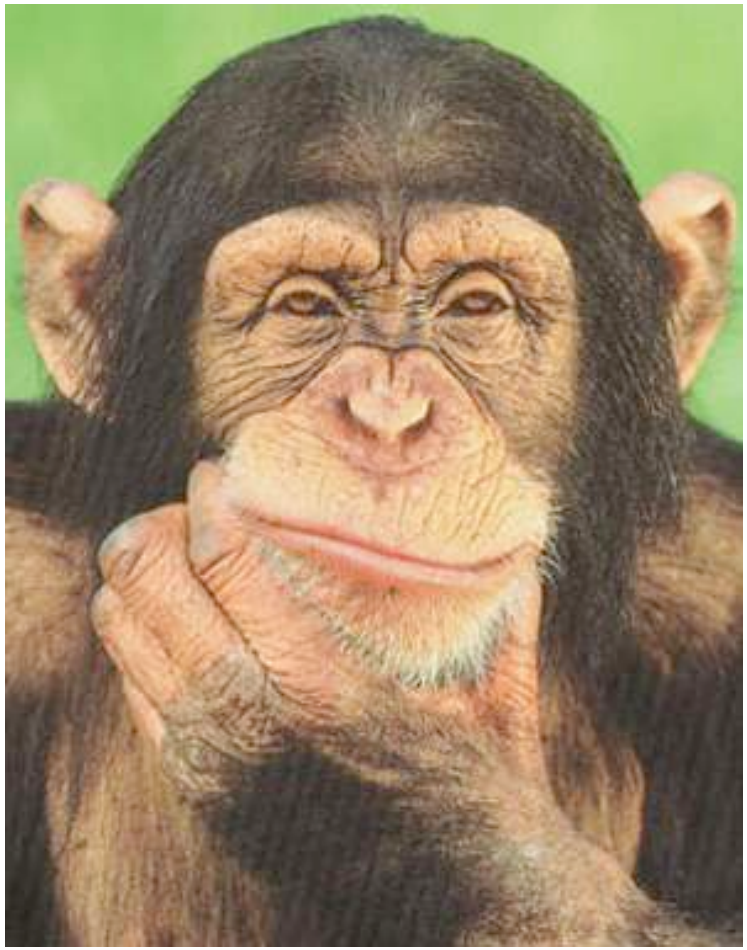
O que é modularidade?

Sérgio Soares
scbs@cin.ufpe.br

AOSD

- Aspectos tem como objetivo aumentar a modularidade dos sistemas...
- ... mas

O que é modularidade???



Parnas, 1972

"modularization is a mechanism for improving the flexibility and comprehensibility of a system while allowing the shortening of its development time"

D.L. Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM, Vol. 15, No. 12, pp. 1053 – 1058, 1972.

Modularidade

- O critério utilizado para dividir o sistema em módulos torna mais ou menos efetiva a modularidade

Ainda Parnas....

1970

A well-defined segmentation of the project effort ensures system modularity. Each task forms a separate, distinct program module. At implementation time each module and its inputs and outputs are well-defined, there is no confusion in the intended interface with other system modules. At checkout time the integrity of the module is tested independently; there are few scheduling problems in synchronizing the completion of several tasks before checkout can begin. Finally, the system is maintained in modular fashion, system errors and deficiencies can be traced to specific system modules, thus limiting the scope of detailed error searching.

Gauthier, Richard; Pont, Stephen. Designing Systems Programs, (C), Prentice-Hall, Englewood Cliffs, 1970.

OK, mas, e os critérios?

- Como separar o sistema em módulos?
- Avanços em programação modular (na década de 70) eram:
  1. allow one module to be written with little knowledge of the code in another module
  2. allow modules to be reassembled and replaced without reassembly of the whole system.

Não é tão simples assim!

- Os exemplos de sistemas problemáticos eram exatamente os altamente modulares!!!
  - Que utilizavam as técnicas mencionadas no slide anterior

Que benefícios esperar da modularidade?

- Gerencial
  - Menor tempo de desenvolvimento (grupos separados trabalhando em paralelo com pouca comunicação)
- Flexibilidade
  - Possibilidade de realizar mudanças drásticas em um módulo sem mudar outros módulos
- Compreensão
  - Deve ser possível estudar o sistema um módulo por vez

Modularização/Decomposição

- Certas decisões de negócio devem ser tomadas antes de trabalhar nos módulos
- Avaliar alternativas considerando decisões que impactam no sistema (em mais de um módulo)

Critérios para decomposição

- Passos do processamento / Flow chart
  - Editor de texto
    - entrada, operação 1, operação 2, saída
- Information hiding
  - Editor de texto
    - armazenamento, interface com o usuário (E/S), operação 1, operação 2

Quais os impactos de mudanças no meio de armazenamento?

Information hiding e um guardanapo

- O "guardanapo da desgraça"

David L. Parnas. The secret history of information hiding. Software pioneers: contributions to software engineering, p.p 399 - 409, Springer-Verlag, 2002.

Qual o segredo da modularidade?

- Definir interfaces que sejam "sólidas"
  - Módulos devem encapsular informações que tendem a variar
    - esconder detalhes e possíveis mudanças
  - Interfaces devem ser duradouras!
    - Não deveriam ser complexas, mas "beautiful"

Information hiding é o segredo!

Information hiding é difícil

- Quanto tempo investir em avaliar o que pode mudar no futuro?
  - Como justificar o investimento?
  - time-to-market
- Software baseado em versões anteriores mal projetadas
- Manutenções podem deteriorar o projeto

Information hiding

- É a base para
  - Tipos de dados abstratos
  - Orientação a objetos
  - Desenvolvimento baseado em componentes

Como medir modularidade?

- Coesão
  - Como calcular?
- Acoplamento
  - Como calcular?
- Separação de interesses
  - Como calcular?

Coesão

Chidamber, S.R.; Kemerer, C.F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, Volume 20, Issue 6, p.p 476 – 493, 1994.

- Coesão em métodos
  - Relação entre os atributos utilizados pelos métodos de uma classe
  - Métodos que utilizam diferentes conjuntos de atributos não tem coesão
  - Falta de coesão = MNC – MC
    - MNC – métodos não coesos
    - MC – métodos coesos

Acoplamento

- Considere uma classe A que usa/ acessa uma classe B

  Se colocarmos uma interface entre as duas classes a métrica aumenta?

- Acoplamento entre classes
  - Número de classes referenciadas por uma classe

Chidamber, S.R.; Kemerer, C.F. A metrics suite for object oriented design. IEEE Transactions on Software Engineering, Volume 20, Issue 6, p.p 476 – 493, 1994.

Separação de interesses

- Em AspectJ um aspecto pode implementar todo o código referente a um interesse transversal
  - Mas o código do aspecto pode fazer referência a
    - interfaces (OK!)
    - classes (hummmm....)
    - métodos concretos (ops...)
    - atributos (danou-se)
    - membros privados (agora ferrou de vez)

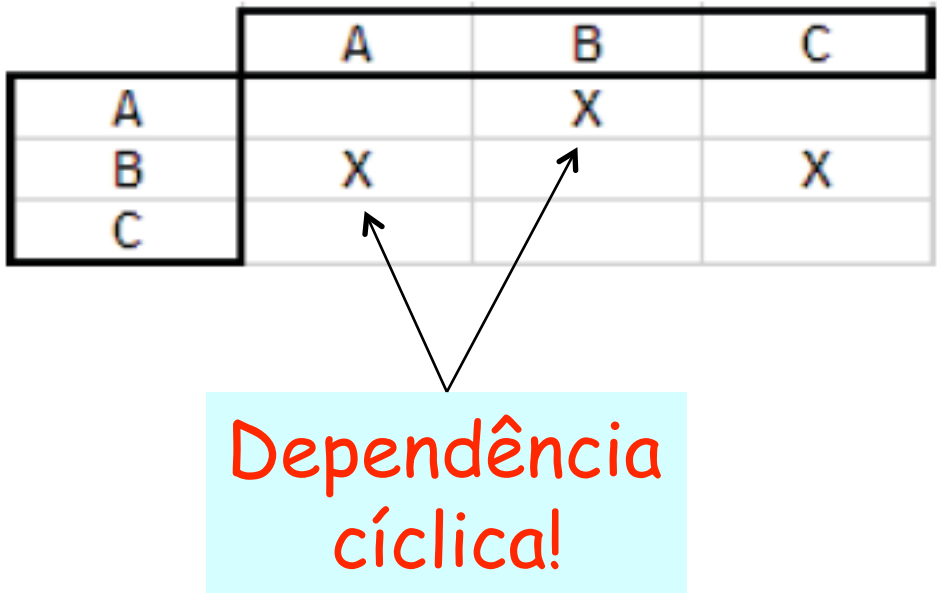
Ou seja

- Aspectos (interesses transversais) diminuem a modularidade de classe
  - Para evoluir classes é necessário olhar para os aspectos que a modificam
    - Deveríamos olhar apenas a própria classe e as interfaces das outras classes que a mesma utiliza

Pontos de vistas diferentes sobre quais são os módulos de um sistema. Cada classe é um módulo?

Design Structure Matrix (DSM)

- Visualização de dependências entre parâmetros (classes, casos de uso, aspectos) de uma aplicação



Design rules

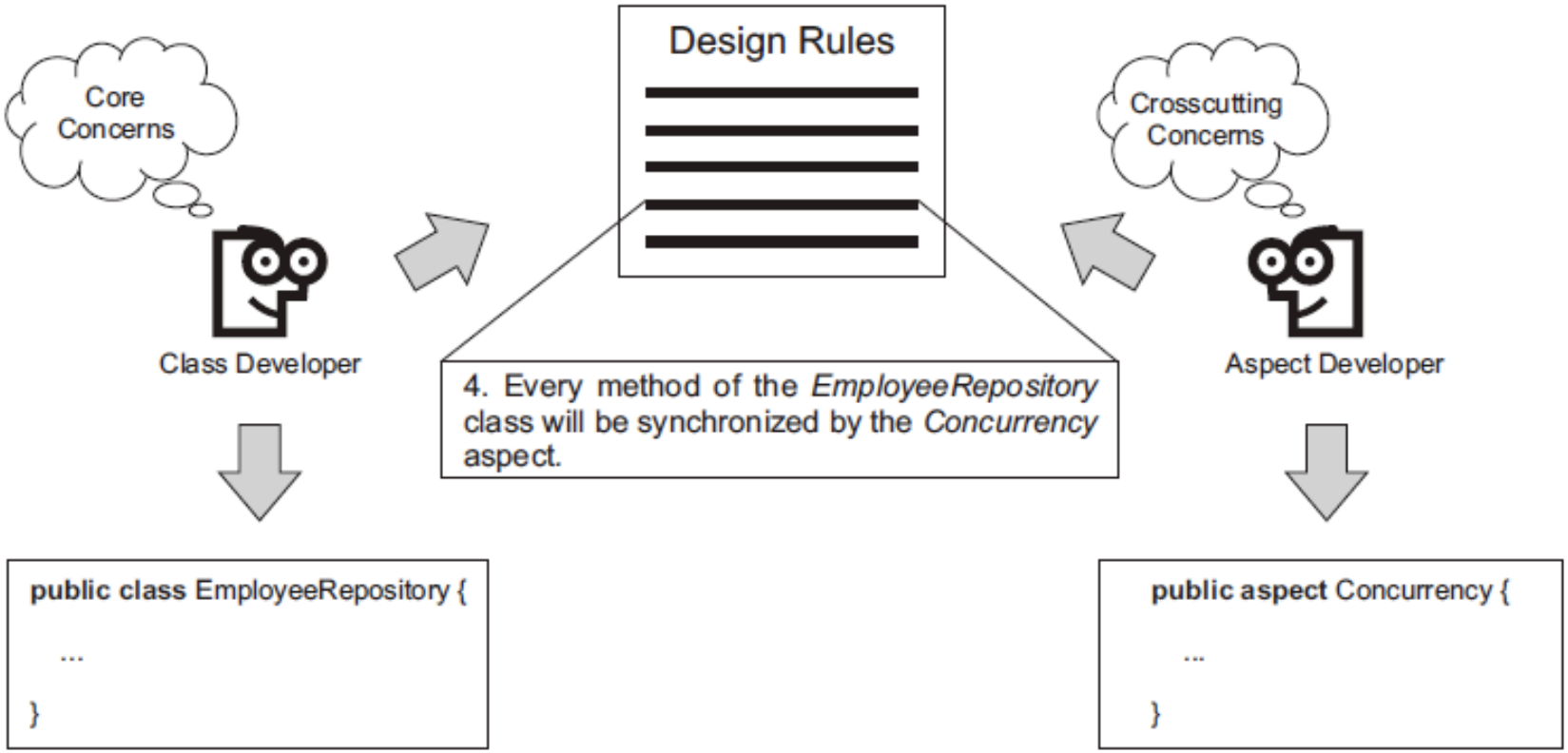
- Um tipo de interface que define restrições que classes satisfazem e que limitam/coordenam o efeito dos aspectos sobre essas classes
  - Join points necessários
  - Nomes de membros
  - Pre/pós-condições

O objetivo é o mesmo de Parnas: permitir que módulos evoluam sem impactar em outros módulos

Márcio Ribeiro et. al. Analyzing Class and Crosscutting Modularity with Design Structure Matrixes. SBES 2007.

Design rules

- Interfaces entre componentes
  - Desacopla componentes



DSM – Versão OO

Design Parameters			1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
	Constraints and Requirements	1																					
Use Cases	Login	2	x	Architectural																			
	Register new employee	3	x																				
	Register new complaint	4	x																				
	Query information	5	x				Application																
	Update employee	6	x																				
	Update complaint	7	x																				
	Update health unit	8	x																				
	Register tables	9	x								Utility												
	Change logged employee	10	x																				
Architectural Decisions	Select architectural style and patterns	11	x	x	x	x																	
	GUI technology	12	x	x	x	x							x										
	Security mechanism	13	x	x	x	x							x										
	Persistence	14	x	x	x	x							x										
	Distribution mechanism	15	x	x	x	x							x										
	Concurrency mechanism	16	x	x	x	x							x										
	Transaction mechanism	17	x	x	x	x							x										
Components	Employee	18			x		x	x					x	x	x	x	x	x	x				
	Health Unit	19					x			x	x		x	x	x	x	x		x				
	Complaint	20				x	x		x				x	x	x	x	x	x	x	x	x		
	Authentication	21		x								x	x	x	x	x	x		x	x			

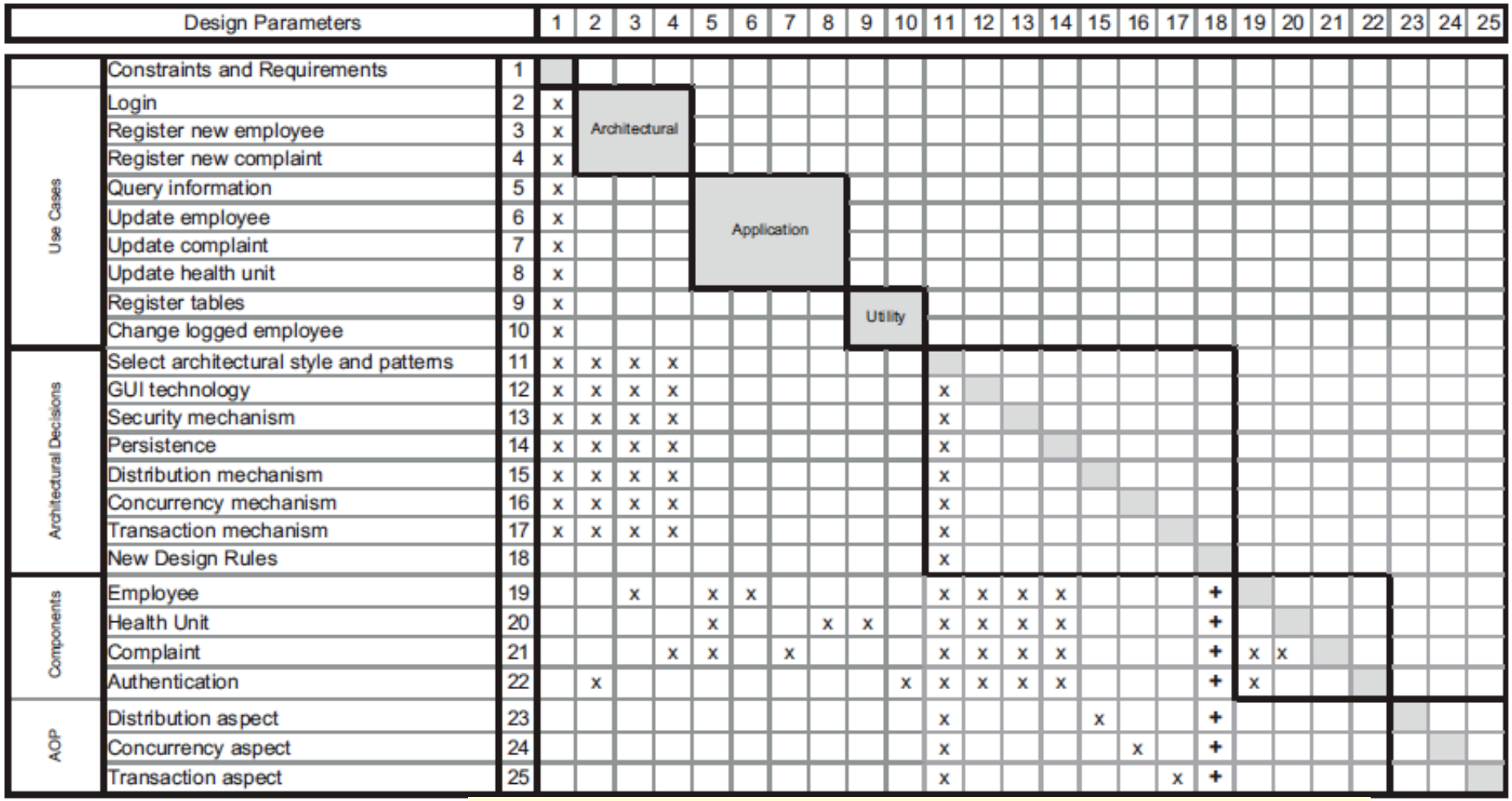
Márcio Ribeiro et. al. Analyzing Class and Crosscutting Modularity with Design Structure Matrixes. SBES 2007.

DSM – Versão AO

	Design Parameters		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
	Constraints and Requirements	1																								
Use Cases	Login	2	x	Architectural																						
	Register new employee	3	x																							
	Register new complaint	4	x																							
	Query information	5	x				Application																			
	Update employee	6	x																							
	Update complaint	7	x																							
	Update health unit	8	x																							
	Register tables	9	x								Utility															
	Change logged employee	10	x																							
Architectural Decisions	Select architectural style and patterns	11	x	x	x	x																				
	GUI technology	12	x	x	x	x							x													
	Security mechanism	13	x	x	x	x							x													
	Persistence	14	x	x	x	x							x													
	Distribution mechanism	15	x	x	x	x							x													
	Concurrency mechanism	16	x	x	x	x							x													
	Transaction mechanism	17	x	x	x	x							x													
Components	Employee	18			x		x	x					x	x	x	x										
	Health Unit	19					x			x	x		x	x	x	x										
	Complaint	20				x	x		x				x	x	x	x				x	x					
	Authentication	21		x								x	x	x	x	x				x						
AOP	Distribution aspect	22											x				x			x	x	x	x			
	Concurrency aspect	23											x					x		x		x				
	Transaction aspect	24											x						x	x	x	x	x			

Márcio Ribeiro et. al. Analyzing Class and Crosscutting Modularity with Design Structure Matrixes. SBES 2007.

DSM – Versão AO + DR



Márcio Ribeiro et. al. Analyzing Class and Crosscutting Modularity with Design Structure Matrixes. SBES 2007.

Resumindo

- Aspectos precisam evoluir para atingir seu objetivo: MODULARIDADE
- DR auxiliam a modularidade de classe e aspecto
- Desenvolver linguagens que garantam a implementação das DR
- Isso se aplica a todo e qualquer tipo de sistema?

CLARO QUE NÃO!

Software Productivity Group

http://www.cin.ufpe.br/

Bibliografia

- Gauthier, Richard; Pont, Stephen. Designing Systems Programs, (C), Prentice-Hall, Englewood Cliffs, 1970.
- Parnas, David L. On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM, Vol. 15, No. 12, pp. 1053 – 1058, 1972.
- Parnas, David L. The secret history of information hiding. Software pioneers: contributions to software engineering, p.p 399 - 409, Springer-Verlag, 2002.
- Ribeiro, Márcio et. al. Analyzing Class and Crosscutting Modularity with Design Structure Matrixes, SBES 2007.