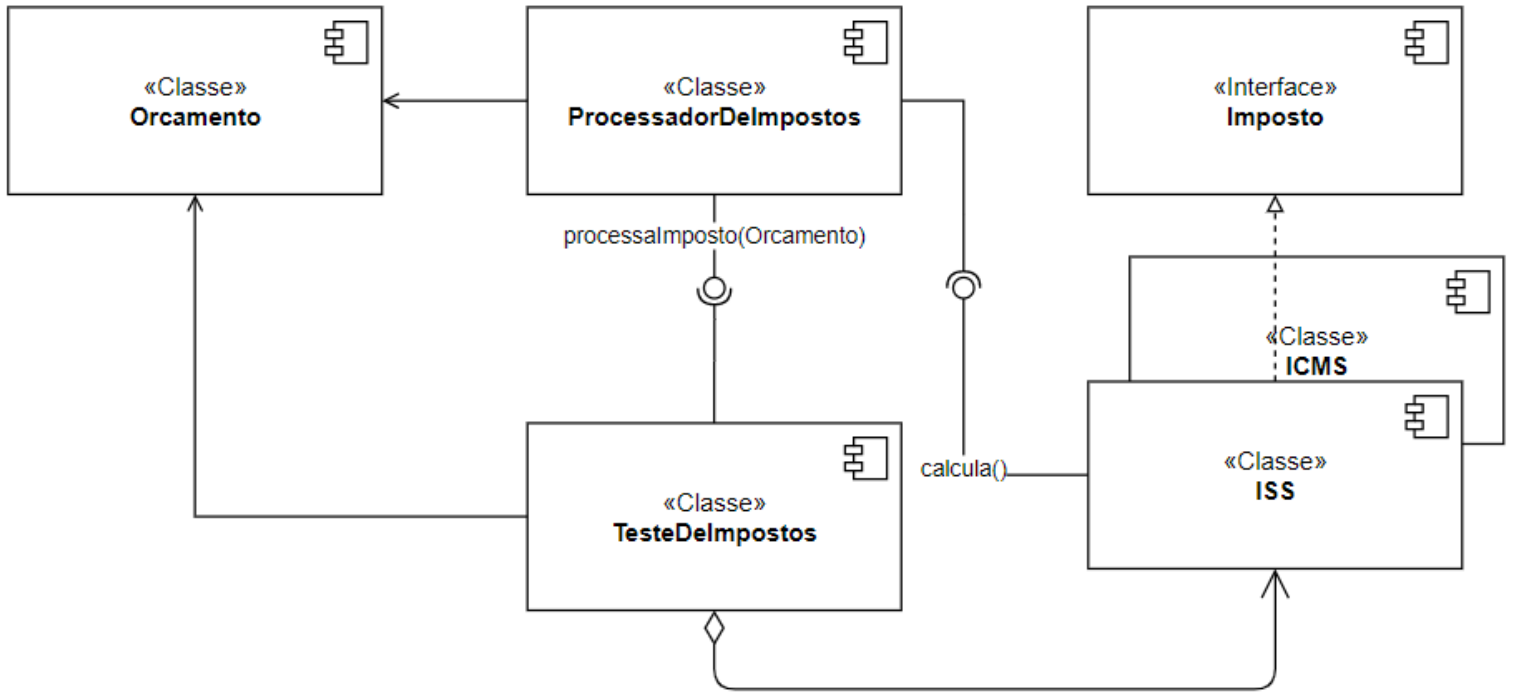
1. Compare e contraste acoplamento e coesão.
2. (Extra classe) Pesquise e defina acoplamento de classe, acoplamento de controle e acoplamento externo.

Atividade prática

- ✧ Análise de acoplamento e coesão pelo levantamento de diagrama UML de uma aplicação de cálculo de impostos.
- ✧ Arquivos da aplicação serão disponibilizados no Blackboard.

Atividade prática (Resposta)

- Análise de acoplamento e coesão pelo levantamento de diagrama UML de uma aplicação de cálculo de impostos.
- Arquivos da aplicação serão disponibilizados no Blackboard.
- Resposta:
  - O acoplamento quantitativo máximo é 2 visto que Orcamento e Imposto são dependências para duas classes diferentes.
  - O acoplamento pode ser considerado qualitativamente baixo pois a quantidade máxima de dependências é baixa e metade dela trata apenas de métodos construtores de objetos.



Tópicos vistos

- O que é Arquitetura de Software?
- Como se desenha bons softwares?
- O que é decomposição de sistema?
- O que é Subsistema?
- O que são serviços?
- O que é acoplamento?
- O que é coesão?
- O que são camadas?
- O que são partições?

Referências

- BRUEGGE, B; DUTOIT, A. Object-oriented software engineering: using UML, patterns and Java. Harlow: Pearson Education, 2014.
- PRESSMAN, R. Engenharia de Software. [Recurso eletrônico, Minha Biblioteca]. 8ª ed. BOOKMAN, 2016.
- SOMMERVILLE, I. Engenharia de Software. [Recurso eletrônico, Biblioteca Virtual Universitária 3.0]. 9ª ed. SARAIVA, 2011
- DEITEL, P. Java: Como Programar. Pearson, 2016.