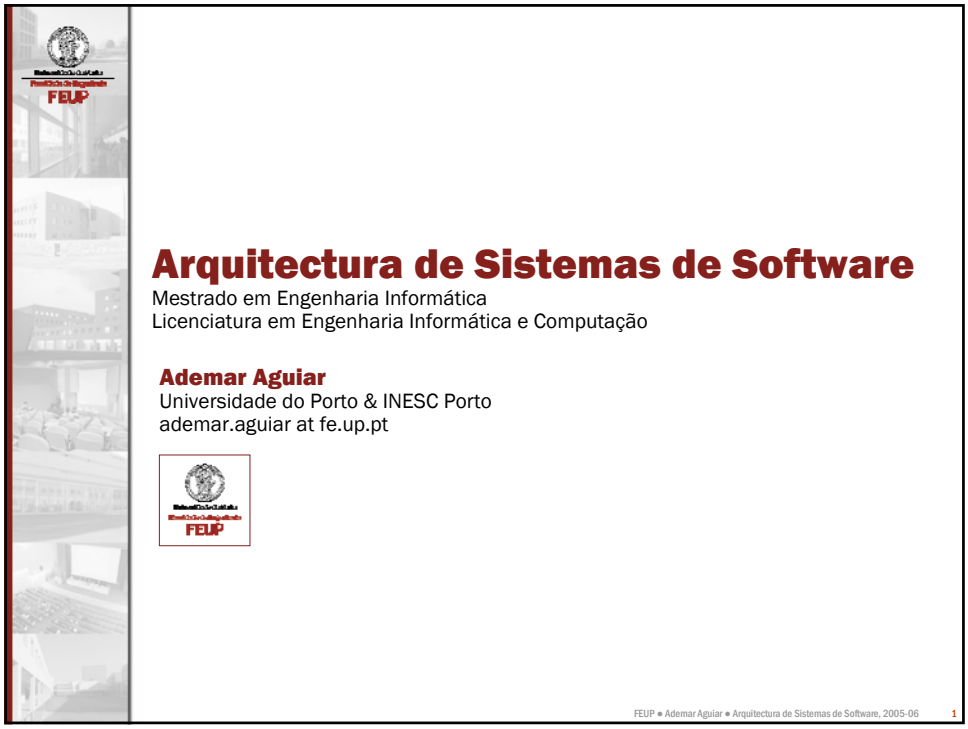

FEUP
Arquitectura de Sistemas de Software
Mestrado em Engenharia Informática
Licenciatura em Engenharia Informática e Computação
Ademar Aguiar
Universidade do Porto & INESC Porto
ademar.aguiar at fe.up.pt
FEUP
FEUP ● Ademar Aguiar ● Arquitectura de Sistemas de Software, 2005-06 1


FEUP
Reutilização de Software, Padrões e Frameworks
FEUP ● Ademar Aguiar ● Arquitectura de Sistemas de Software, 2005-06 2

Reutilização: a (única) alternativa!

- A produtividade de software aumentou significativamente nas últimas três décadas, mas ainda não o suficiente para satisfazer a procura colocada no mercado.
- O factor chave para a produtividade de software é o número de instruções que têm que ser escritas para implementar uma determinada funcionalidade.
- "Em vez de procurar formas de escrever código mais rapidamente, deve-se procurar formas de escrever menos código" [Boehm 1987].
- Reutilização de software é a actividade de utilizar artefactos de software existentes para o desenvolvimento de novos sistemas.
- "A reutilização é considerada a (única) abordagem realista para se conseguir alcançar os níveis de produtividade que a indústria necessita" [Mili et al., 1995].

Actividades típicas

- Abstracção
  - Criação de abstracções para os artefactos reutilizáveis.
  - As abstracções permitem ao reutilizador perceber para que servem os artefactos, quando podem ser reutilizados e como reutilizá-los.
- Selecção
  - Compreende a classificação, pesquisa, comparação, compreensão e escolha de artefactos.
- Especialização
  - Configuração dos artefactos à situação em mãos.
- Integração
  - Combinação de colecções de artefactos num sistema completo.

Abordagens

- Existem dois tipos de abordagens:
  - *Composicionais*: baseiam-se na reutilização de *artefactos*
  - *Geradores*: baseiam-se na reutilização de *processos*
- Exemplos de artefactos reutilizáveis:
  - Código-fonte
  - Componentes (pequenos)
  - Padrões (*patterns*)
  - Infraestruturas aplicacionais (*frameworks*)
  - Componentes de domínio (grandes)
  - Linhas de produção de aplicações (*product lines*)
  - Especificações abstractas
  - Documentação
  - ...

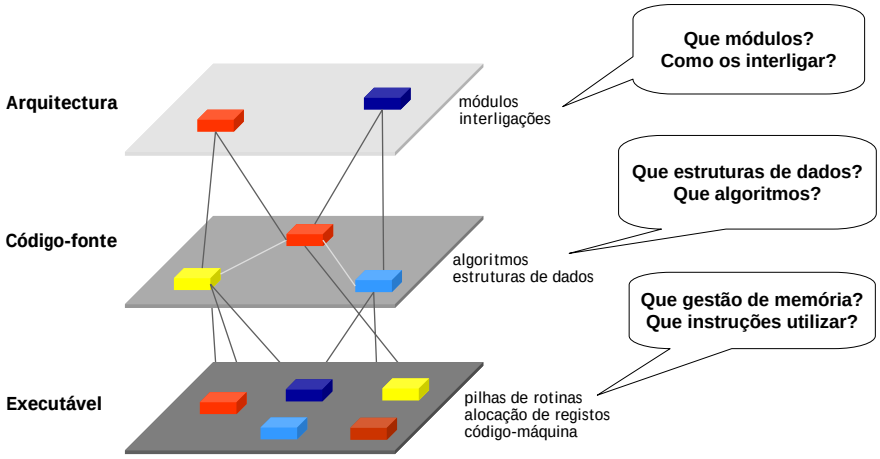
Reutilização de Código vs Desenho

- Reutilização de código
  - É fácil, mas tem limitações.
  - É adequado quando o domínio de problema é pequeno, bem compreendido e a tecnologia de base é bastante estática.
- Reutilização de desenho
  - A médio-longo prazo é melhor em termos económicos.
  - O desenho é o principal conteúdo intelectual do software, sendo muito mais difícil de criar e re-criar do que o código.
  - A reutilização ao nível de desenho é uma das mais importantes formas de reutilização existentes.
  - Padrões, frameworks e arquitecturas são exemplos de abordagens que permitem a reutilização ao nível de desenho.

Desenho de Software

- Qualquer sistema pode ser visto a diferentes níveis de abstracção.
- A cada nível podem-se encontrar componentes, regras de composição, regras de funcionamento, e diferentes notações, técnicas e problemas.
- Os elementos constituintes dos vários níveis estão interrelacionados pois são desenvolvidos com base em construções do nível inferior.
- Em software podem-se identificar pelo menos três níveis: *arquitectura*, *código-fonte* e *código executável*

Desenho de Software: níveis principais



Padrões e Frameworks

- Padrões de Desenho
  - *Micro-arquitecturas* que descrevem mecanismos abstractos de cooperação entre objectos que permitem resolver um problema recorrente de desenho.
- Frameworks
  - *Macro-arquitecturas* que interligam diversos padrões e que também incluem normalmente a infraestrutura que suporta a sua integração.
  - São conjuntos de classes cooperantes, concretas e abstractas, que foram desenhadas para implementar funcionalidade específica a um determinado domínio de problemas.
  - As frameworks possuem partes inalteráveis e partes configuráveis pelo utilizador através de mecanismos de herança (*white-box*) e/ou composição (*black-box*).

Desenho é arte ou ciência?

- Não só ciência!
  - Os métodos analíticos são difíceis de aplicar, devido à enorme dimensão do espaço de soluções possíveis e também devido à ausência de critérios quantitativos de selecção.
- Não só arte!
  - Bons arquitectos "(re)utilizam" soluções que resultaram bem anteriormente e adaptam-nas ligeiramente e incrementalmente a novas situações.
  - Conhecer, imitar, adaptar e inovar são actividades importantes.
- Intuição e criatividade são portanto "ainda" qualidades muito importantes no desenvolvimento de software ☺

Padrões de Software

FEUP Universidade do Porto Faculdade de Engenharia

Arquitectura de Sistemas de Software, LEIC/MEI,

Cristopher Alexander

- A noção de "padrões" tem as suas origens no trabalho do arquitecto Cristopher Alexander.
- Durante 10 anos, Alexander recolheu e documentou soluções genéricas para problemas recorrentes no domínio da arquitectura.
- O objectivo inicial foi o de habilitar não-especialistas a projectar as suas próprias casas e comunidades.
- O resultado deste trabalho originou vários livro
  - *"A Pattern Language"*
  - *"The Timeless Way of Building"*
  - *"The Oregon Experiment"*
  - ...
  - *"The Nature of Order", vol. I, II, III, IV.*

- Ver mais em http://c2.com/cgi/wiki?ChristopherAlexander

FEUP ● Ademar Aguiar ● Arquitectura de Sistemas de Software, 2005-06 12

"A Pattern Language" [Alexander77]

- 253 padrões
  - O livro apresenta os padrões de Alexander, i.e., descrições textuais de soluções para problemas recorrentes em arquitectura civil.
- Padrão de Alexander

  "Each pattern describes a problem that occur over and over again in our environment and then describes the core of the solution to that problem in such a way that you can use this solution a million times over without ever doing it the same way twice" [Alexander77, p. x]

- Definição
  - Os padrões são uma descrição *textual* de uma *solução genérica* para um *problema recorrente* num determinado *contexto*.

"A Pattern Language" ...

- "the quality without a name" (TQWAN)
  - O objectivo de Alexander neste trabalho de pesquisa de padrões, consistiu em identificar o que distingue uma construção com 'qualidade' de uma outra.
  - Segundo Alexander, as agradáveis sensações que uma construção nos transmite de liberdade, conforto, harmonia e vida são tudo reflexos da presença ou não desta multifacetada 'qualidade'.
  - A linguagem de padrões que documentou em livro auxilia na detecção dos elementos que contribuem para o aparecimento ou ausência desta 'qualidade', referida por TQWAN.

"A Pattern Language": exemplos

- 253 padrões: apresentados do global para o particular
  - Em "A Pattern Language" são descritos 253 padrões, interrelacionados, que variam no nível de detalhe, sendo a sua apresentação iniciada pelos padrões de nível mais global e seguindo depois para os de nível mais particular.
- Alguns exemplos destes 253 padrões:
  - 1. Independent Regions
  - 2. The Distribution of Towns
  - 16. Web of Public Transportation
  - 83. Master and Apprentices
  - 134. Zen View
  - 251. Different Chairs
  - 253. Things from your life

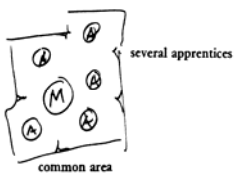
"83. Master and Apprentices"

- Problema

  "The fundamental learning situation is one in which a person learns by helping someone who really knows what he is doing." [Alexander77, p. 413]

- Solução

  "Arrange the work in every workgroup, industry, and office, in such a way that work and learning go forward hand in hand. Treat every peace of work as an opportunity for learning. To this end, organize work around a tradition of masters and apprentices: and support this form of social organization with a division of the workspace into spacial clusters - one for each master and his apprentices - where they can work and meet together." [Alexander77, p. 1159]



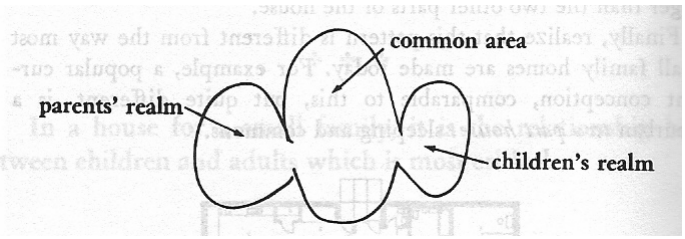
"76. House for a Small Family"

- Problema

  "In a house for a small family, it is the relationship between children and adults which is most critical." [Alexander77, p. 382]

- Solução

  "Give the house three distict parts: a realm for parents, a realm for children, and a common area. Conceve these three realms as roughly similar in size, with the commons the largest." [Alexander77, p. 384]



"137. Children's Realm"

- Problema

  "If children do not have space to release a tremendus amount of energy when they need to, they will drive themselves and everybody else in the family up the wall." [Alexander77, p. 652]

- Solução

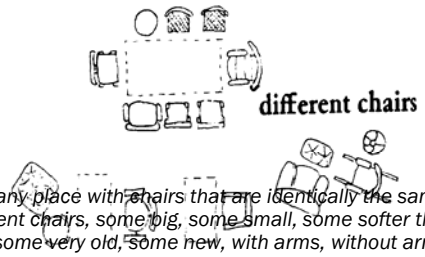
  "Start y placing the small area which will belong entirely to the children-the cluster of their beds. Place it in a separate position toward the backof the house (...)." [Alexander77, p. 654]

"251. Different Chairs"

- Problema

  "People are different sizes: they sit in different ways. And yet there is a tendency in modern times to make all chairs alike." [Alexander77, p. 1158]

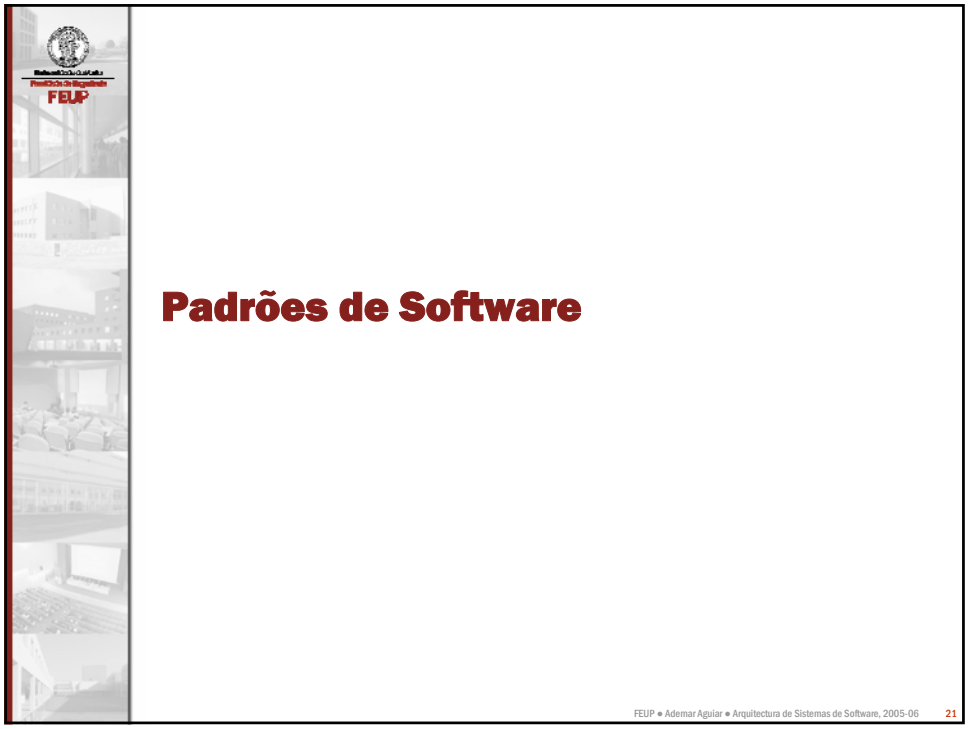
- Solução



"Never furnish any place with chairs that are identically the same. Choose a variety of different chairs, some big, some small, some softer than others, some rockers, some very old, some new, with arms, without arms, some wicker, some wood, some cloth." [Alexander77, p. 1159]

Formato dos Padrões de Alexander

Nome	Uma frase ou nome descritivo identificativo do padrão.
Exemplo	Fotografias, desenhos e descrições que ilustram a aplicação dos padrões.
Contexto	As circunstâncias nas quais o padrão pode ser aplicado. Explica a razão da existência do padrão e a sua generalidade.
Problema	Descreve as forças relevantes e restrições e como interactuam entre si.
Solução	Relações e regras dinâmicas que descrevem como construir artefactos em concordância com o padrão. São referidos padrões similares e variantes, bem como padrões de níveis superior e inferior relevantes para a concepção da solução proposta.



Padrões de Software

- Existem problemas e soluções recorrentes também em Engenharia de Software:
  - Arquitectura
  - Análise
  - Desenho
  - Codificação
- "GoF book": 1º livro sobre Padrões de Software
  - Foi apresentado na OOPSLA'94, e é uma co-autoria de um grupo de quatro indivíduos conhecido pelo *Gang of Four* (GoF).
  - Documenta 23 padrões que descrevem soluções reconhecidamente boas para problemas recorrentes de desenho orientado por objectos [Gamma95].

FEUP ● Ademar Aguiar ● Arquitectura de Sistemas de Software, 2005-06 22

O que é um Padrão de Software?

- Uma definição:
  - "Um padrão de software descreve um problema recorrente num contexto específico e uma solução genérica comprovadamente boa para o resolver. A solução descreve os componentes que a constituem, as suas responsabilidades, as associações e os mecanismos de colaboração entre eles" [Buschmann96]
- Alguns exemplos bem populares:
  - Observer (Model-View-Controller)
  - Iterator: para abstrair a navegação em estruturas de dados ...
  - Broker: para estruturar sistemas distribuídos ...
  - Proxy: para controlar acessos, melhorar eficiência ...
  - Composite: para representar hierarquias de agregação...
  - Command: para encapsular um pedido num objecto...
  - Strategy: para encapsular um algoritmo num objecto...
  - Singleton: para implementar classes com uma única instância...

Forma dos Padrões de Software

Nome	Nome sugestivo do que o padrão representa
Exemplo	Exemplo de instanciação do padrão.
Contexto	Enquadramento das situações onde o problema ocorre.
Problema	Descrição e enumeração das forças em conflito.
Solução	Descrição abstracta da estrutura e principais colaborações.
Consequências	Efeitos positivos e negativos de aplicação do padrão.
Outros Padrões	Padrões relacionados com importância para a solução.
Utilizações	Pelo menos três utilizações bem conhecidas.

Principais Tipos de Padrões

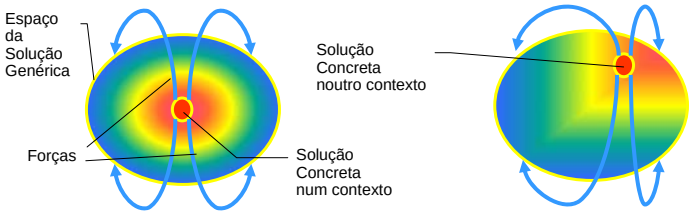
- "Design Patterns"
  - Os primeiros a merecer atenção alargada
  - Incidem sobre soluções para problemas de desenho
  - Popularizados pelo [Gamma95] e [Buschmann96]
- "Architectural Patterns"
  - Fornecem orientações em como estruturar sistemas e subsistemas de software com o intuito de lhes conferir determinadas propriedades
- "Idioms"
  - Padrões de um nível de abstracção mais baixo por serem específicos a uma determinada linguagem de programação
  - Descrevem como implementar aspectos particulares recorrendo às capacidades específicas de uma linguagem

Padrões de Software

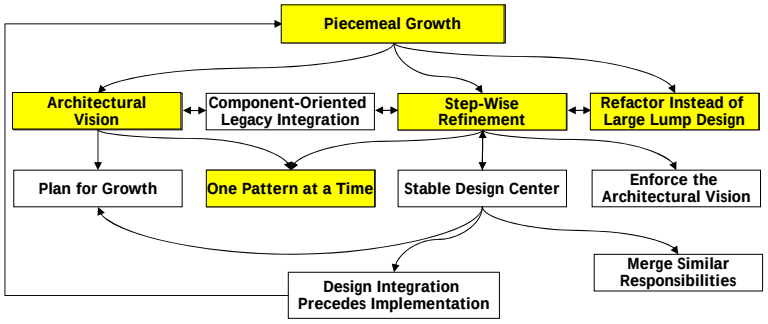
- Uma forma de partilhar experiência
  - Provaram ser uma excelente forma de partilhar experiência e um óptimo veículo de comunicação de conhecimentos em engenharia de software.
- Documentam soluções de qualidade para problemas típicos
  - Os padrões de software documentam soluções de comprovada qualidade para uma grande diversidade de problemas de desenvolvimento software
- O património actual é bastante grande e continua a aumentar...
  - Existem hoje publicados inúmeros padrões de diversos tipos: padrões de arquitectura de software, padrões de desenho de software e padrões de análise.
- Da curiosidade à familiaridade
  - Gradualmente, os padrões de software têm deixado de ser vistos como uma simples curiosidade e têm evoluído para algo bastante familiar e que se utiliza todos os dias quando se desenvolve software.
- Questão: Como e quando os aplicar?

Instanciação de Padrões

- Os padrões fornecem soluções em abstracto
  - Têm de ser adaptados e instanciados ao contexto específico em que vão ser aplicados
  - Na instanciação normalmente recorre-se a outros padrões
- Os padrões facilitam a avaliação de compromissos
  - Expõem como uma solução equilibra as forças envolvidas
  - Auxiliam a tomar decisões de desenho e de implementação



"Building Software with Patterns"



[Buschmann99]

Principais Qualidades dos Padrões

- Veiculam sabedoria entre quem desenvolve software
  - Os padrões de software permitem documentar e transmitir soluções comprovadas para a resolução de problemas típicos de desenvolvimento de software.
- Vocabulário Comum
  - Os padrões oferecem um vocabulário e um entendimento comum sobre desenhos de software.
- Aumentam a "largura de banda"
  - Os nomes dos padrões, se bem escolhidos, podem ser utilizados como abreviaturas para estruturas de desenho complexas, que aumentam assim a "*largura de banda*" entre quem desenvolve software.
- Uma forma mais rápida de aprender na prática ...
  - Estes conhecimentos e experiências podem depois de publicados serem amplamente conhecidos e utilizados pelos menos experientes (mas com sentido crítico!).

Exemplo de um Padrão de Arquitectura: Layers Pattern

Padrão Layers (POSA 31)

- O que é o padrão layers?
- Forças
- Exemplos
- Dinâmica
- Variantes
- Utilizações conhecidas
- Consequências

O que é o padrão Layers?

- É um padrão de arquitectura
- Decompõe um sistema em grupos de responsabilidades --> diferentes níveis de abstracção
- Combina questões de baixo nível e de alto nível
- Decomposição vertical
- Decomposição horizontal

Forças

- Modificações de código tardias
- Componentes intermutáveis
- Modularidade
  - módulos fáceis de compreender
  - módulos de fácil manutenção
- Coerência
  - integridade
- Desenvolvimento paralelizável

Exemplo: I/O

Layers of I/O Software System

Ref: Tanenbaum, A. Modern Operating Systems, Prentice Hall, 2001.

Exemplo UMIC/FSC

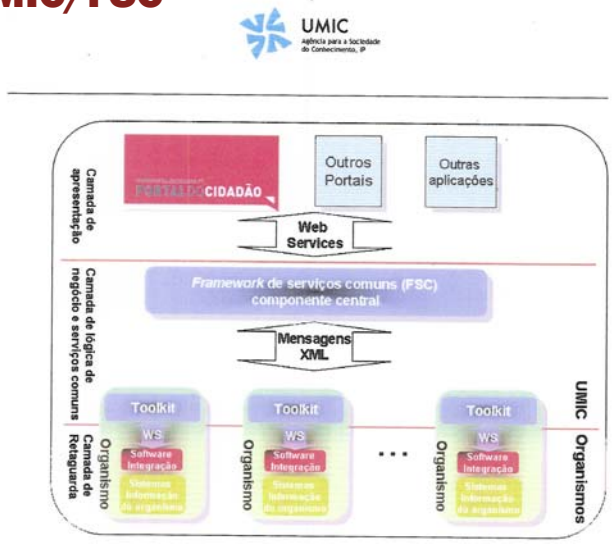


Figura 1 – Arquitectura de Referência

FEUP Universidade do Porto Faculdade de Engenharia

Forças

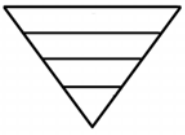
- Portabilidade das aplicações
- Fácil desenvolvimento e distribuição das aplicações

Dinâmica

- Pedidos Top -> Bottom
  - fan out
- Notificações Bottom -> Top
  - condense / 1:1
- Podem ser usados mecanismos de caching (em ambas as direcções)

Dinâmica

- Pirâmide de reutilização invertida
  - É melhor para localizar serviços em camadas superiores do que em camadas inferiores.
  - Os componentes nas camadas inferiores devem ser específicos.
  - Permite a adição de novos serviços alto-nível em cima de primitivas existentes.
  - Fomenta a facilidade de desenvolvimento
  - RUP



Dinâmica

- Abordagem Iô-iô
  - Passos de desenvolvimento *top-down* e bottom-up até de camadas estáveis e naturais
  - Refinamento iterativo
  - Refactoring e eXtreme Programming (XP)

Dinâmica

- Black / White / Gray box
  - Façade
- Tratamento de erros
  - Propagação das mensagens de erro
  - São lançadas excepções
- Acoplamento
  - Callbacks / Observer pattern
  - Command Pattern
  - Polling
  - Depende de quem está a desenvolver as camadas

Variantes

- Relaxed
- Opaque (partially relaxed)
- Layering através de herança
  - Problema “Fragile base class”
  - Herança múltipla

Utilizações conhecidas

- Protocolos de rede
- APIs
- Sistemas operativos
- Máquinas Virtuais
  - Arquitectura em camadas ou uma camada?

Consequências

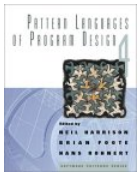
- Vantagens
  - Reutilização -> reduz esforço de desenvolvimento & melhora robustez
  - Normalização
  - Localidade das alterações
  - Mais fácil de testar
  - Mais fácil de trocar/intermutar componentes
- Problemas
  - Efeito ripple (1 pedido pode originar múltiplos pedidos/respostas)
  - Performance
  - Trabalho desnecessário
  - Granularidade das camadas
    - Grossa (não facilita a reutilização nem a e troca)
    - Fina (complexidade)

Livros

"GoF book" [Gamma95]

"POSA" [Buschmann96]

"POSA2" [Buschmann00]



"PLOPD 1,2,3,4"

Web

- Patterns Home Page
  - http://hillside.net/
- Wiki Wiki Web
  - http://c2.com/cgi-bin/wiki