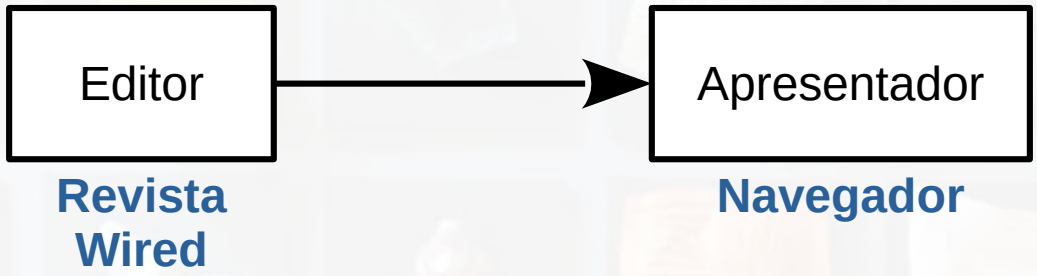
Componentização e Reúso de Software

Componentes, Mensagens e Serviços

André Santanchè
Laboratory of Information Systems – LIS
Instituto de Computação – UNICAMP
Museu Exploratório de Ciências da Unicamp
Agosto de 2019


Editor / Apresentador

Cenário 1



Editor

Wired

SHARE

SHARE

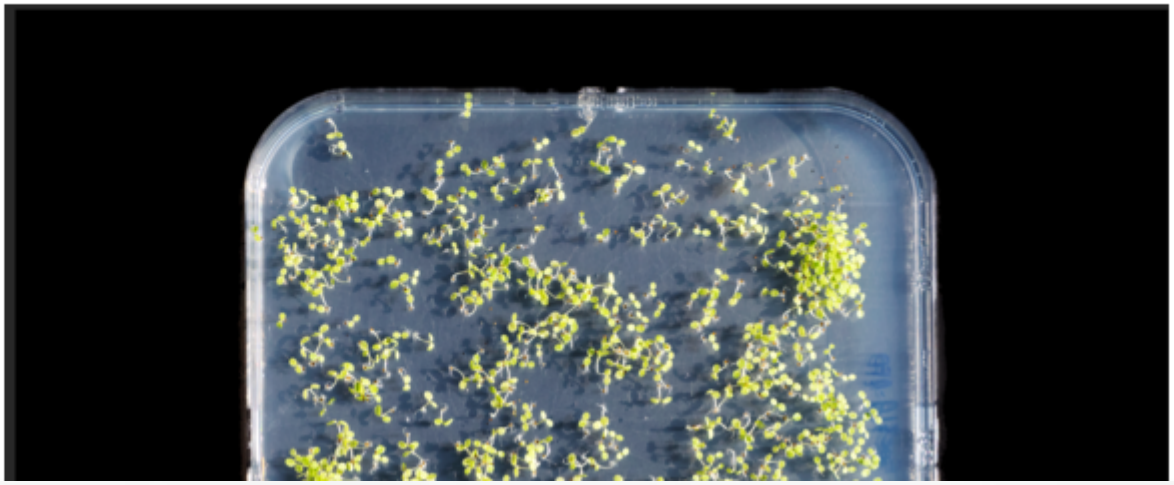
TWEET

COMMENT

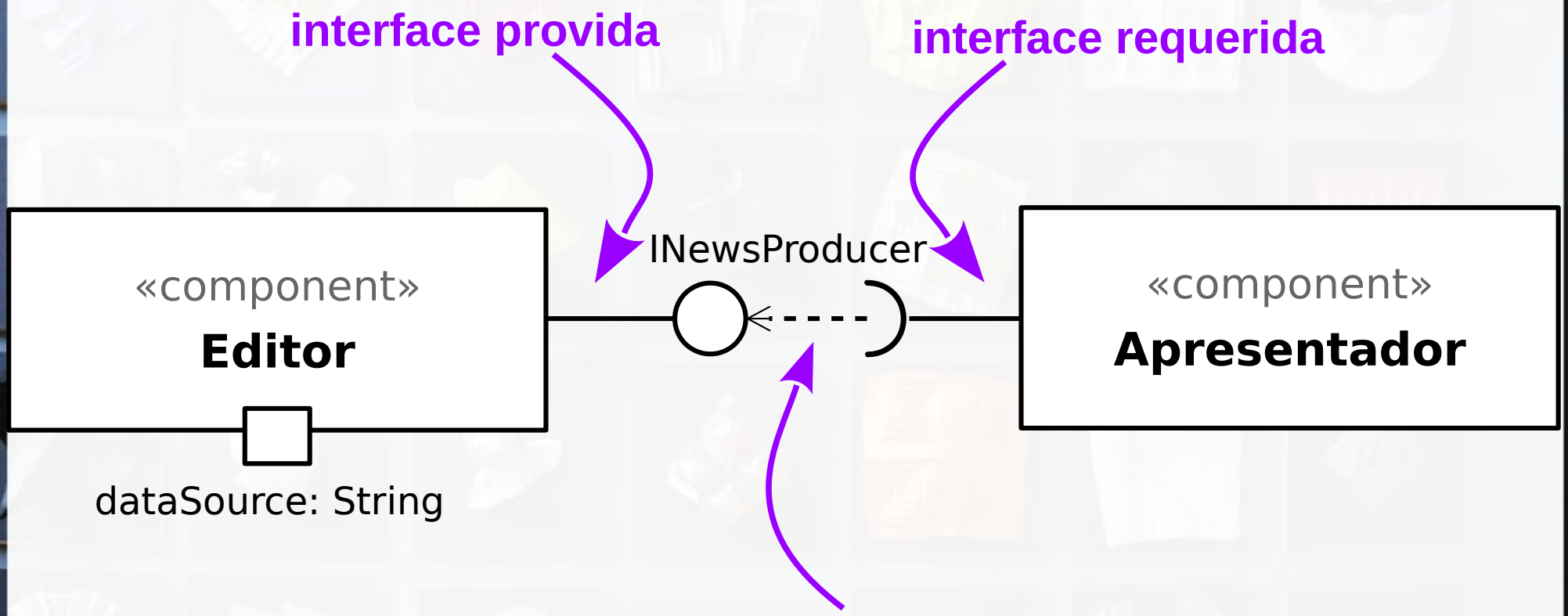
EMAIL

MEGAN MOLTENI SCIENCE 08.08.19 01:41 PM

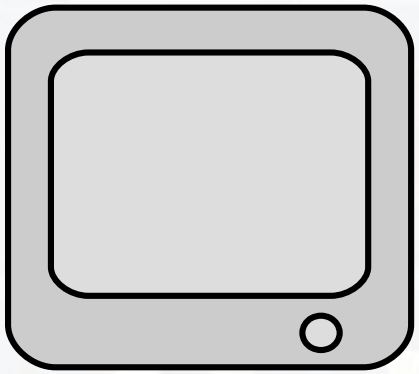
CRISPR CAN HELP SOLVE OUR LOOMING FOOD CRISIS—HERE'S HOW



Interface Provida e Requerida (blackbox)



Conectando Componentes



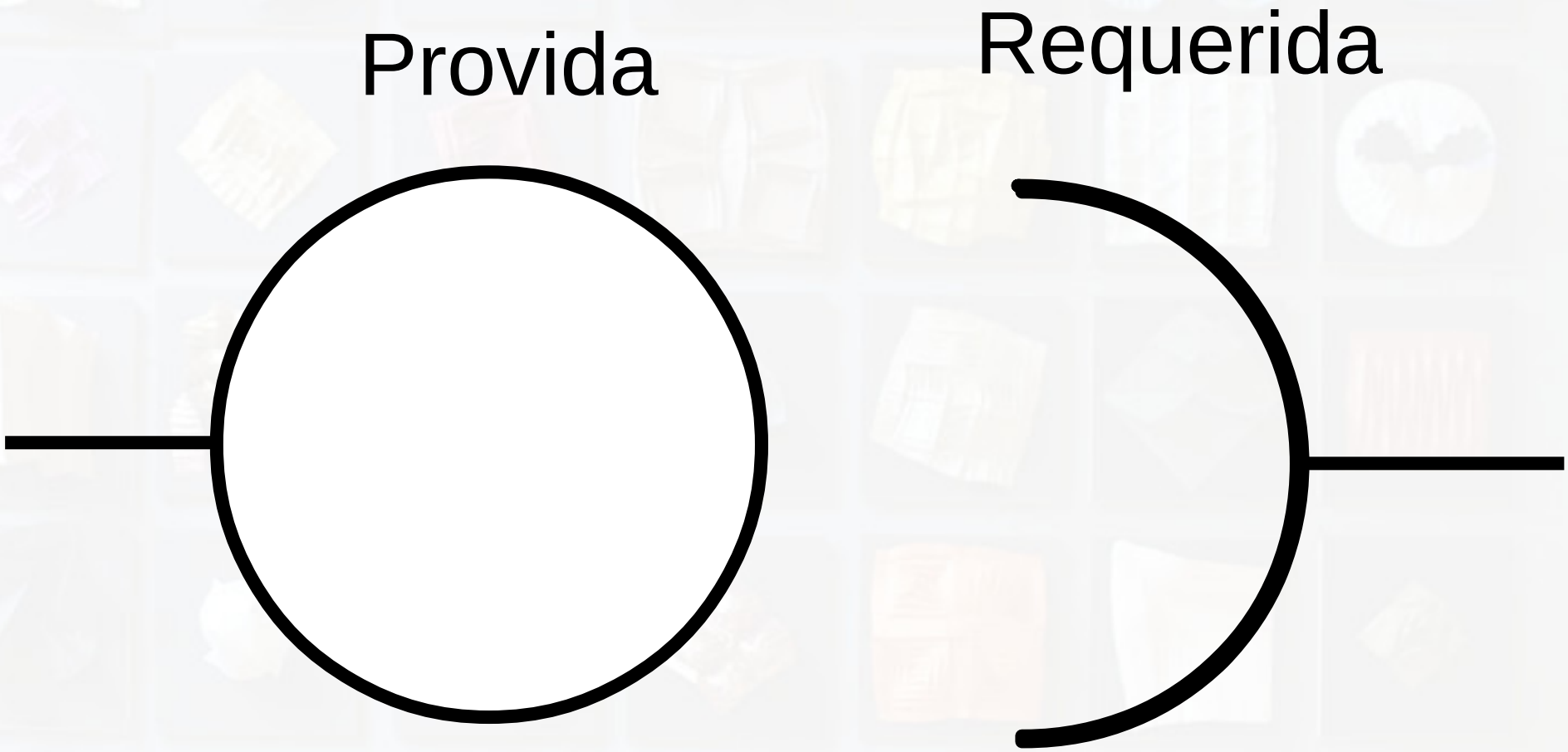
«component»
Apresentador

INewsProducer

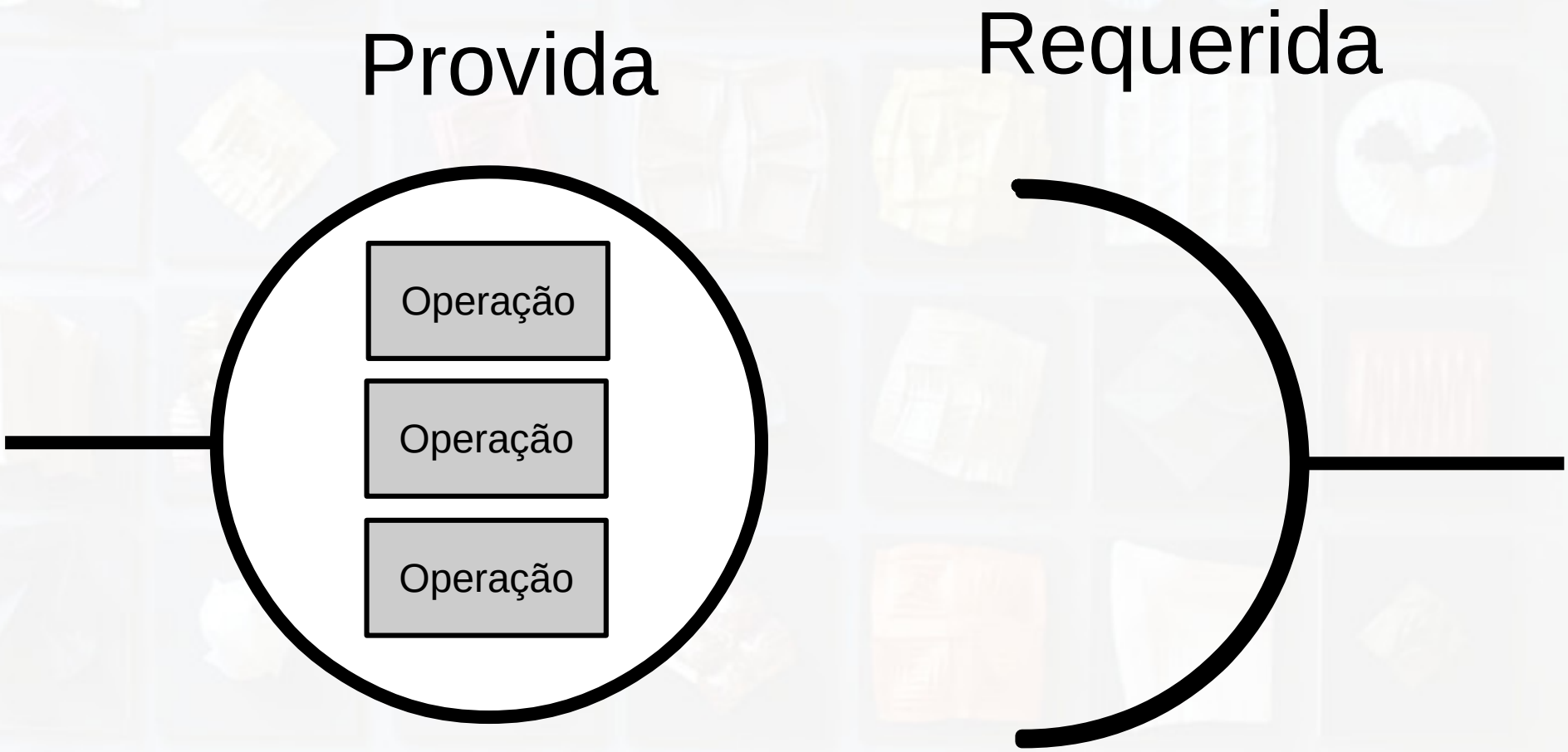
«component»
Editor

INewsProducer

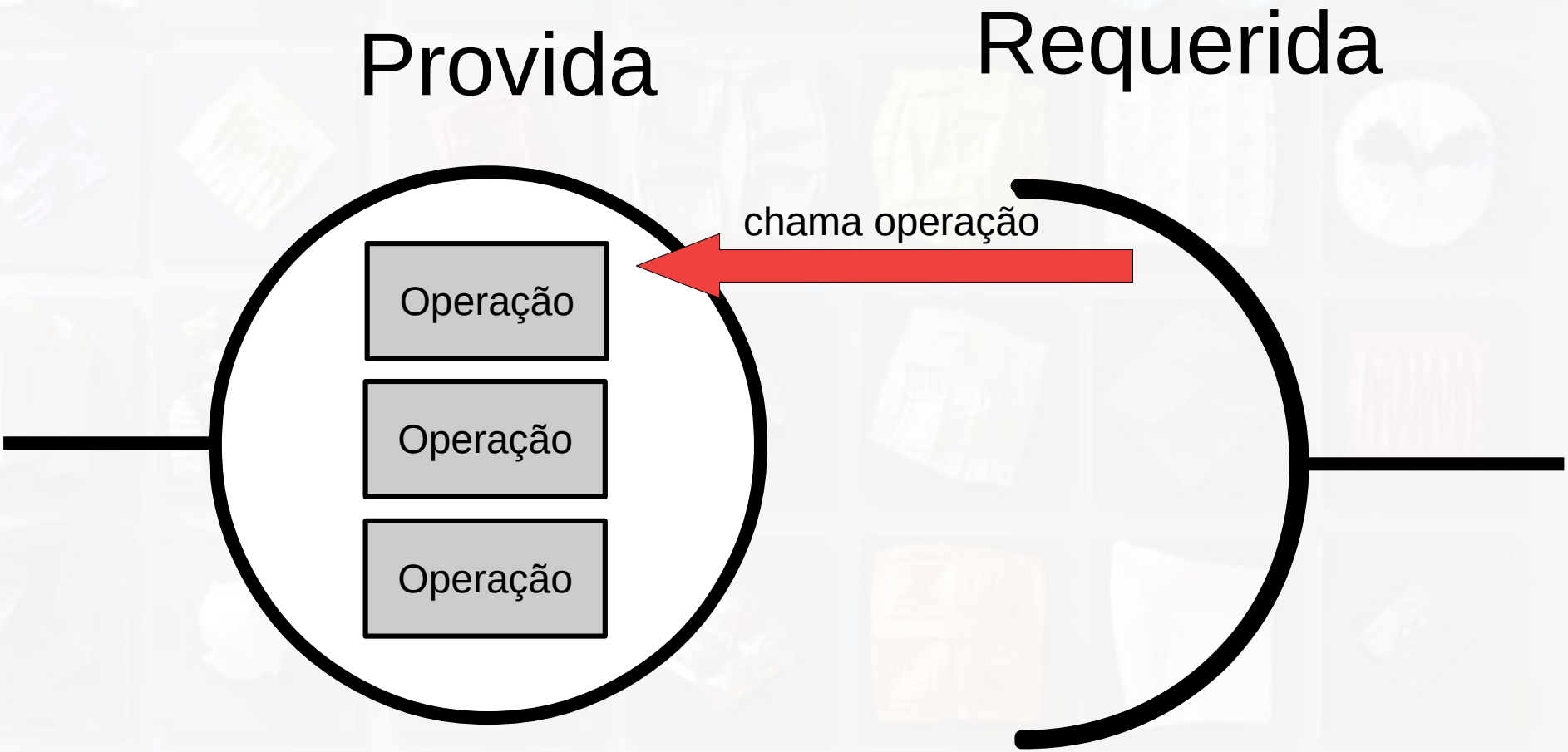
Interface Provida e Requerida



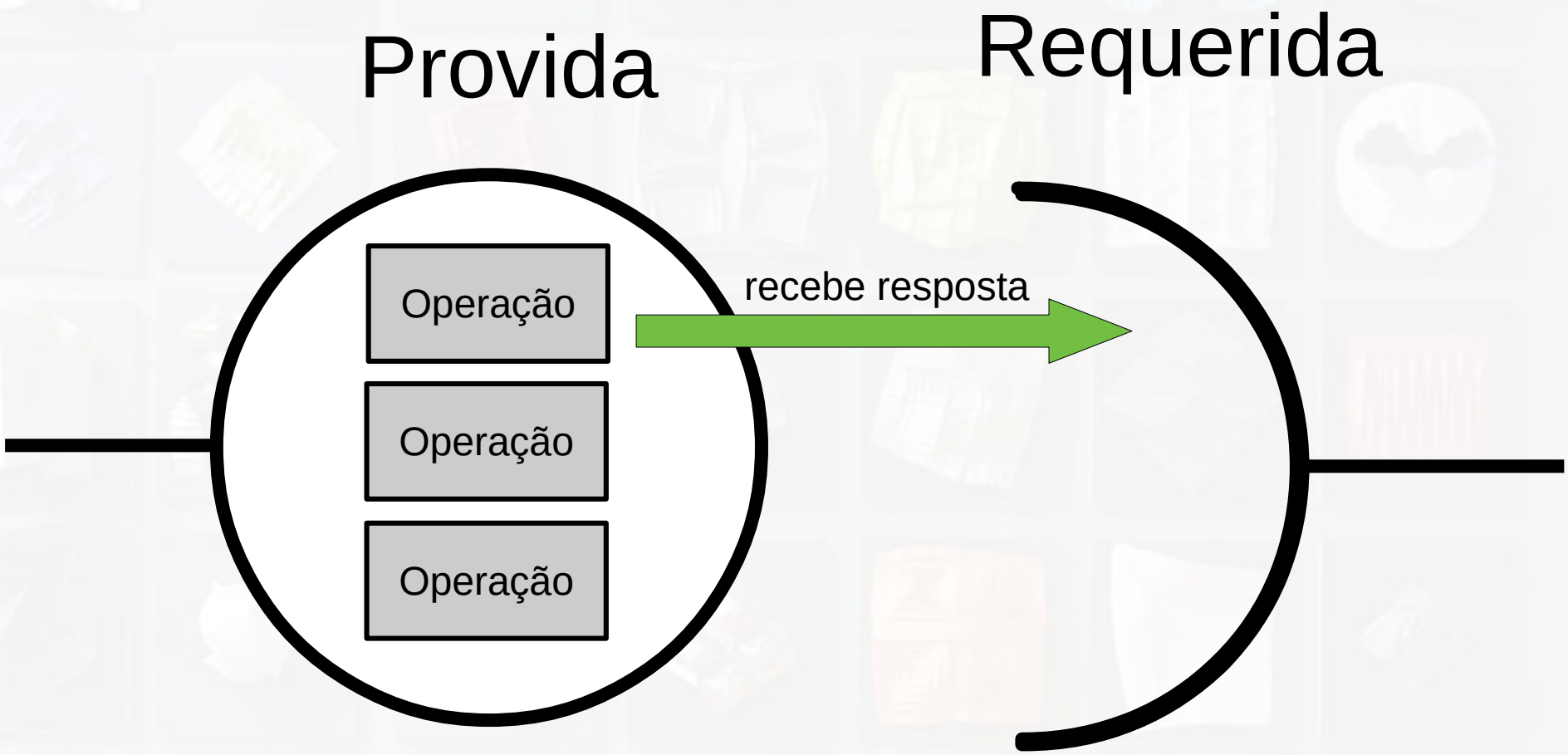
Interface Provida e Requerida



Interface Provida e Requerida

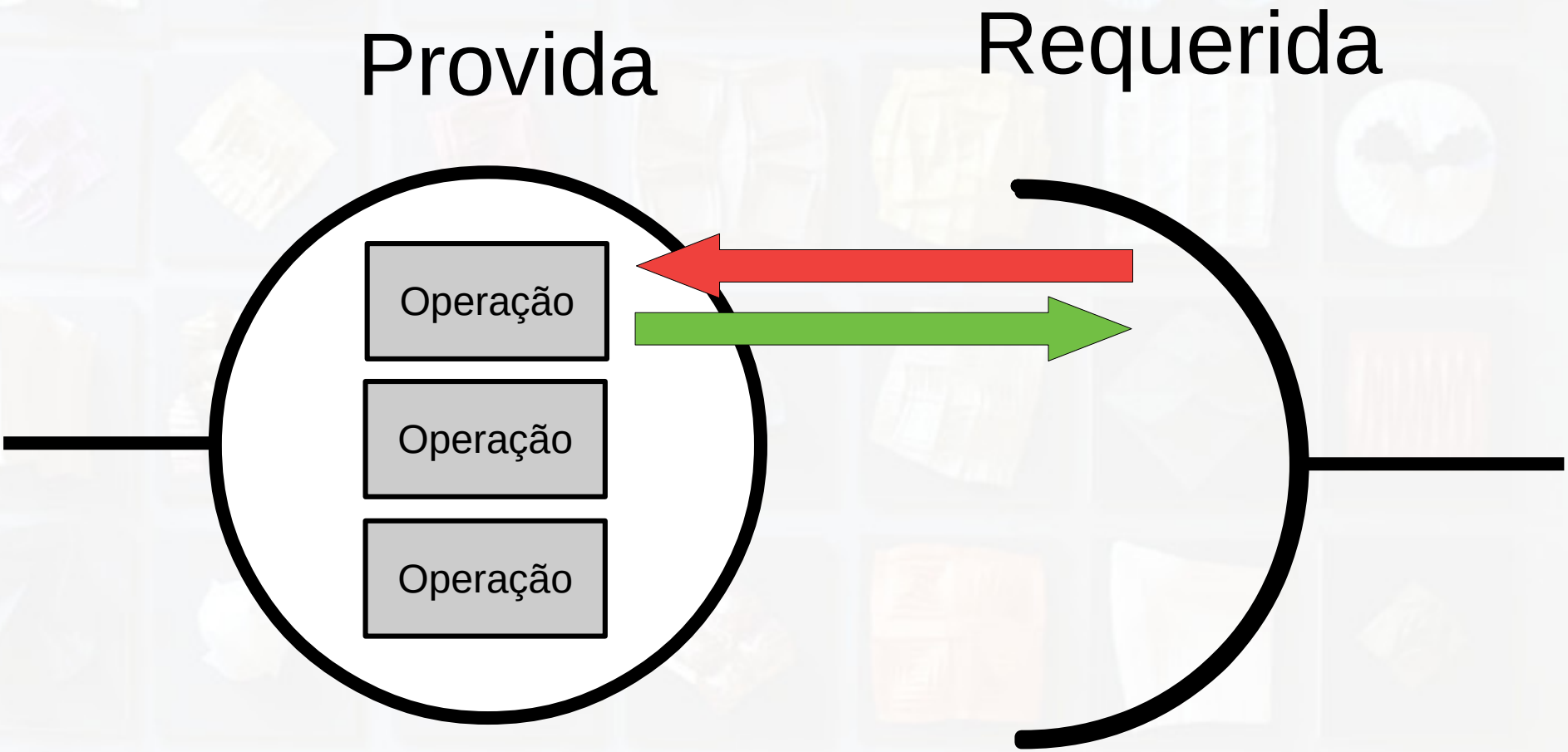


Interface Provida e Requerida

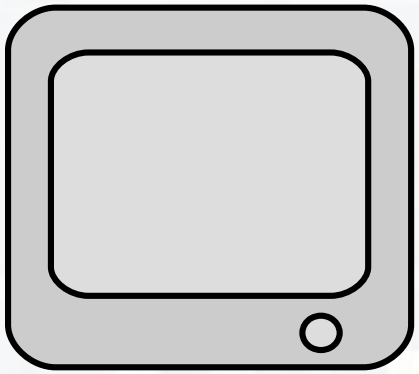


Interface Provida e Requerida

Usualmente Síncrono



Conectando Componentes



«component»
Apresentador

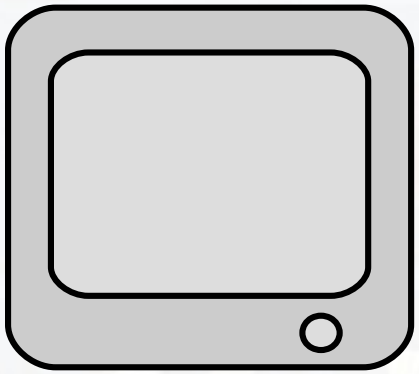
INewsProducer

requestNews()

«component»
Editor

INewsProducer

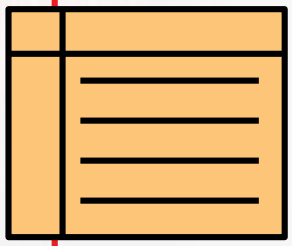
Conectando Componentes



«component»
Apresentador

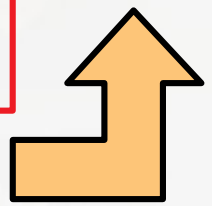
INewsProducer

`requestNews()`

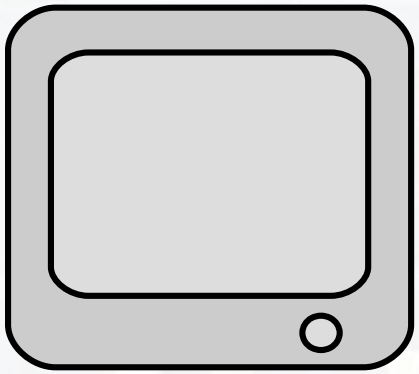


«component»
Editor

INewsProducer



Conectando Componentes



«component»
Apresentador

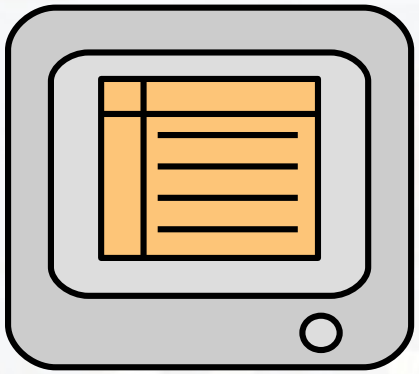
INewsProducer

`requestNews()`

«component»
Editor

INewsProducer

Conectando Componentes



«component»

Apresentador

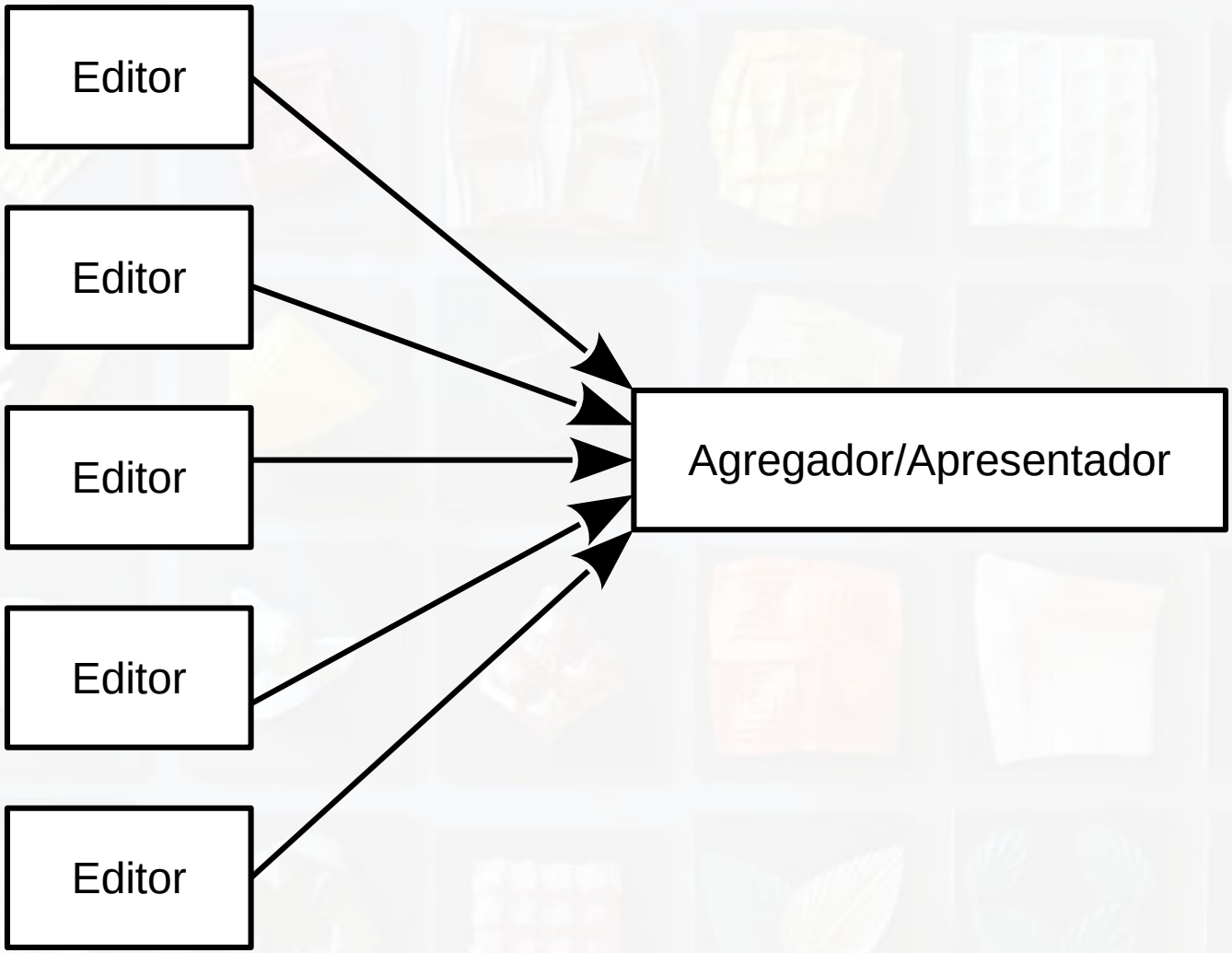
INewsProducer

«component»

Editor

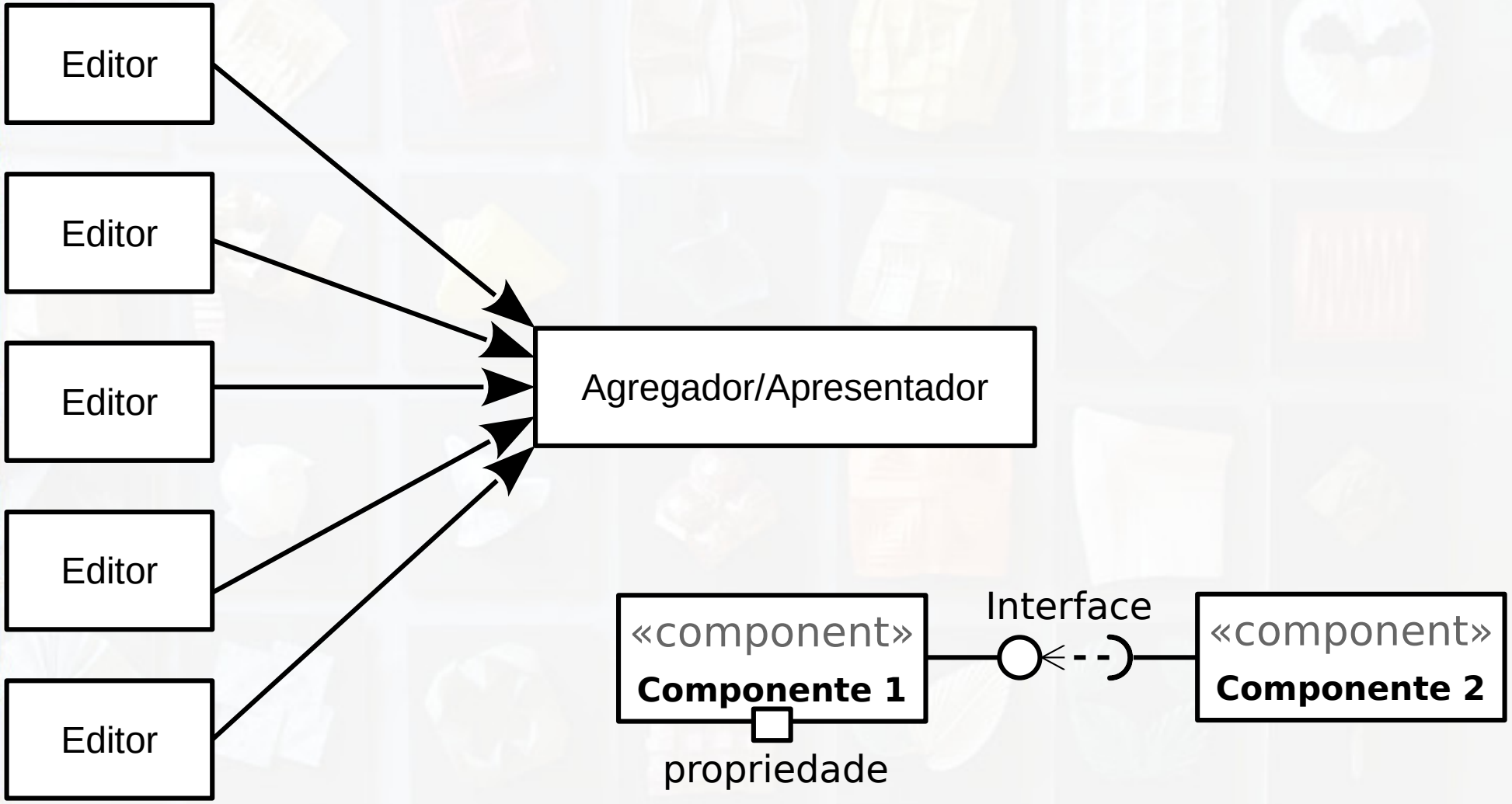
INewsProducer

Cenário 2


Editor
Editor
Editor
Editor
Editor
Agregador/Apresentador

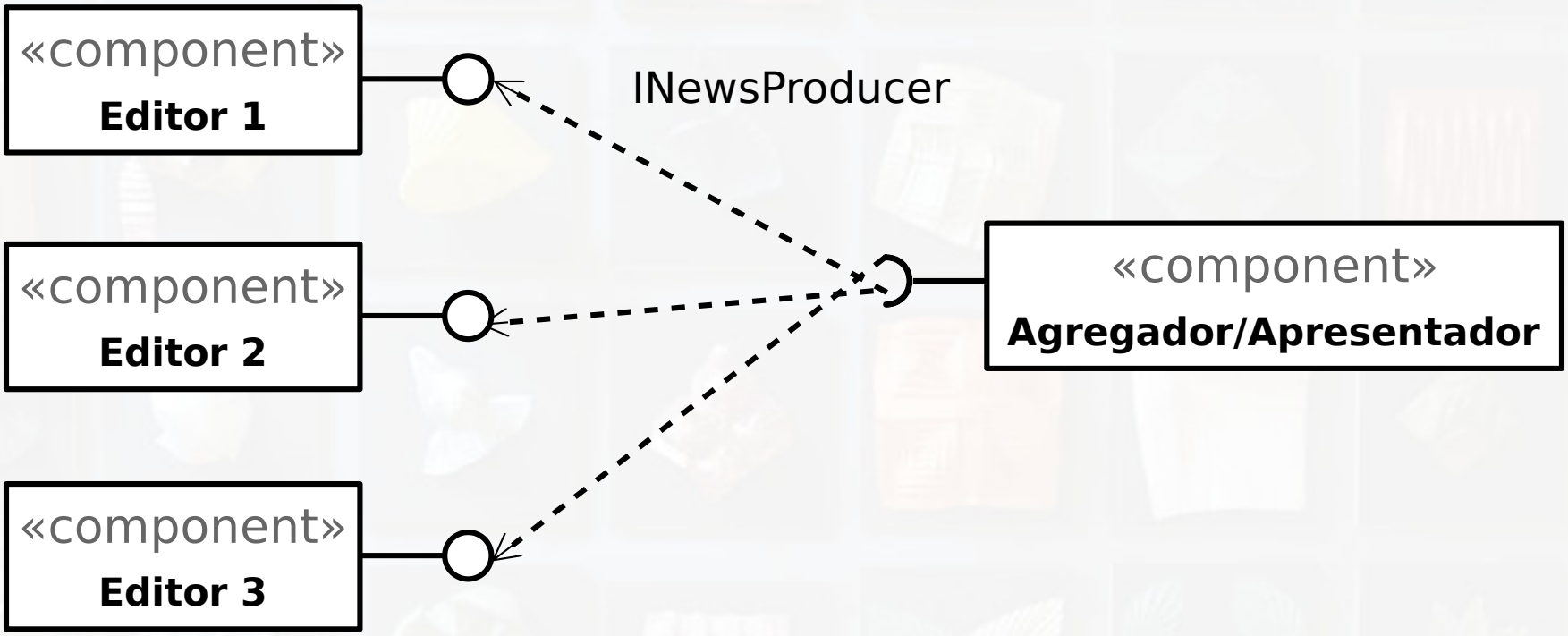
Tarefa

- Monte um diagrama de como ficaria a configuração de componentes no Cenário 2.



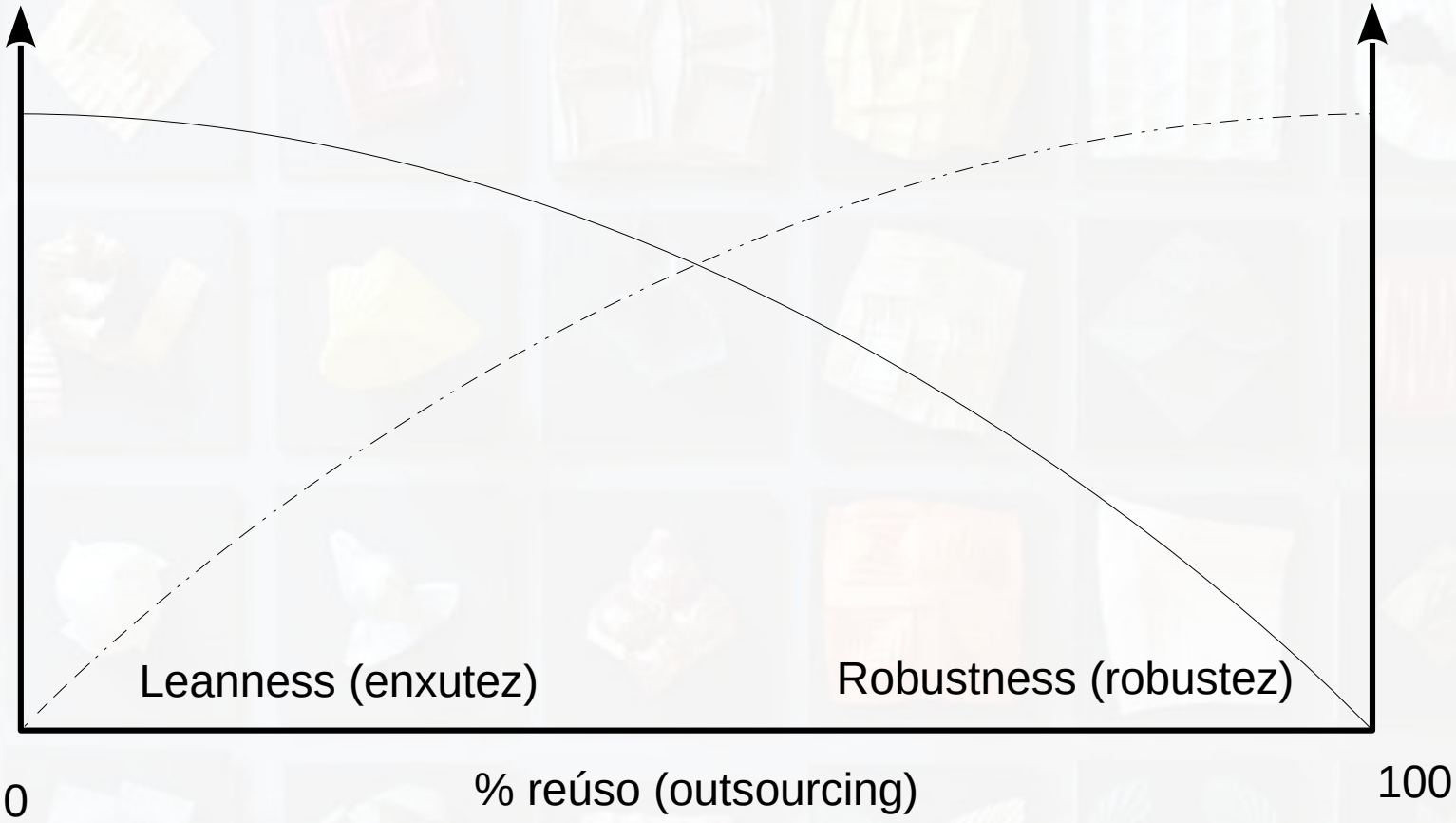
Tarefa

- Monte um diagrama de como ficaria a configuração de componentes no Cenário 2.




Enxutez x Robustez

Enxutez x Robustez



(Szyperski, 2002)

Enxutez x Robustez

- Quanto mais enxuto, maior dependência de externos
- Máxima autonomia = nenhuma dependência
  - por que usar componentes?
- Máxima exutez pode levar a muitas dependências
  - qual a diferença dos objetos?

Robustez

"... um sistema de componentes é tão robusto quanto o seu componente menos robusto." (Szyperski, 2002)

Tradução do original feita pelo autor: "... a component system is only as strong as its weakest component." (Szyperski, 2002)

Coesão & Acoplamento

Cohesion & Coupling (C&C)

Coesão

- “[...] interação interna dos componentes dentro de um módulo.”[1]
- “[...] grau em que elementos pertencem juntos dentro de um módulo.”[2]

1. “[...] internal interaction of components within the module.” (Jha et al., 2014)
2. “[...] degree to which the elements inside a module belong together.” (Stevens & Myers, 1974)

Acoplamento

- “[...] interação externa do módulo com outros módulos [...]”[1]
- “[...] grau de interdependência entre módulos de software; uma medida de quão conectadas duas rotinas ou módulos estão.”[2]

---

1. “[...] external interaction of the module with other modules [...]” (Jha et al., 2014)
2. “[...] the degree of interdependence between software modules; a measure of how closely connected two routines or modules are.” (ISO/IEC, 2010)

Acoplamento

	Complexidade da Interface	**Tipo de Conexão**	**Tipo de Comunicação**
baixo	simples, óbvia	para o módulo pelo nome	dados
ACOPLAMENTO			controle
alto	complicada, obscura	para elementos internos	híbrida

(Stevens & Myers, 1974)


Meta:
Maximizar a Coesão
Minimizar o Acoplamento

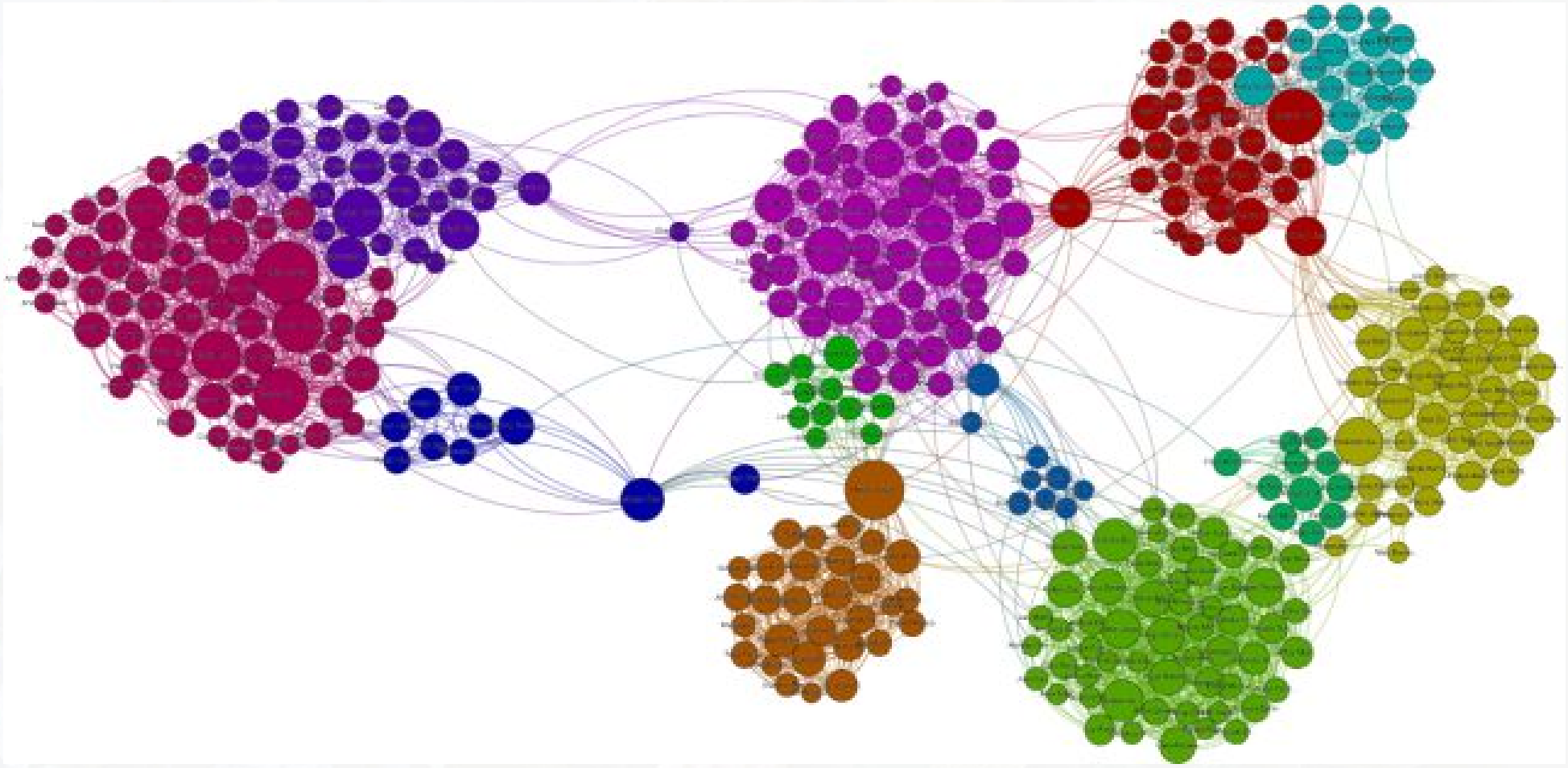
Maximizar a Coesão
Minimizar o Acoplamento

- "O acoplamento é reduzido quando o **relacionamento entre elementos que não estão no mesmo módulo são minimizados**. Há dois modos de alcançar isso – **minimizando os relacionamentos entre módulos** e **maximizando os relacionamentos entre elementos do mesmo módulo**. Na prática ambos são usados."[1]

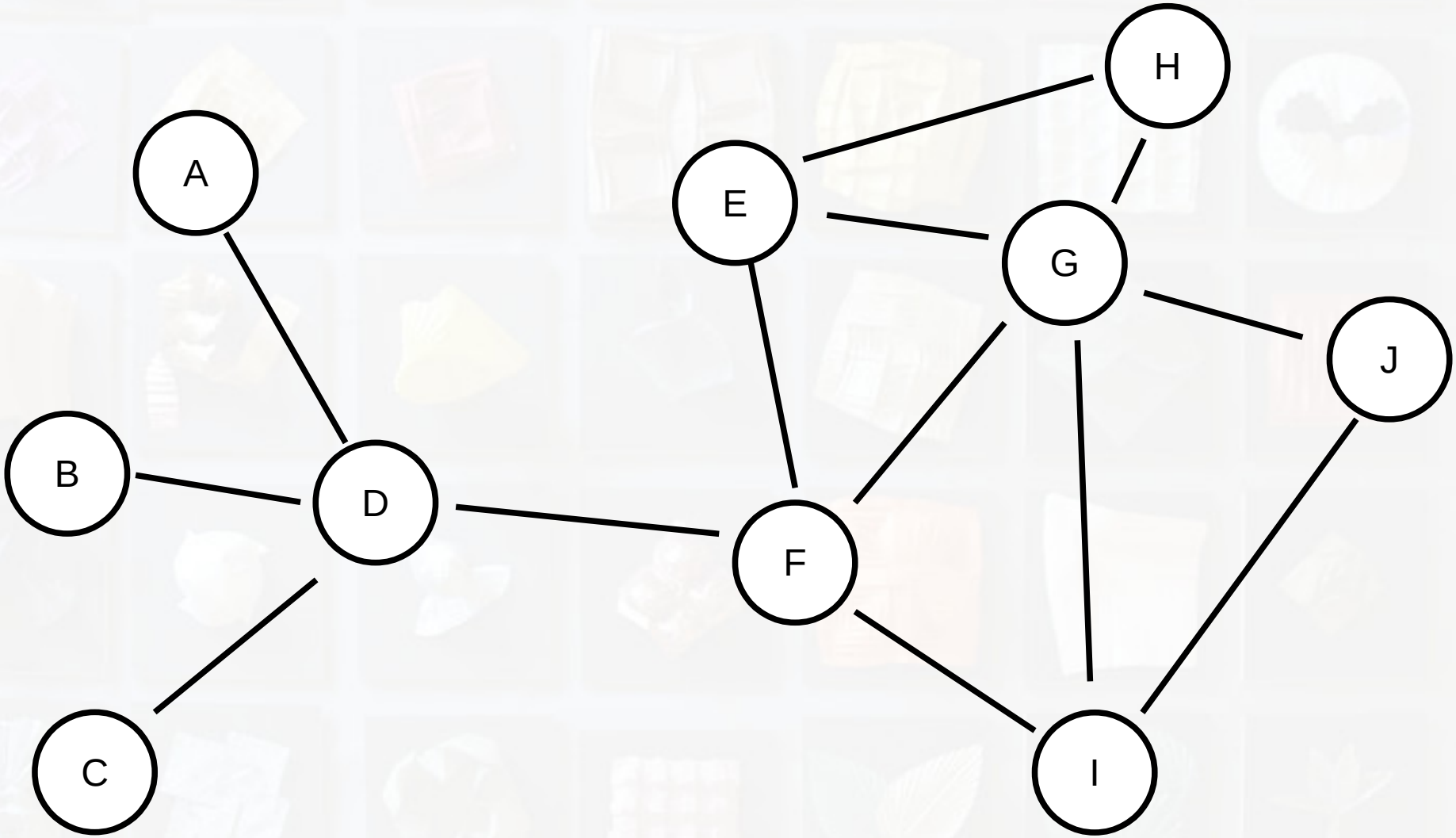
---

1. "Coupling is reduced when the relationships among elements not in the same module are minimized. There are two ways of achieving this: minimizing the relationships among modules and maximizing the relationships among elements in the same module. Both are used in practice."

Coesão e Acoplamento



Coesão e Acoplamento



Como fazer isso?

- Como desmembrar um frango?

Princípios de Projeto

- Projetos para sistemas confiáveis e extensíveis, exigem atenção para diversos aspectos relacionados à sua construção
- Princípios para a elaboração de projetos
  - criados sobre a tecnologia de orientação a objetos
  - baseados em diversos casos de sucesso
  - aumentam as chances de se produzir projetos de qualidade.

Sintomas de Problemas

- Sintomas que nos indicam que um projeto é suscetível a problemas:
  - Rigidez
  - Fragilidade
  - Imobilidade
  - Viscosidade

(Martin, 2000)

Sintomas de Problemas

Rigidez

- Dificuldade de se realizar modificações em um software
- Consequência do efeito cascata das mudanças em módulos relacionados

(Martin, 2000)

Sintomas de Problemas

Fragilidade

- Relacionado com o sintoma anterior
- Tendência ao surgimento de erros em pontos diversos de um sistema, como consequência de alguma alteração feita
- Os erros geralmente não são localizados no ponto de modificação, mas são efeitos colaterais da mudança.

(Martin, 2000)

Sintomas de Problemas
Imobilidade

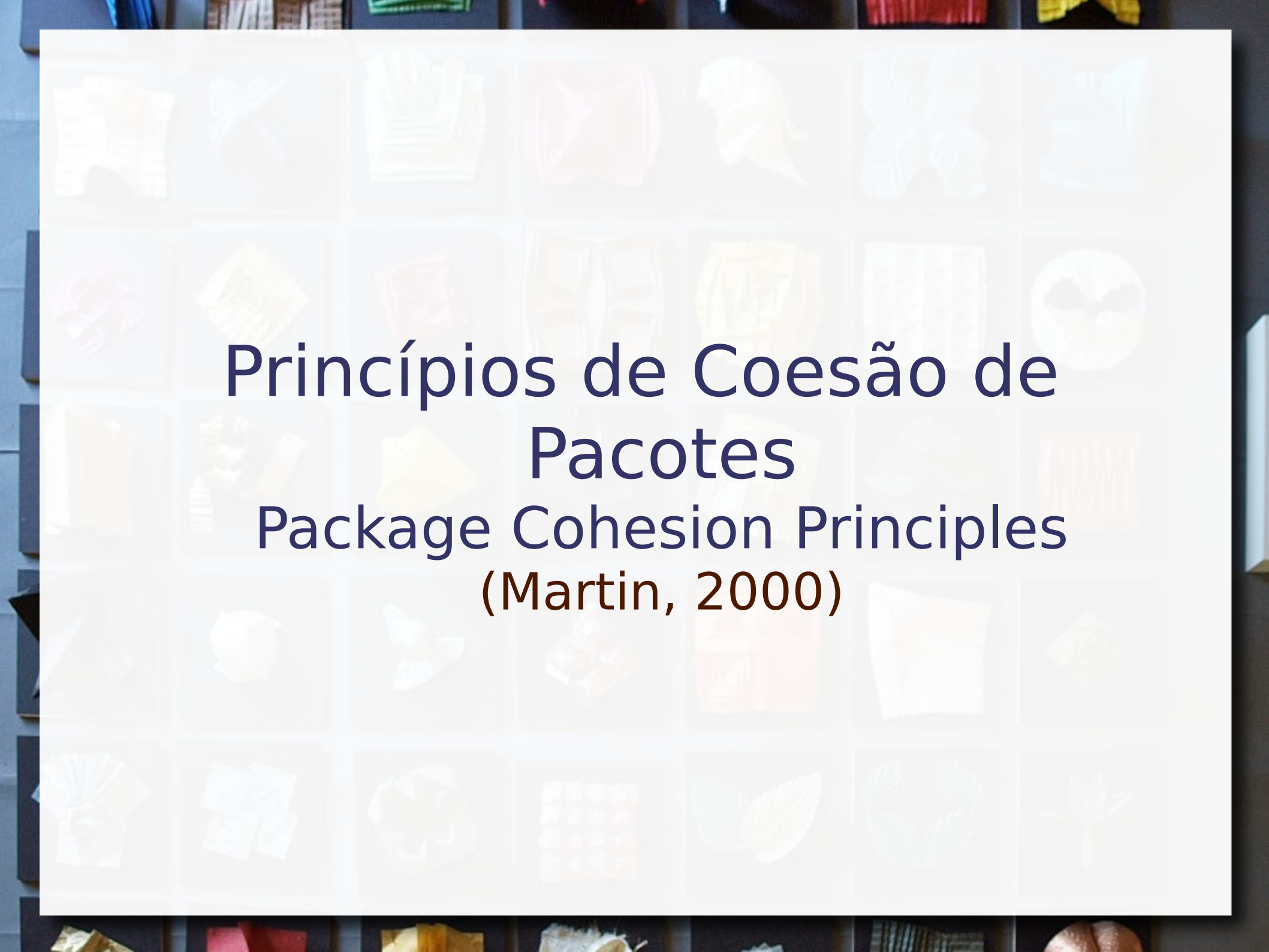
- Incapacidade de se realizar reuso de software (Martin, 2000)
- “Para que uma técnica de reuso seja efetiva, tem que ser mais fácil reusar os artefatos do que desenvolvê-los da estaca zero” (Krueger, 1992)

Sintomas de Problemas

Viscosidade

- Dificuldade em se manter a concepção original do projeto, ou requisitos do ambiente de desenvolvimento, quando se executam modificações no sistema.

(Martin, 2000)



Princípios de Coesão de Pacotes

Package Cohesion Principles

(Martin, 2000)

Release Reuse Equivalency Principle (REP)

- “The granule of reuse is the granule of release.” (Martin, 2000)
- Componentes precisam de um sistema de versionamento (release)
- Novas versões não podem afetar (quebrar) antigas versões

Common Closure Principle (CCP)

- “Classes that change together, belong together.” (Martin, 2000)
- Classes como elementos de implementação de componentes
- Classes com alta interdependência
  - Modificação de uma classe impacta muito os demais
  - Exigem testes e validação em conjunto quando modificadas
  - Devem estar no mesmo componente para alta coesão e baixo acoplamento

Common Reuse Principle (CRP)

- “Classes that aren’t reused together should not be grouped together.” (Martin, 2000)
- Dificuldade de reusar um componente sem carregar os demais
- A modificação de um componente só devia afetar quem o usa
  - não quem usa os que o acompanha

Tensão entre os Princípios de Coesão de Pacotes

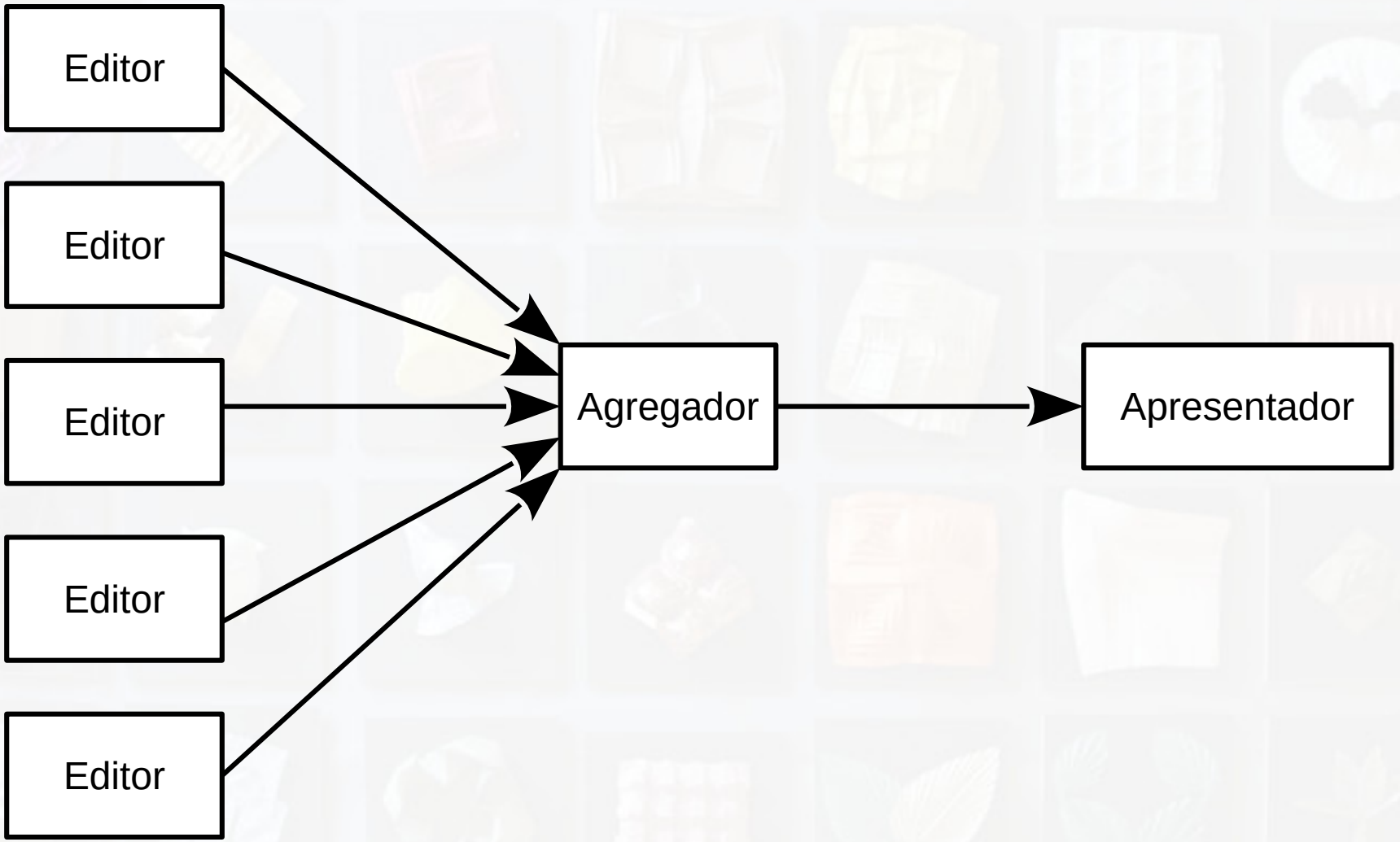
- Princípios:
  - Release Reuse Equivalency Principle (REP)
  - Common Reuse Principle (CRP)
  - Common Closure Principle (CCP)
- São mutuamente exclusivos
- REP e CRP → para reusuários
  - CRP → pacotes o menor possível
- CCP→ para quem mantém
  - pacotes o maior possível
- Buscar o equilíbrio

Tarefa

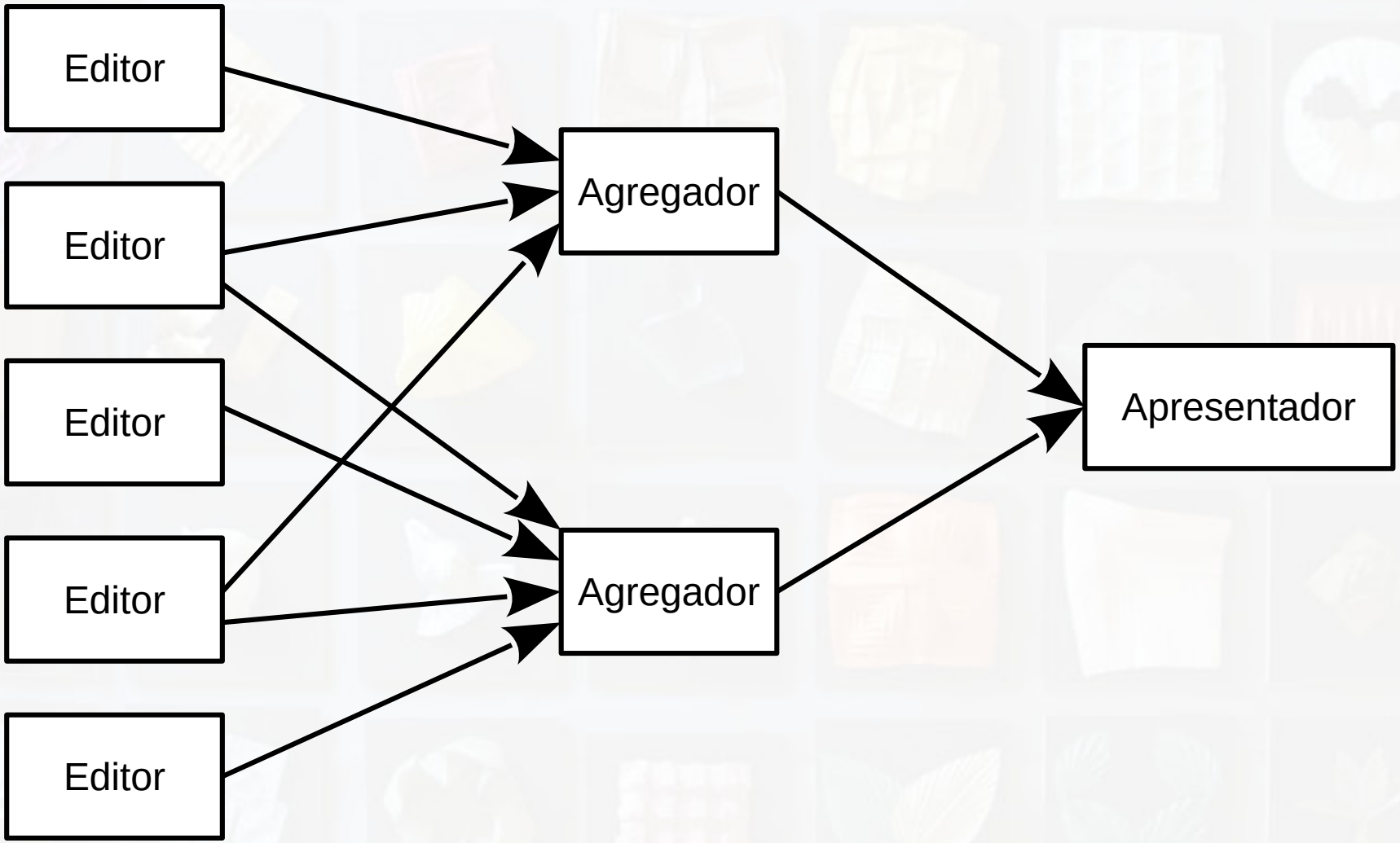
- Faça uma revisão dos componentes da tarefa anterior quanto à coesão e acoplamento, tomando como base os Princípios de Coesão de Pacotes.
- Se for possível, proponha uma melhoria nos mesmos.

Agregador de Notícias

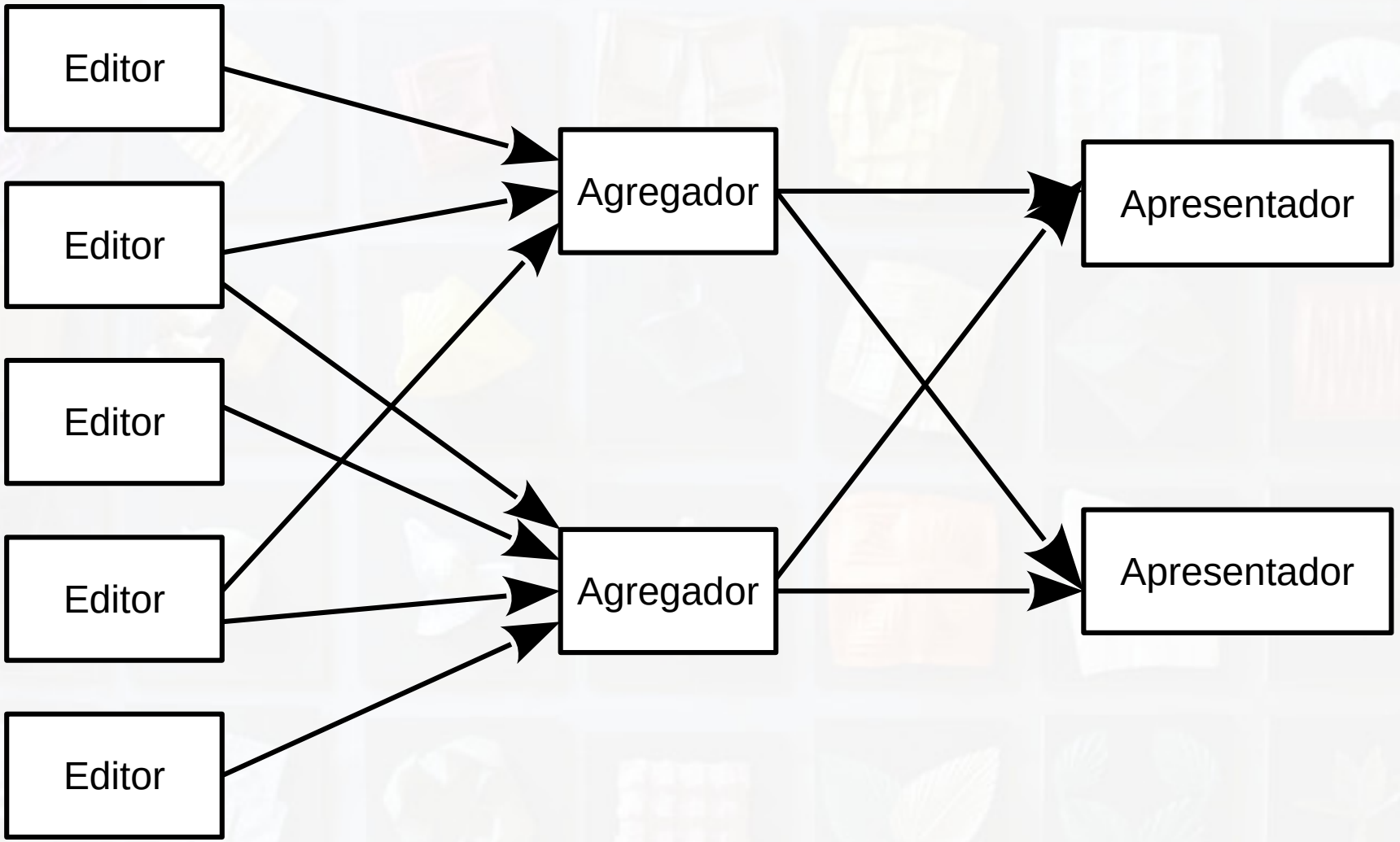
Cenário 3a


Editor
Editor
Editor
Editor
Editor
Agregador
Apresentador

Cenário 3b


Editor
Editor
Editor
Editor
Editor
Agregador
Agregador
Apresentador

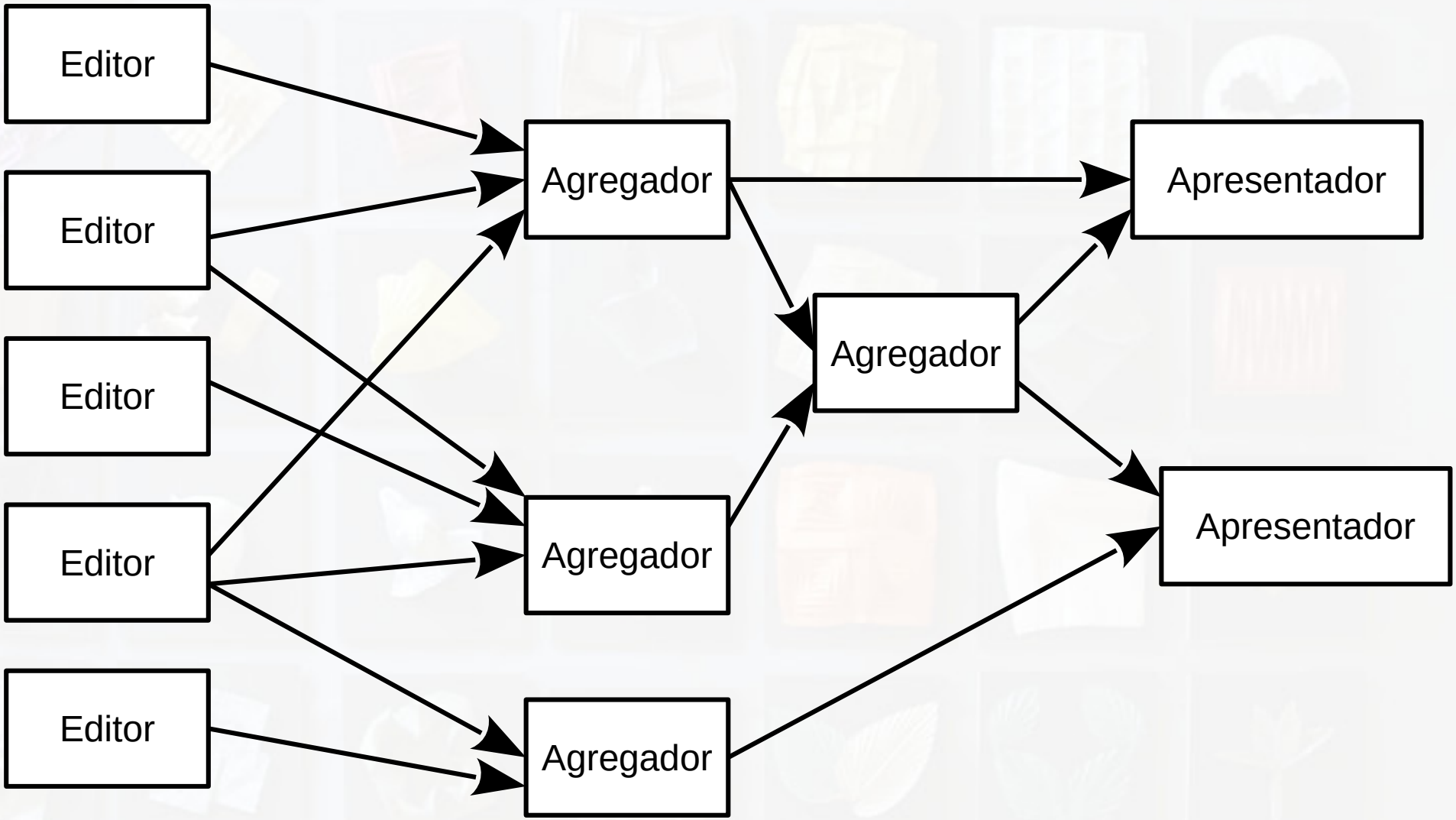
Cenário 3c


Editor
Editor
Editor
Editor
Editor
Agregador
Agregador
Apresentador
Apresentador

Agregadores

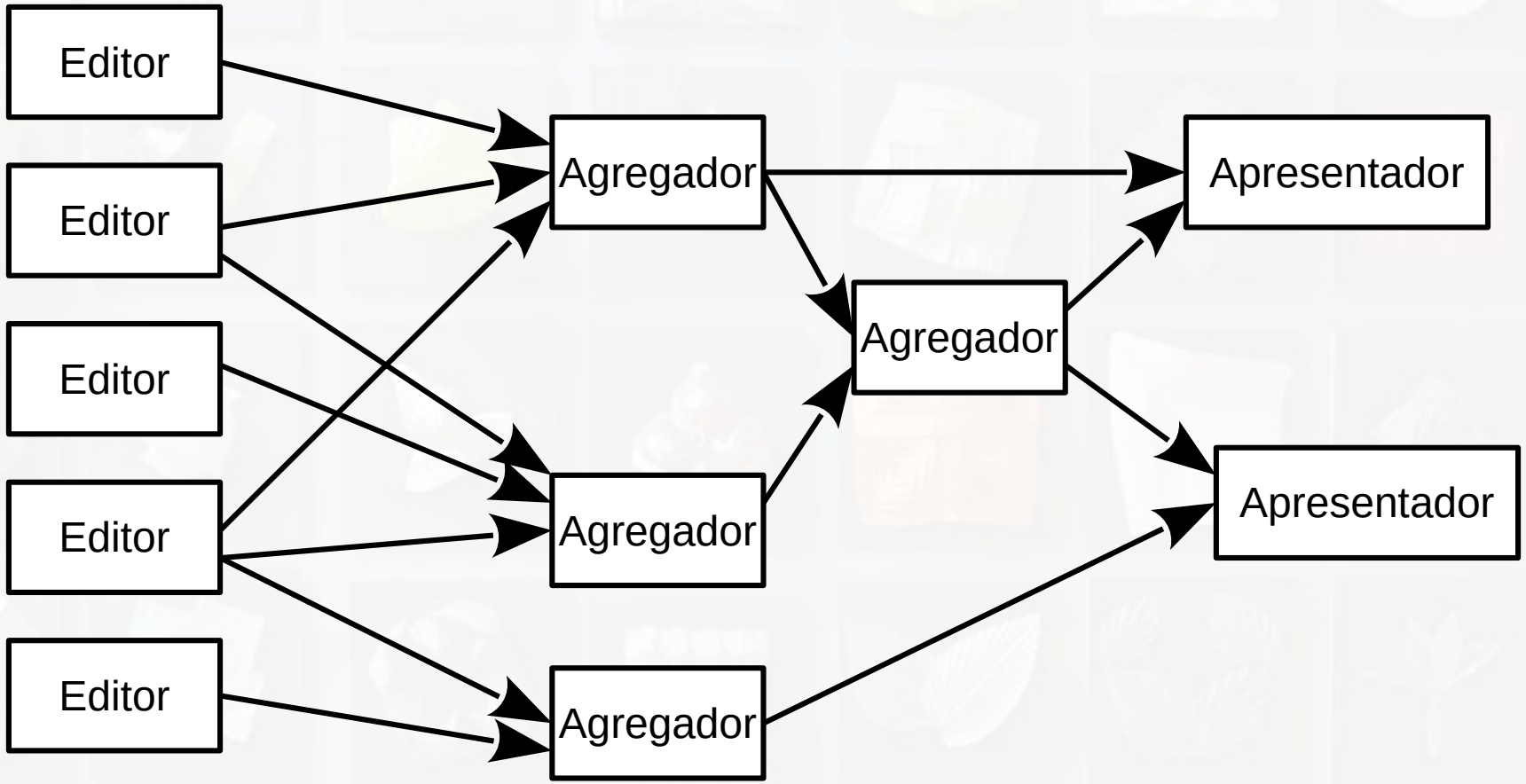
- Agregadores podem ser concatenados em diversas camadas

Cenário 4a


Editor
Editor
Editor
Editor
Editor
Agregador
Agregador
Agregador
Agregador
Apresentador
Apresentador

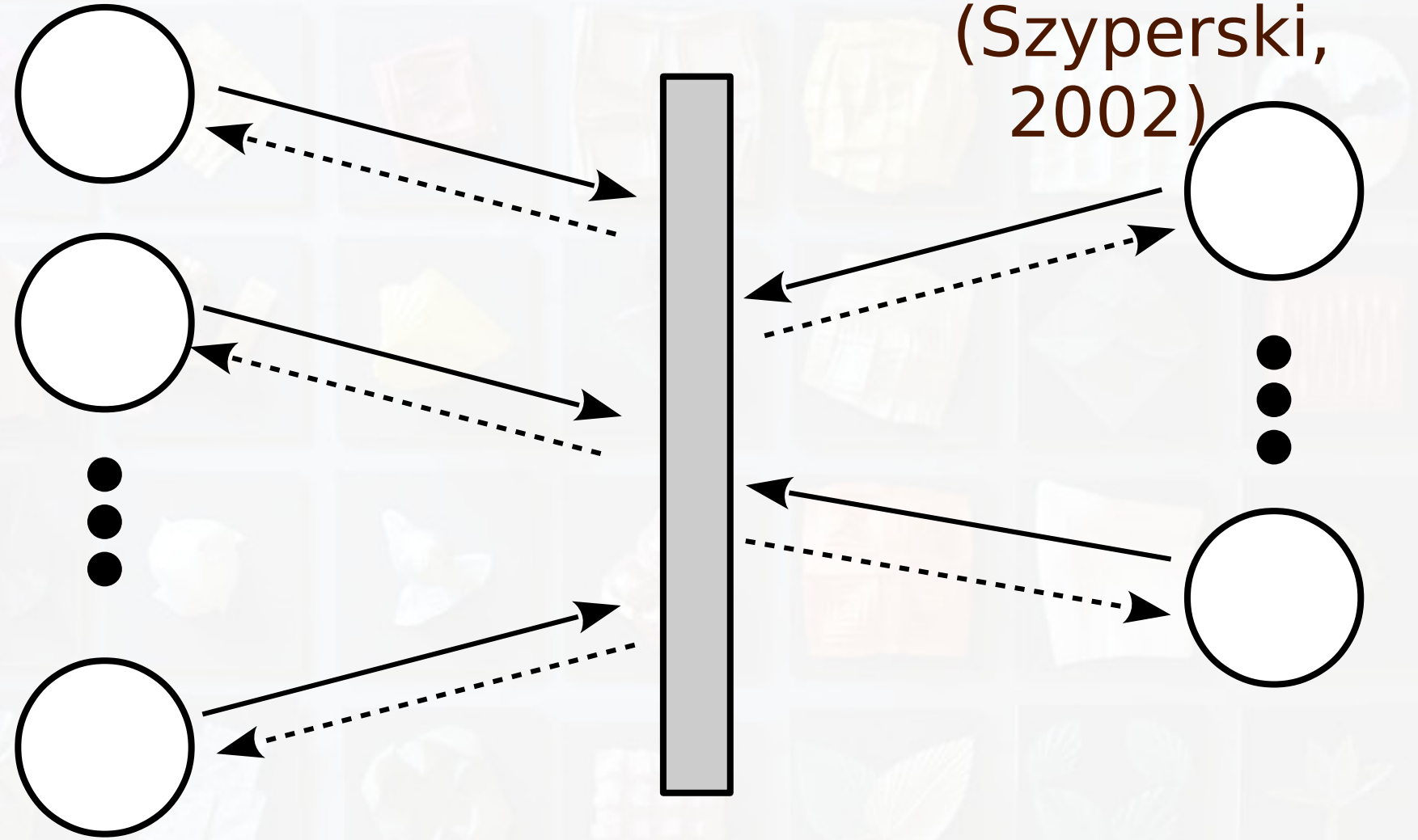
Tarefa

- Adapte o modelo da última tarefa para que atenda múltiplas camadas.

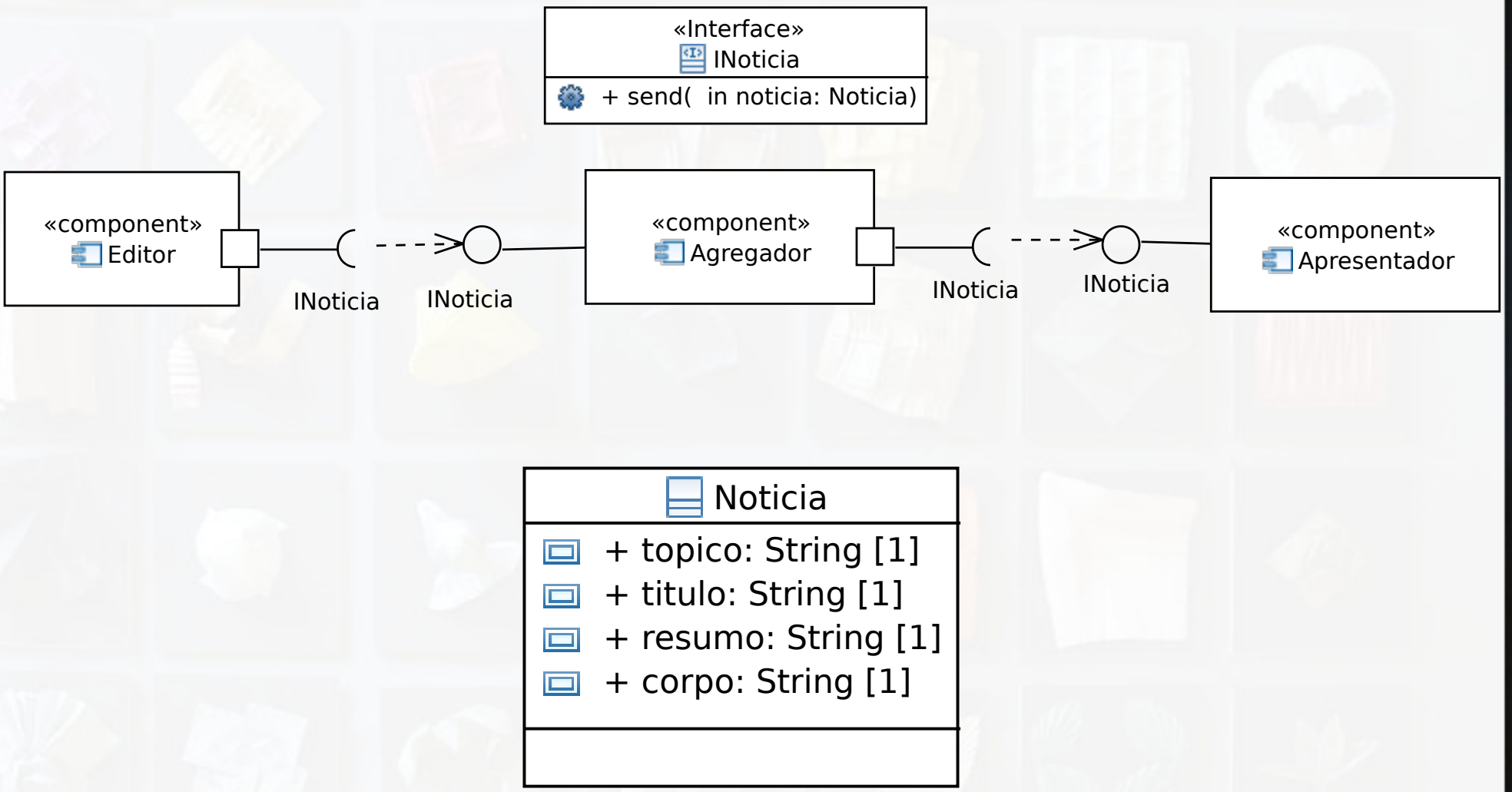


Contrato

Interface Pivô



Tarefa 4


«Interface»
INoticia
+ send(in noticia: Noticia)
«component»
Editor
INoticia
INoticia
«component»
Agregador
INoticia
INoticia
«component»
Apresentador
Noticia
+ topico: String [1]
+ titulo: String [1]
+ resumo: String [1]
+ corpo: String [1]

Dependency Inversion Principle (DIP)

- “Depender das Abstrações. Não depender das Concretizações.” (Martin, 2000)

Liskov Substitution Principle (LSP)

- Associado à noção de Tipo Abstrato de Dados – Abstract Data Type (ADT)
- Foi enunciado por Barbara Liskov
- Baseado na noção de subtipo:
  - Dado que um programa P que faz uso de um objeto O1; O2 será subtipo de O1 se for possível substituir O1 por O2 no programa P, sem que P altere seu comportamento (Liskov, 1987).
- Em OO: noção de subclasse equivale a noção de subtipo

Arquitetura de Software

Arquitetura de Software

- Trata de grandes estruturas do sistema
- Abstração – desconsidera detalhes de implementação, algoritmos e estruturas de dados
- Se concentra na interação de elementos do sistema como “caixas pretas”

(Bass, 2003)

Arquitetura de Software
Definição

- “A organização fundamental de um sistema personificado pelos seus componentes, seus relacionamentos entre si, e com o ambiente, e os princípios que guiam seu projeto e evolução.” (IEEE, 2007)

Arquitetura de Software
Definição

- “A organização fundamental de um sistema personificado pelos seus componentes, seus relacionamentos entre si, e com o ambiente, e os princípios que guiam seu projeto e evolução.” (IEEE, 2007)

Estilo Arquitetural

Padrão ou Estilo Arquitetural

- Famílias de programas
  - conjuntos de programas que possuem tantas propriedades em comum, que torna-se mais vantajoso estudá-las a partir de suas similaridades, antes mesmo de analisar membros individuais (Parnas, 1976)
- Similaridades apontam para “padrões arquiteturais” ou “estilos arquiteturais”

Padrão ou Estilo Arquitetural

- “Um padrão arquitetural é uma descrição de tipos de elementos e relações junto com um conjunto de restrições relativas a como eles podem ser usados.” (Bass, 2003)

Arquiteturas e Estilos Arquiteturais
Componentes e Conectores

- Necessidade de documentação para que sejam compartilhados, analisados e reusados
- Esforços de formalização representam arquiteturas como:
  - Componentes
  - Conectores

Eventos, Arquitetura baseada em Mensagens e Barramento

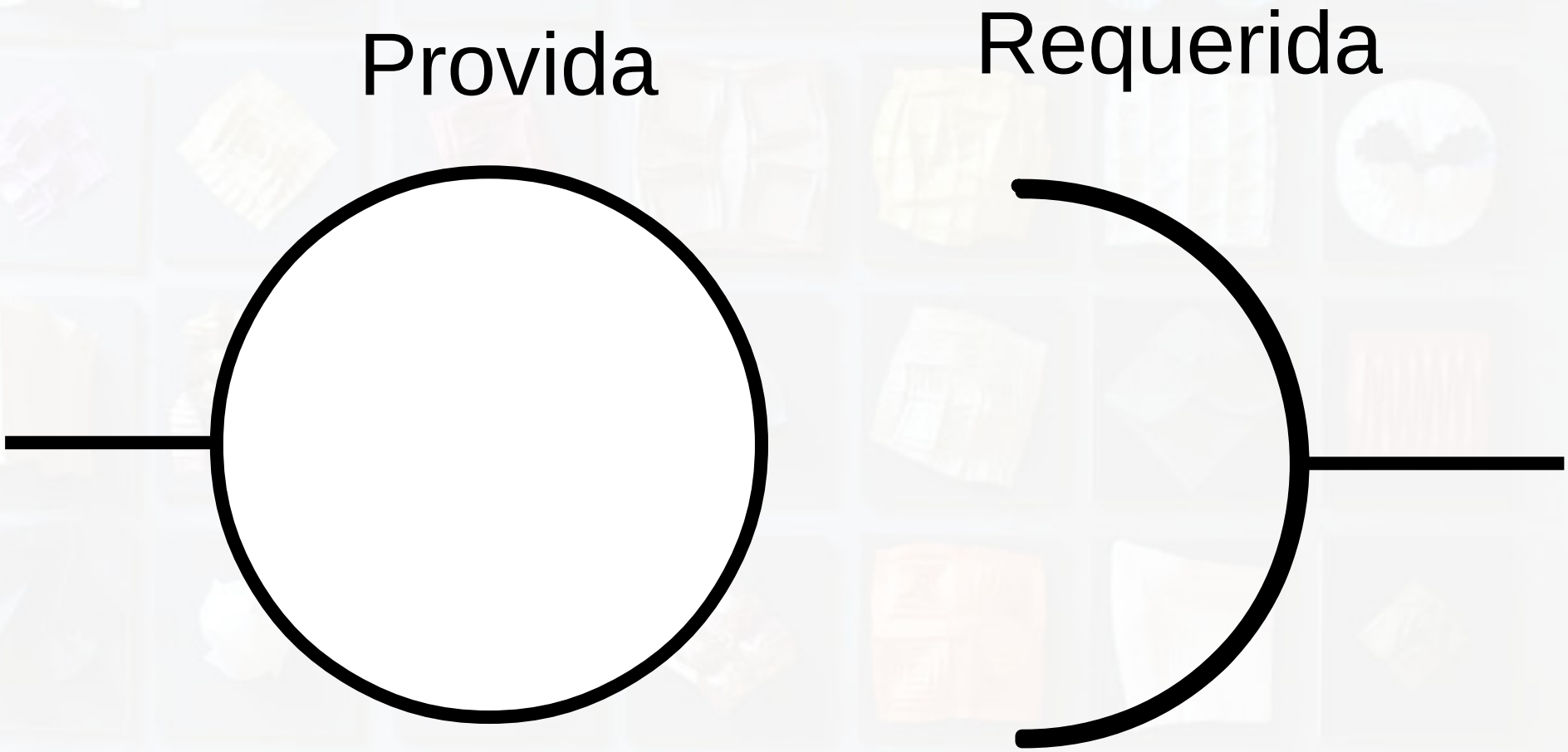
Eventos

- Componentes podem interagir através da difusão (broadcast) de eventos
- Ação inicia com um componente que 'anuncia' um evento
- Evento anunciado pode disparar operações em outros componentes

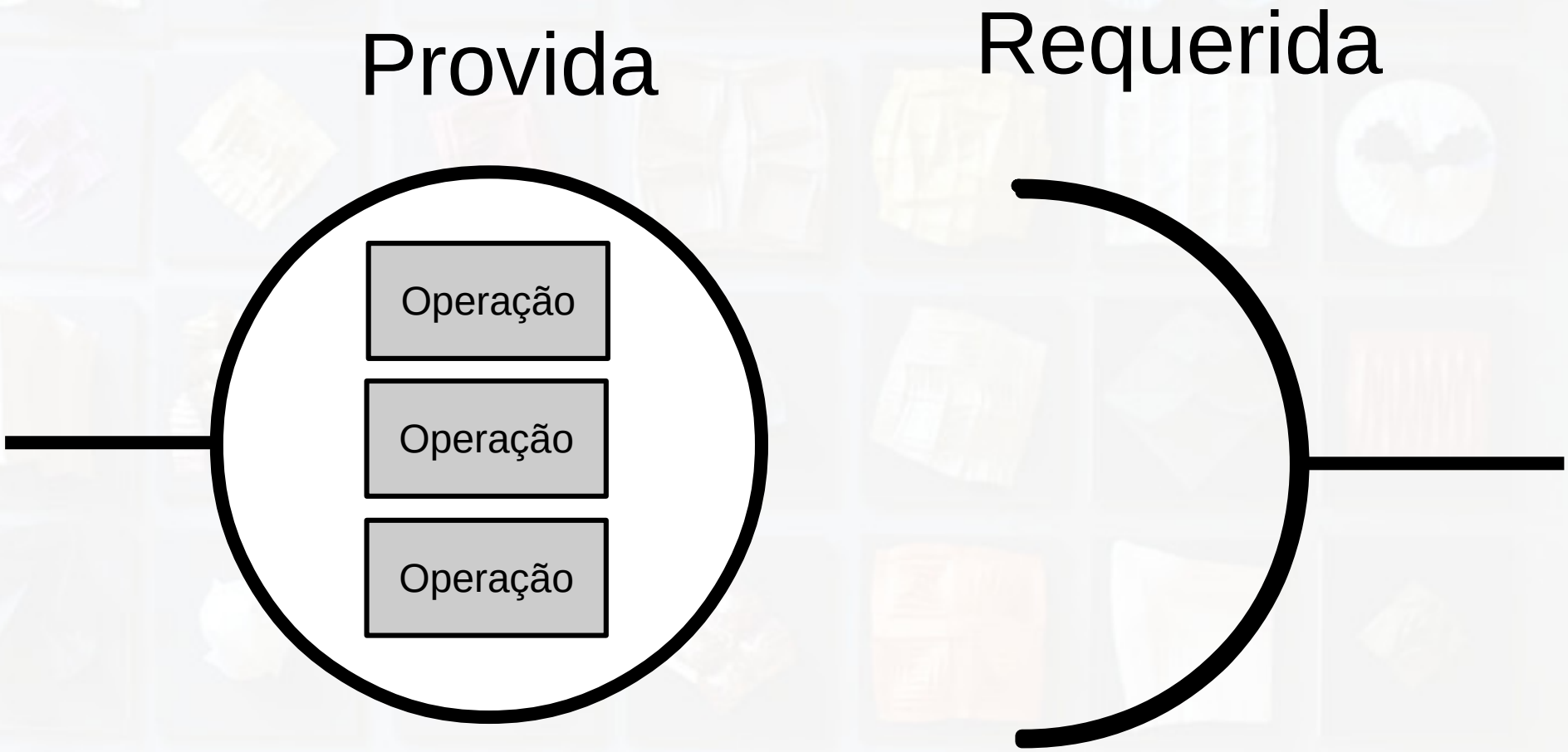
(Abowd, 1995)

- Exemplo: Publish-Subscribe

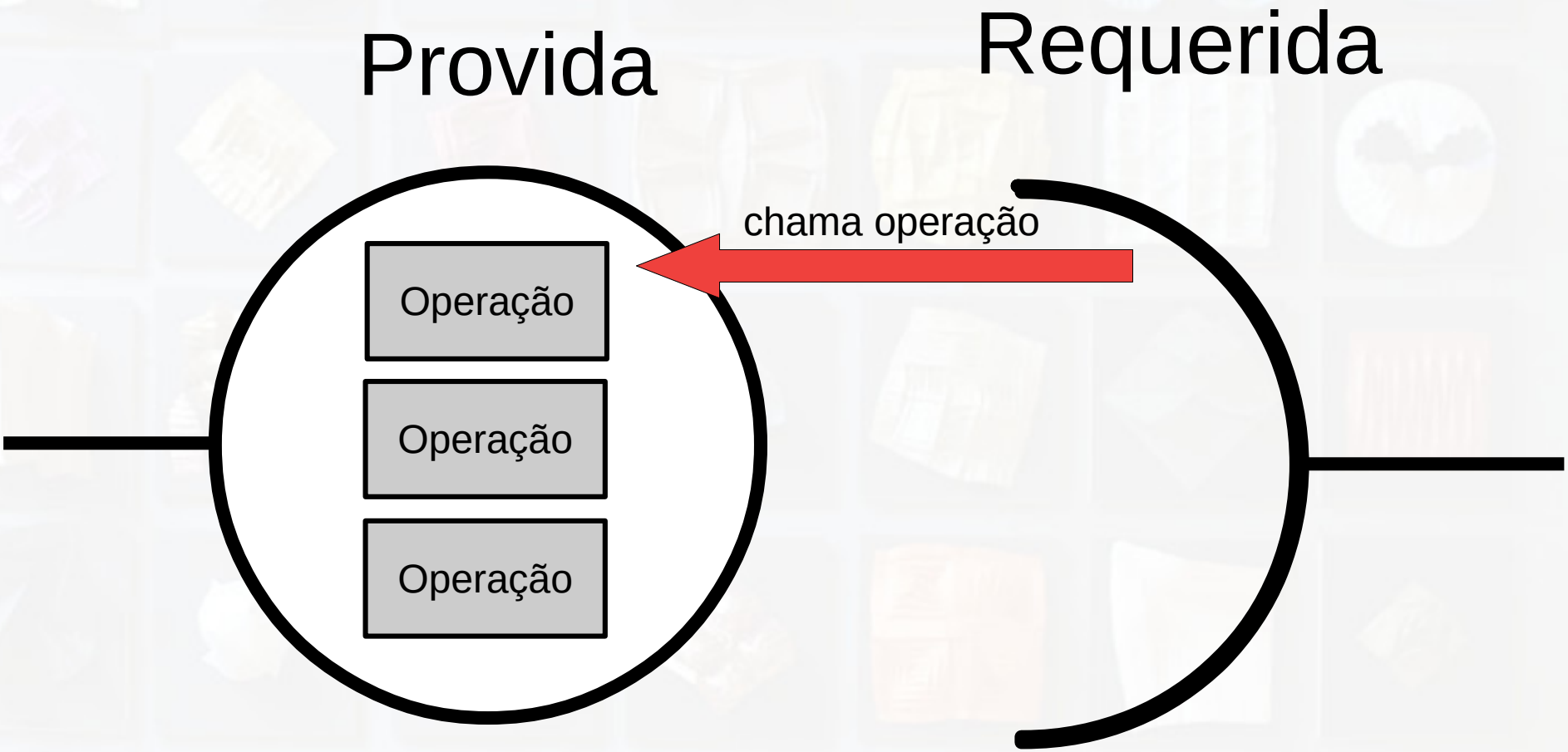
Interface Provida e Requerida



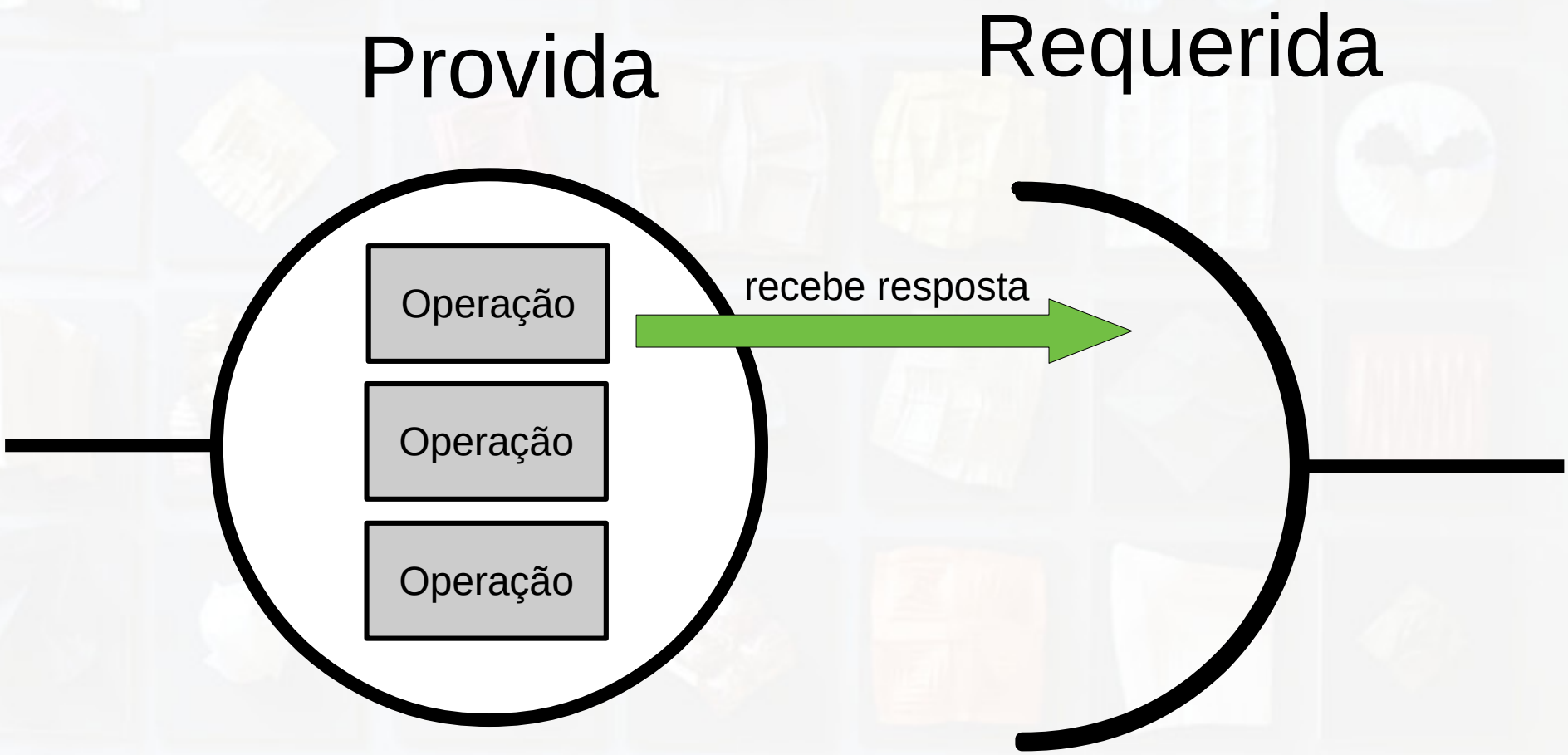
Interface Provida e Requerida



Interface Provida e Requerida

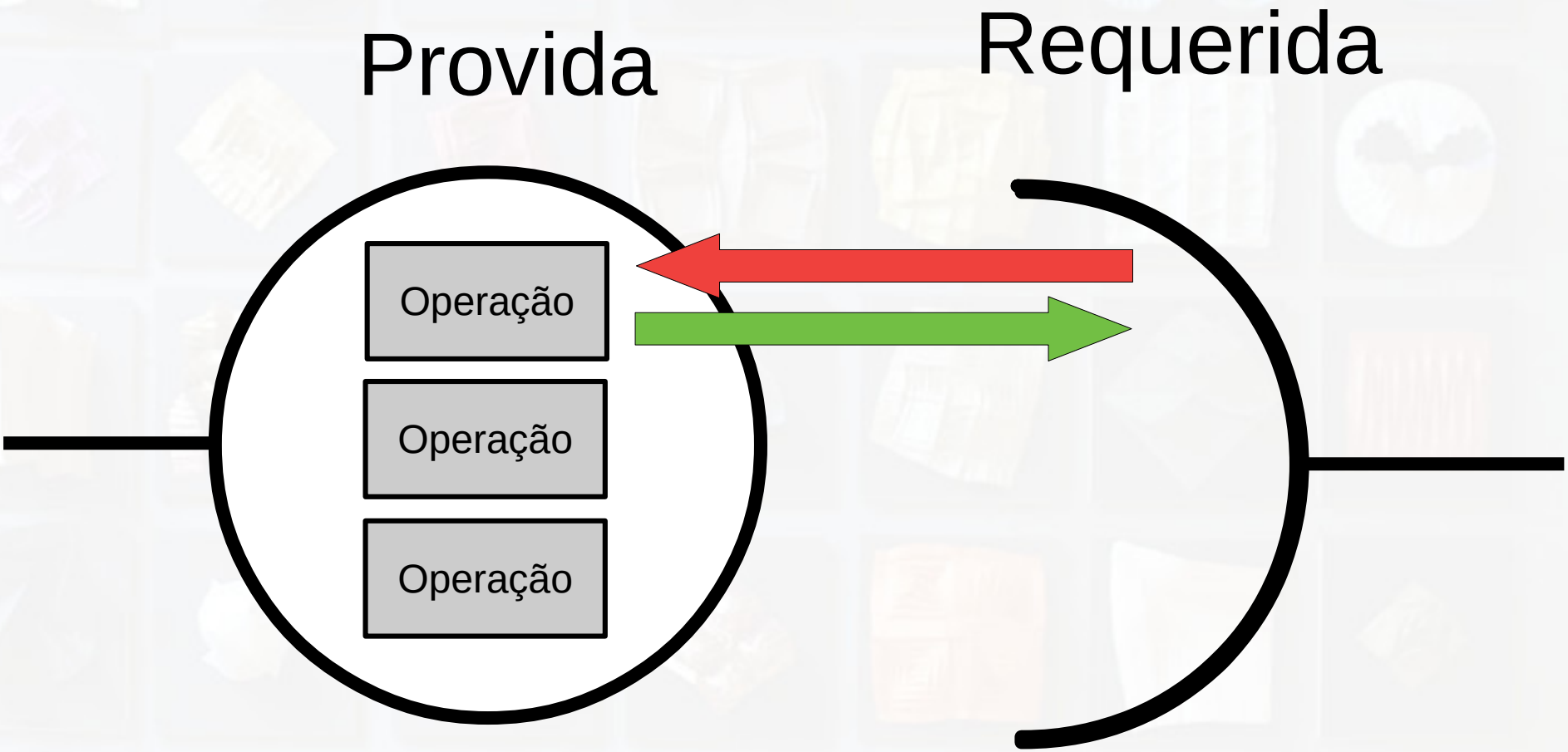


Interface Provida e Requerida

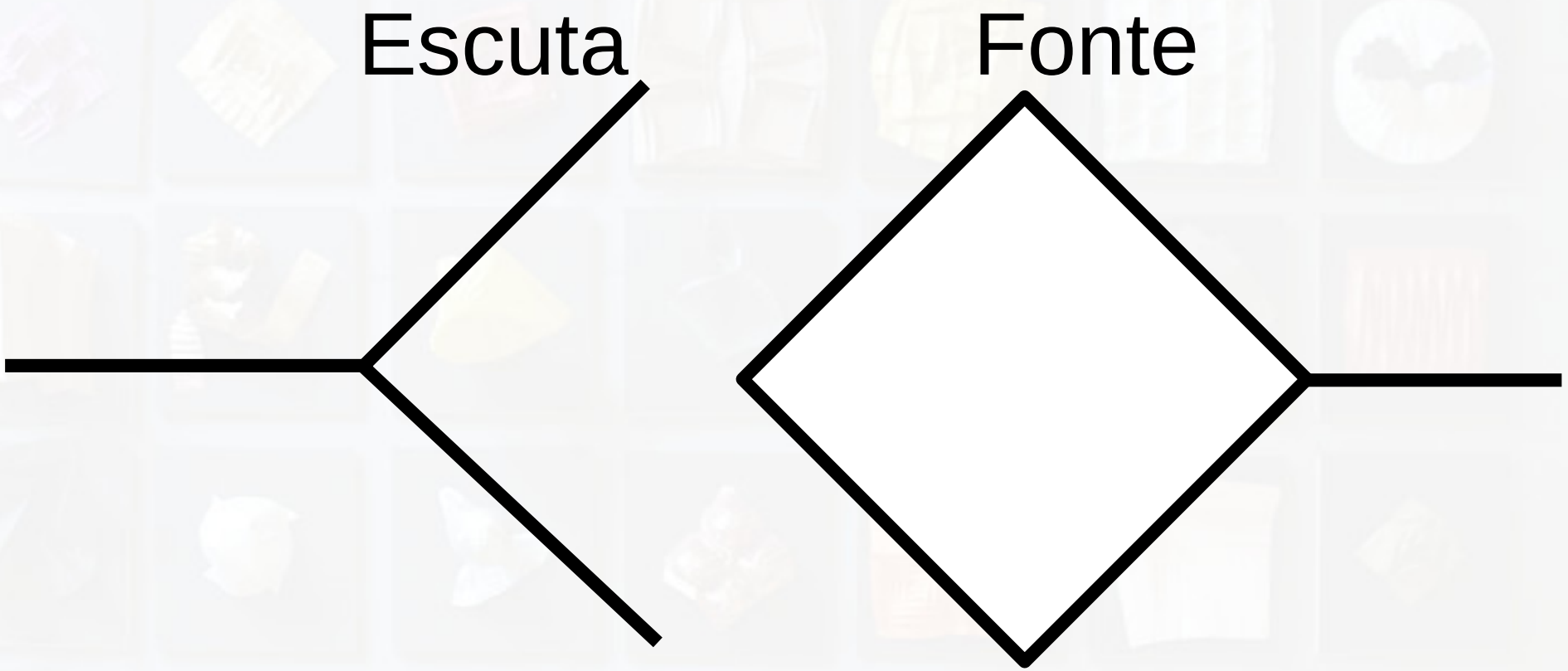


Interface Provida e Requerida

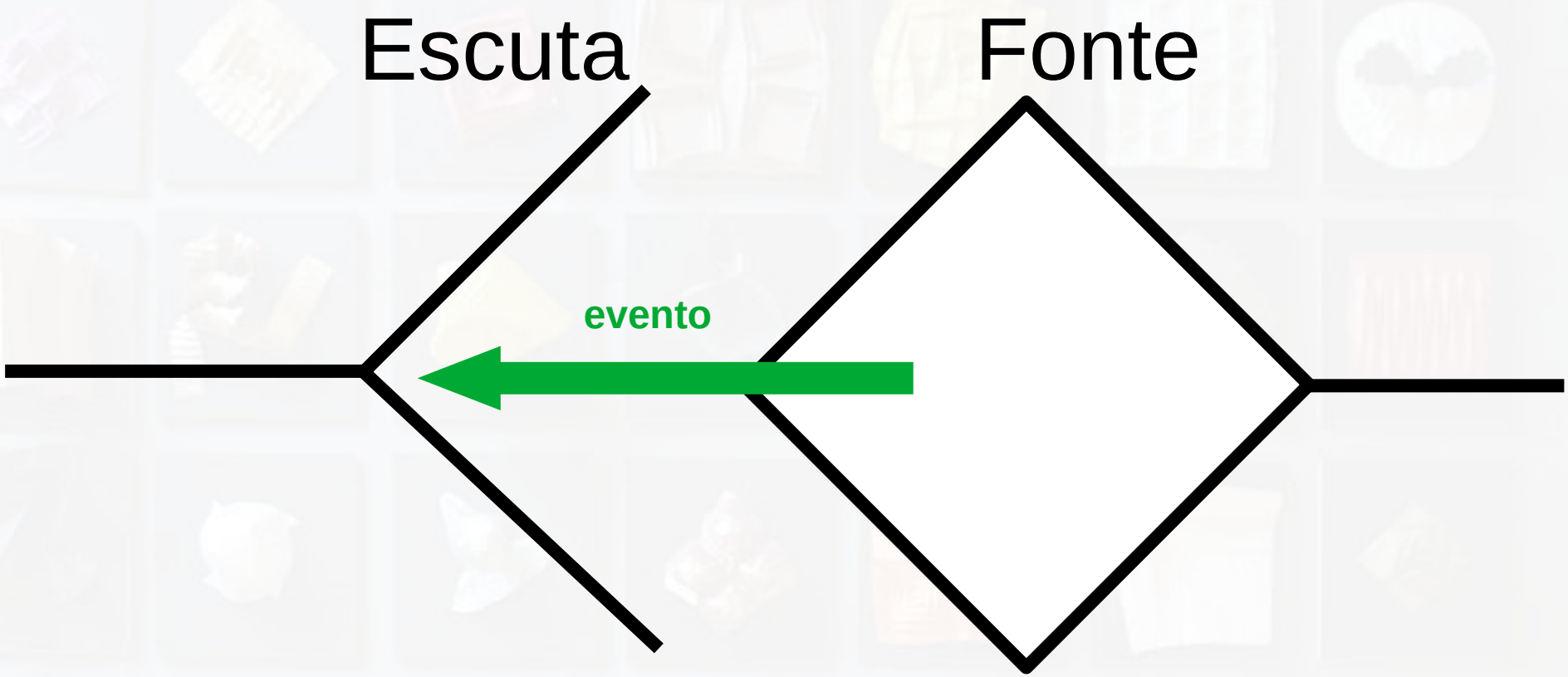
Usualmente Síncrono



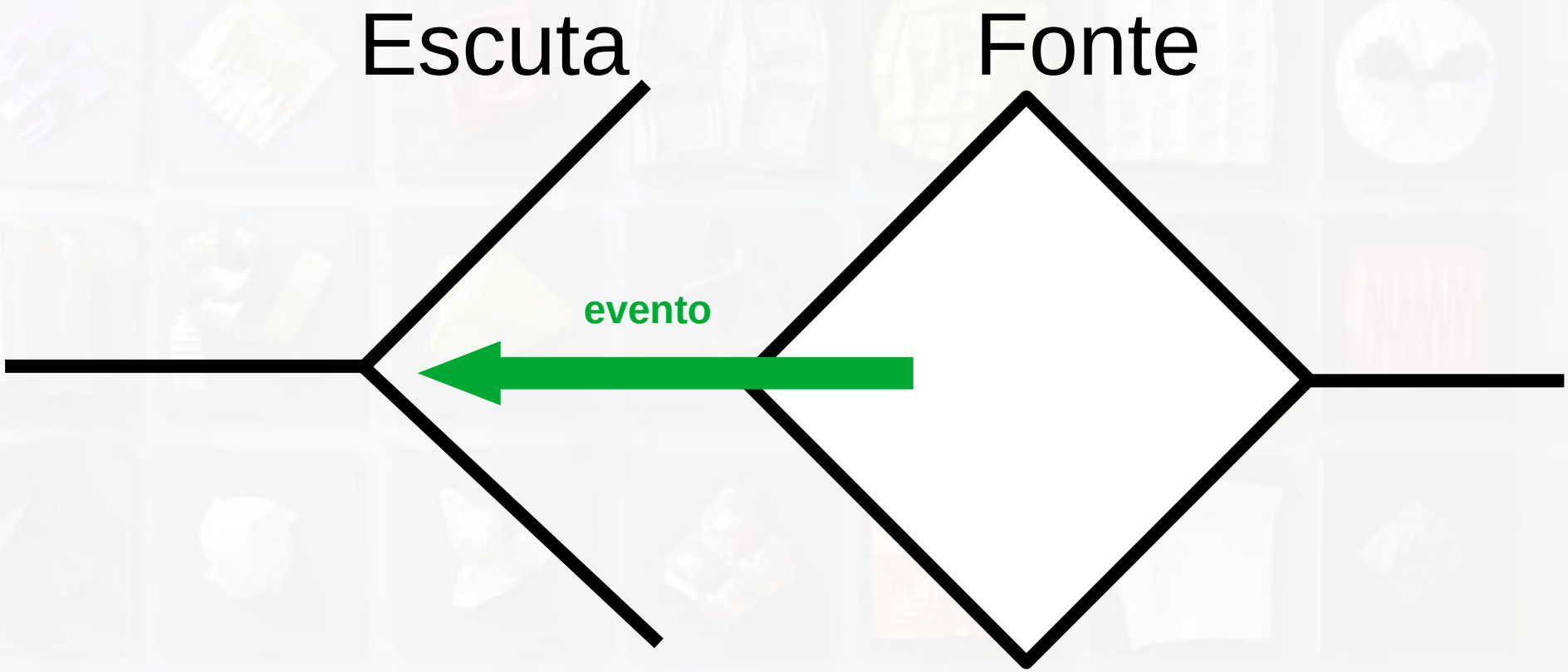
Fonte e Escuta do Evento



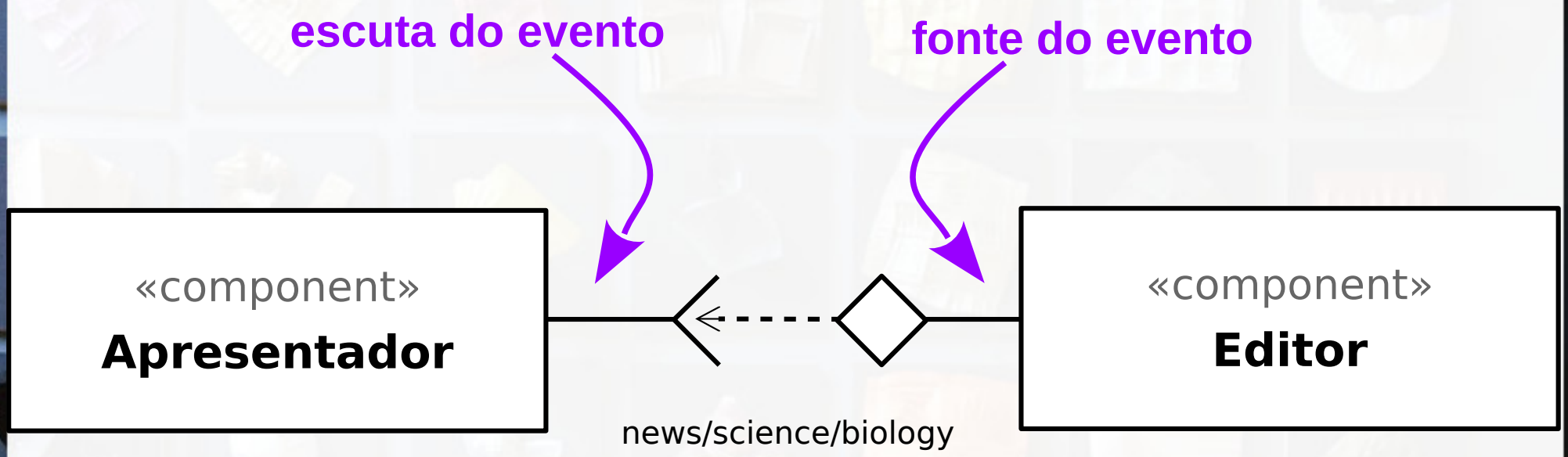
Fonte e Escuta do Evento



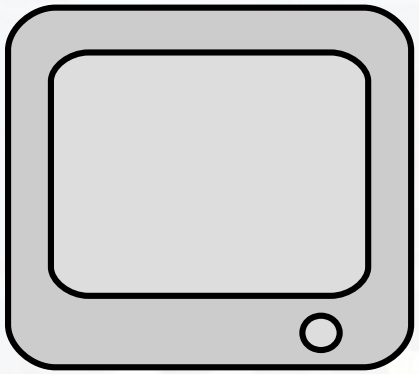
Fonte e Escuta do Evento
Usualmente Assíncrono



Fonte e Escuta do Evento (notação CORBA Component Model)



Transformando a Interface em Eventos



«component»
Apresentador

escuta do evento

news/science/biology

«component»
Editor

fonte do evento

news/science/biology

Estilos Arquiteturais

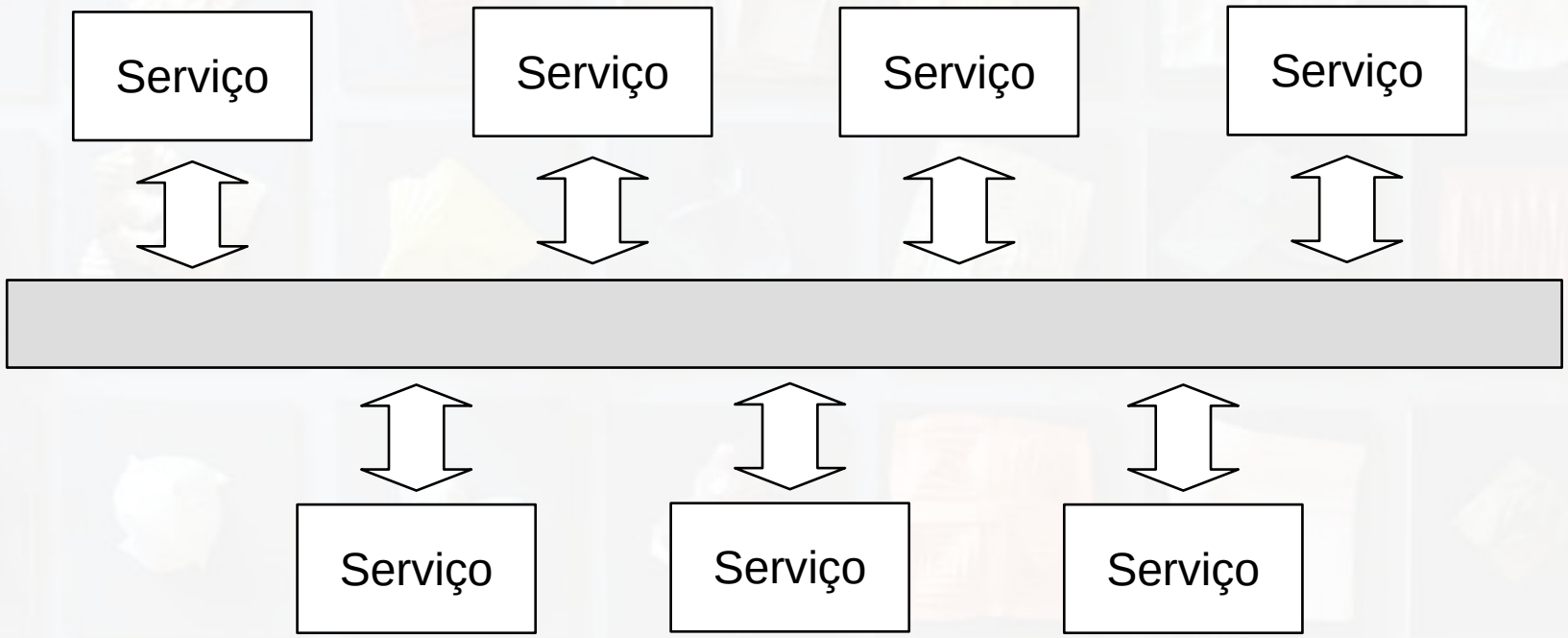
Baseado em Mensagens

Baseado em Mensagens

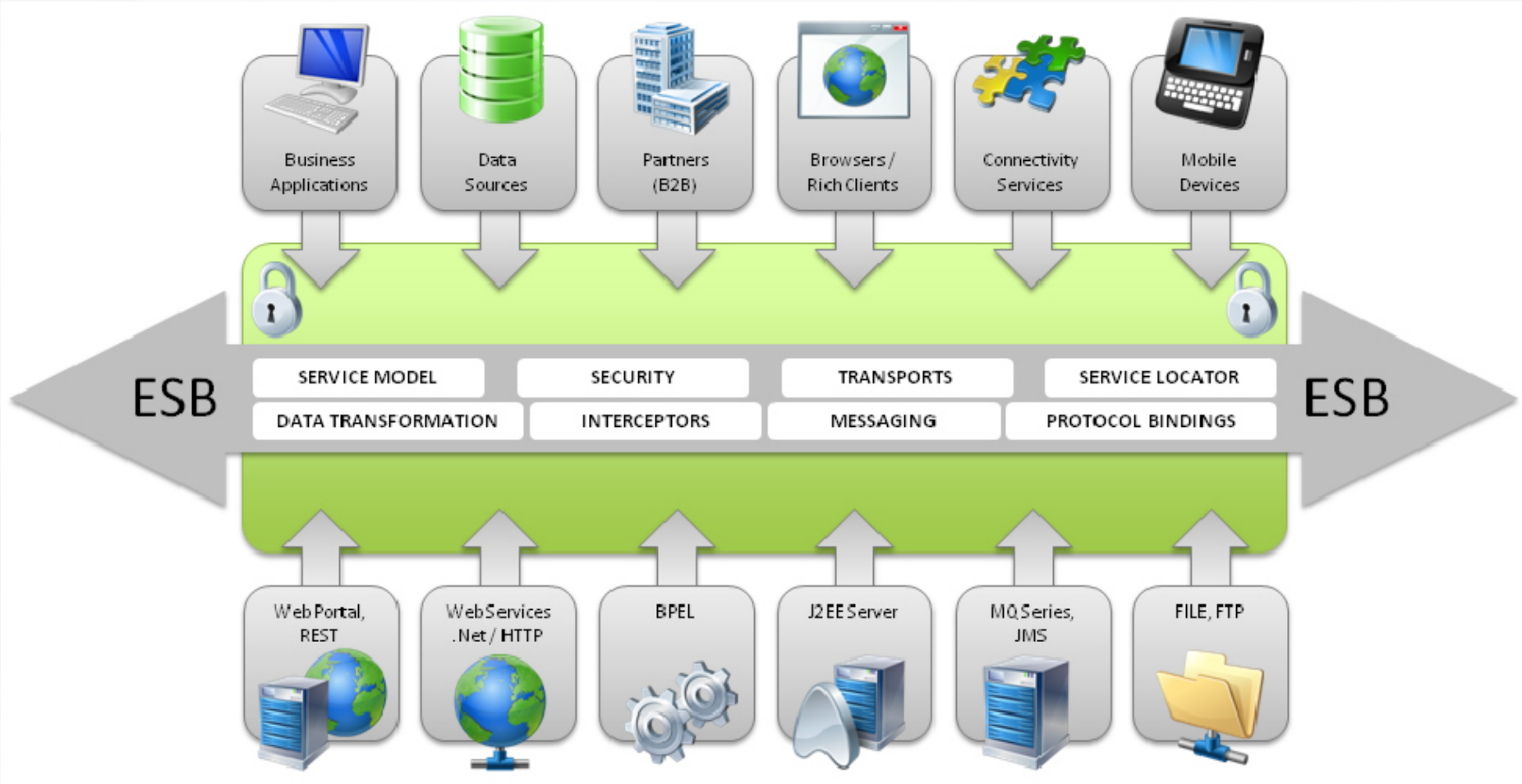
- Comunicação realizada exclusivamente pela troca de mensagens
- Tipicamente usada por aplicações assíncronas

(Taylor, 1992)

Barramento de Mensagens
Message Bus



Enterprise Service Bus (ESB)



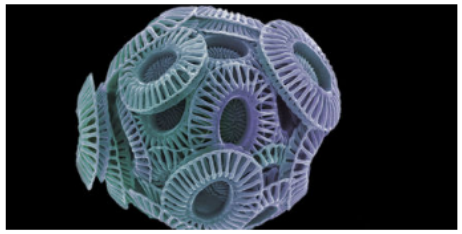
Everything you need to know about Enterprise Service Bus (ESB)
Sanchit Agrawal | October 25, 2016
https://www.hcltech.com/blogs/everything-you-need-know-about-enterprise-service-bus-esb

Agregador de Notícias

Flipboard

FOR YOU **SCIENCE** WHAT'S YOUR PASSION?

dunner99
Omnis 20h



PARTICLES

This alga may be seeding the world's skies with clouds

Sid Perkins

An algae-killing virus may be helping seed the skies with clouds. That's the implication of a new

5 likes Add comment

Beryl Starkovic
Mind, Body and Medicine 5h

MEDICAL RESEARCH

Scientists may have uncovered exactly how eating certain vegetables can prevent colon cancer

Rich Haridy

Newsflash: Vegetables are good for you! OK, so

Add comment

mthunger
Court of Leaves 17h

EVOLUTION

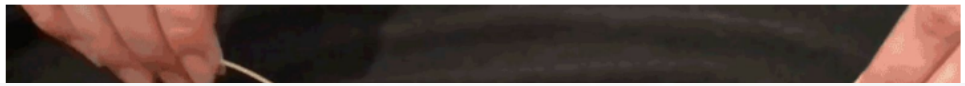
How Accurate Is Alpha's Theory of Dog Domestication?

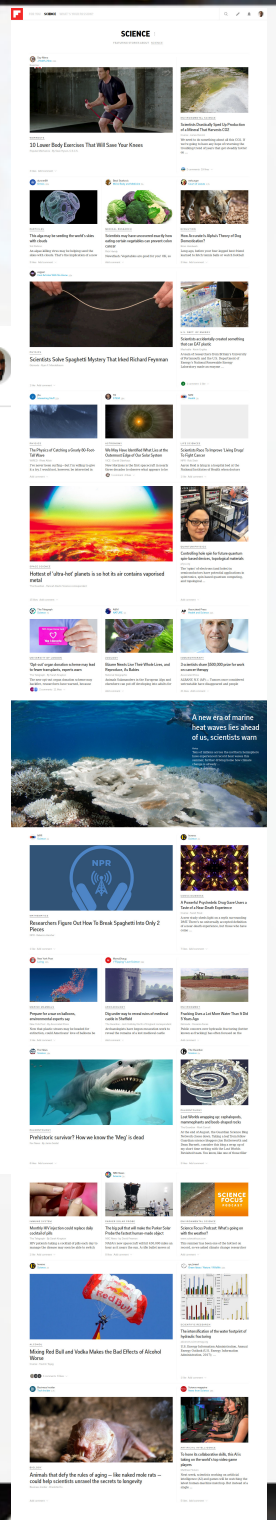
Brian Handwerk

Long ago, before your four-legged best friend learned to fetch tennis balls or watch football

3 likes Add comment

vegpwr
Cool Articles With No Home 15h





Google News

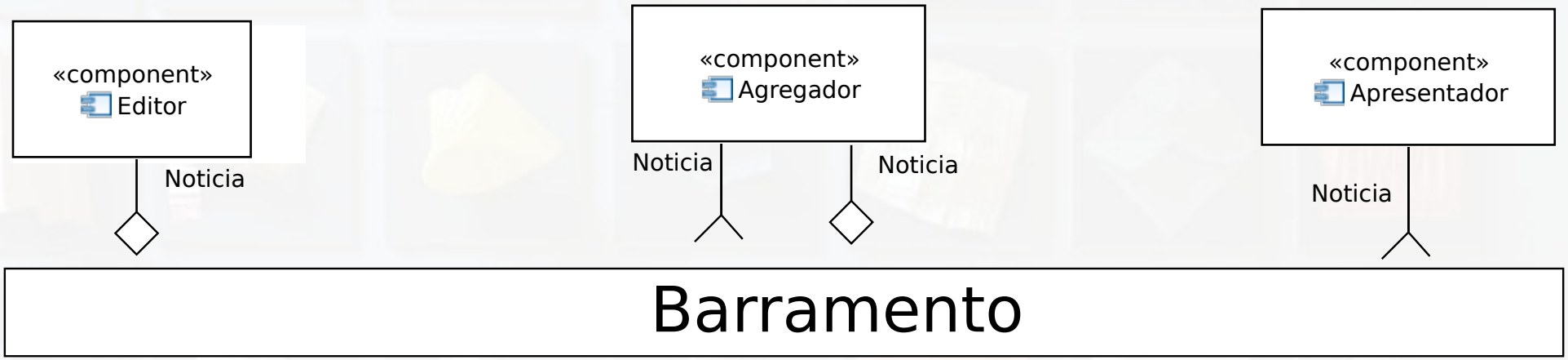

Google Notícias
1
A
Saúde
Seguir
Compartilhar
Mais recentes
Fitness
Por que ainda não sabemos tudo o que gostaríamos sobre a enxaqueca
BBC Brasil • hoje
Vacinação contra sarampo e pólio está "abaixo do esperado", diz PBH
Estado de Minas • ontem
Vacinação contra polio e sarampo acontece neste sábado (18)
PIRANOT • hoje
Ver mais

Agregador de Notícias
Papéis

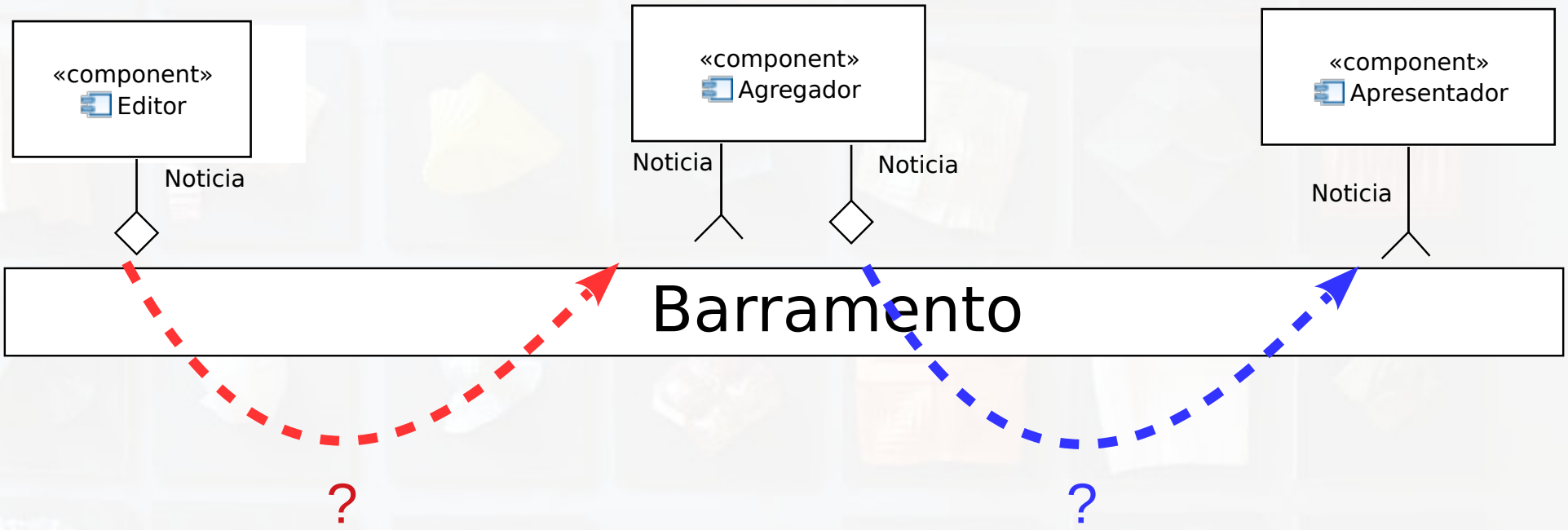
- Notícia composta de:
  - Tópico, Título, Resumo, Corpo da Notícia
- Editor
  - Publica notícias
- Agregador
  - Recebe notícias, as agrega e as publica agregadas
- Apresentador
  - Apresenta notícias

Transformando a Interface em Eventos

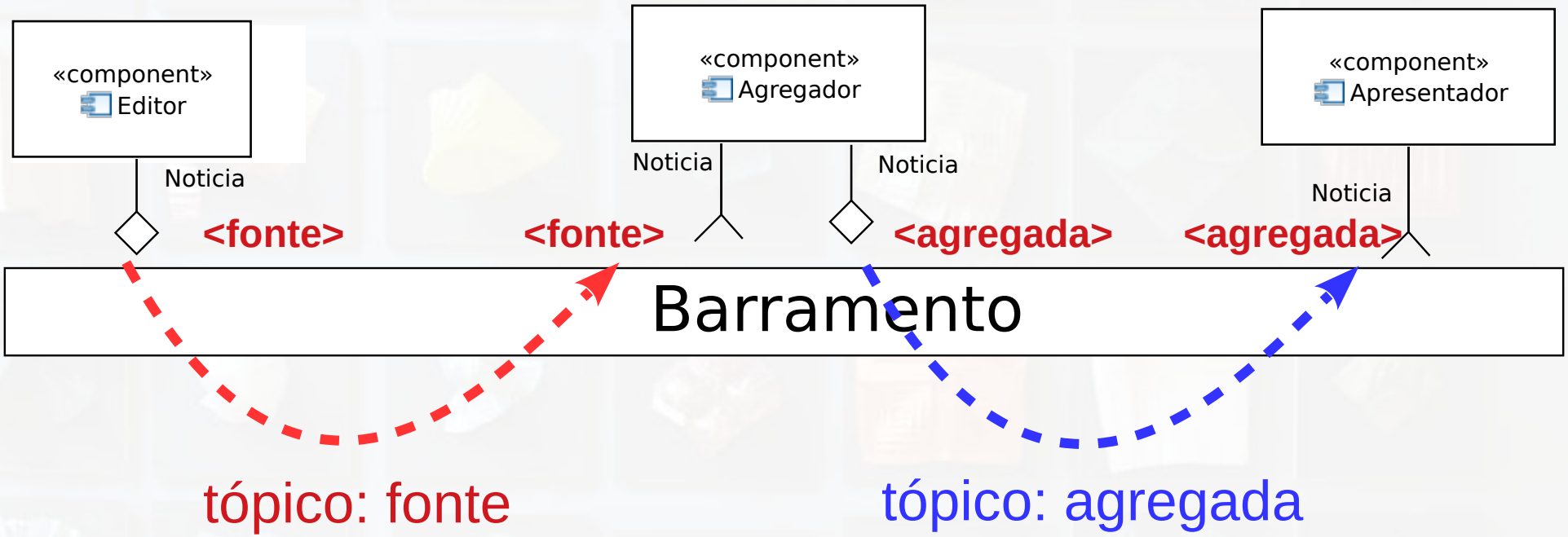
Acrescentando o Barramento



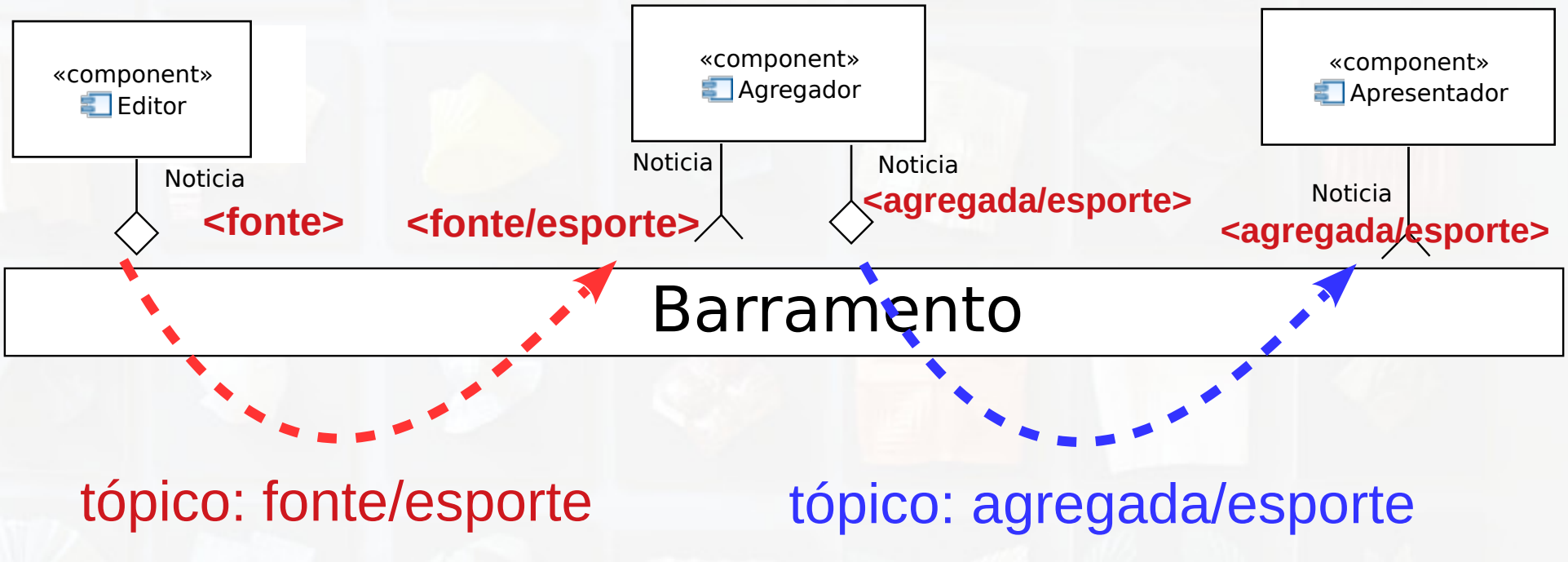
Acrescentando o Barramento



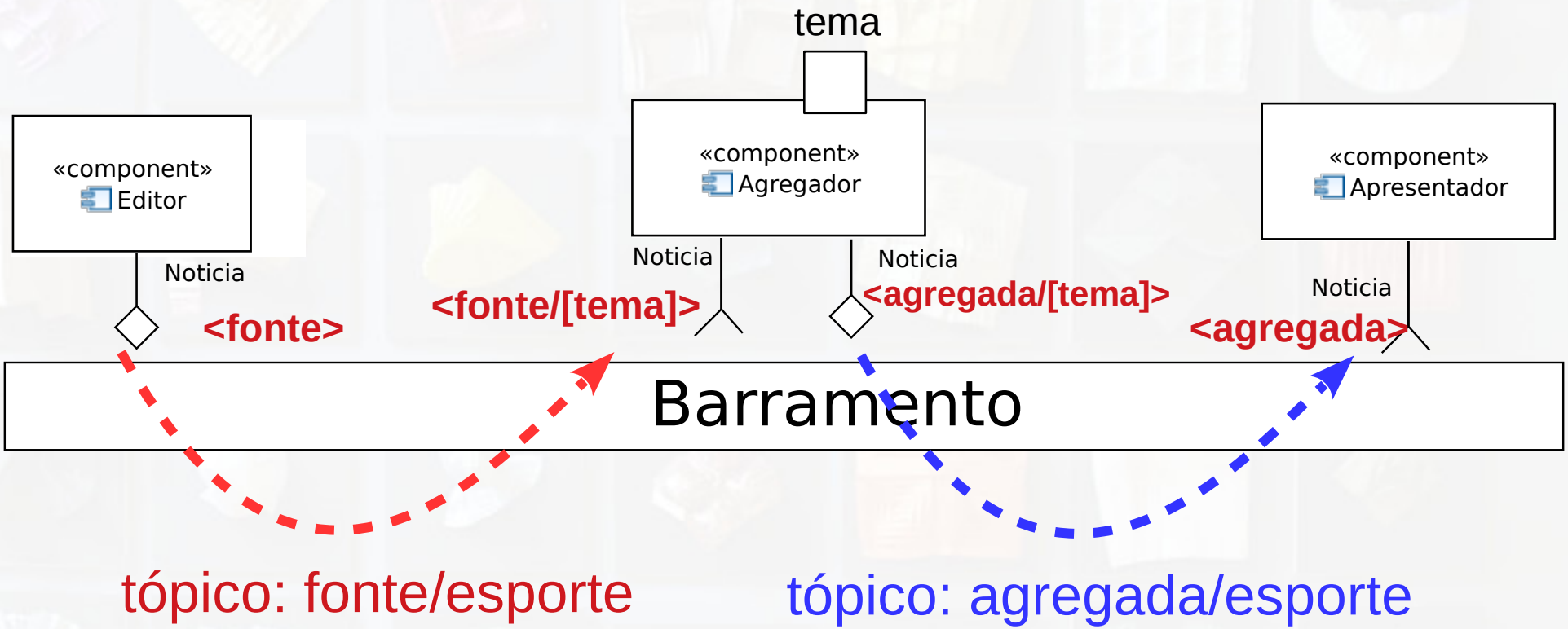
Acrescentando o Barramento



Acrescentando o Barramento



Acrescentando o Barramento



- fonte/#
- fonte/esporte/#
- fonte/esporte/futebol
- fonte/*/futebol

Transformando Objetos em Mensagens

Serialização

- Transformação do estado de um objeto em um formato de dados que possa ser armazenado ou transmitido
- Deserialização – processo inverso

Java
Interface java.io.Serializable

- Implementada por objetos que podem ser serializados
- Não define métodos
  - funciona como marcação
- Serialização padrão
  - feita na forma de reflexão
- Serialização customizada
  - Devem ser implementados métodos writeObject, readObject e readObjectNoData

Serializando e Deserializando Objetos

Formato Binário

- ObjectOutputStream → serialização
- ObjectInputStream → deserialização

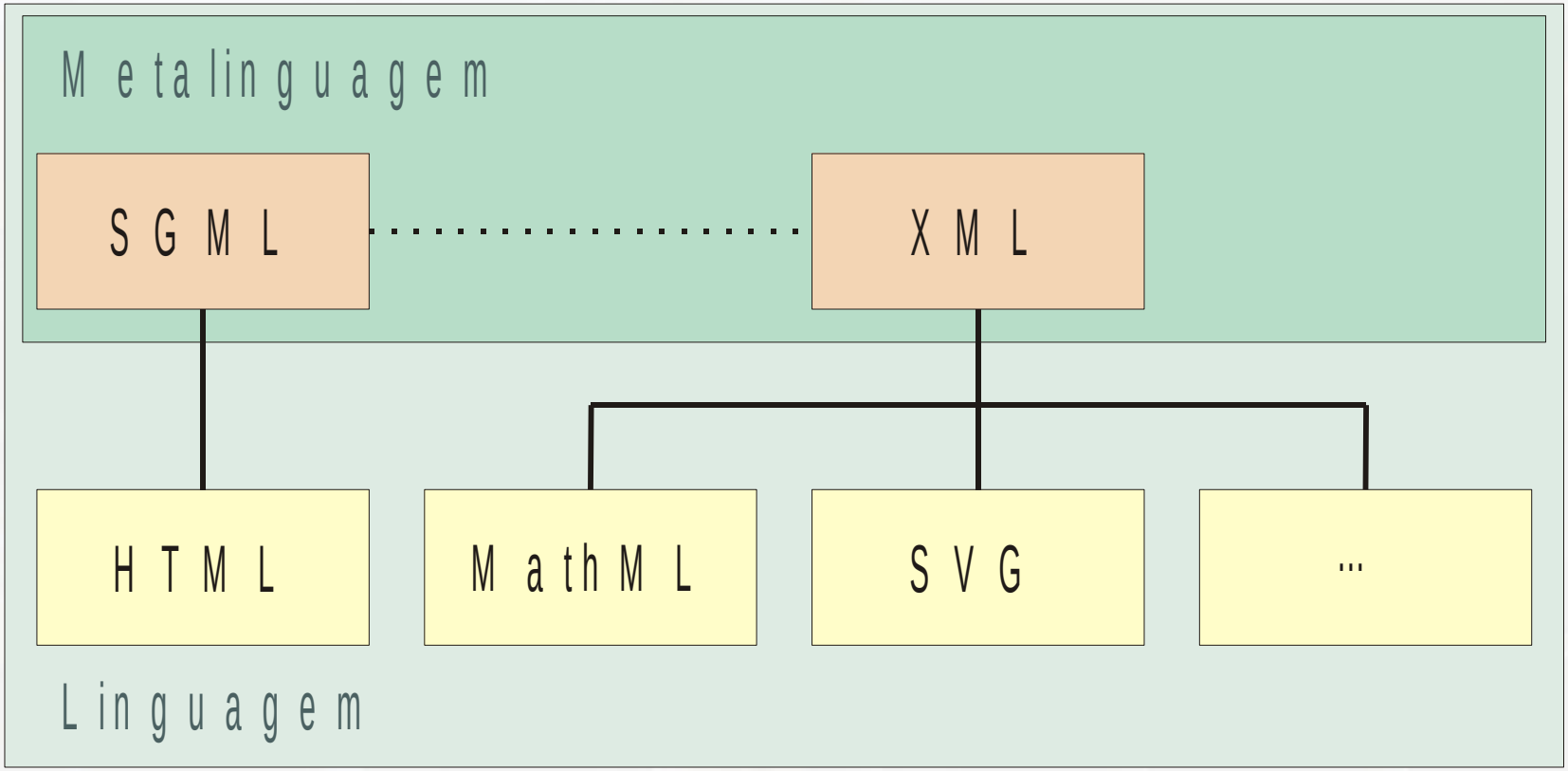
XML – eXtensible Markup Language

XML

- Lançada em 1996 como uma versão simplificada da SGML (*Standard Generalized Markup Language*), para ser utilizada na *Web*.

Metalinguagem

- Tal como SGML, XML é uma metalinguagem.
- HTML ao contrário, foi escrita em SGML.



Linguagem de Marcação

- Utiliza marcadores para agregar informações adicionais a documentos.
- Tomemos como exemplo a seguinte frase:

Horácio escreveu o livro Vida dos Dinossauros.

- Desejamos agregar informações que identifiquem quem é o **autor** e qual a **ação** realizada.

Linguagem de Marcação

- Os marcadores se diferenciam do conteúdo pelos símbolos “<” e “>” (seguem o mesmo princípio de HTML):

<autor>Horácio**</autor> <ação>**escreveu o livro Vida dos Dinossauros**</ação>**

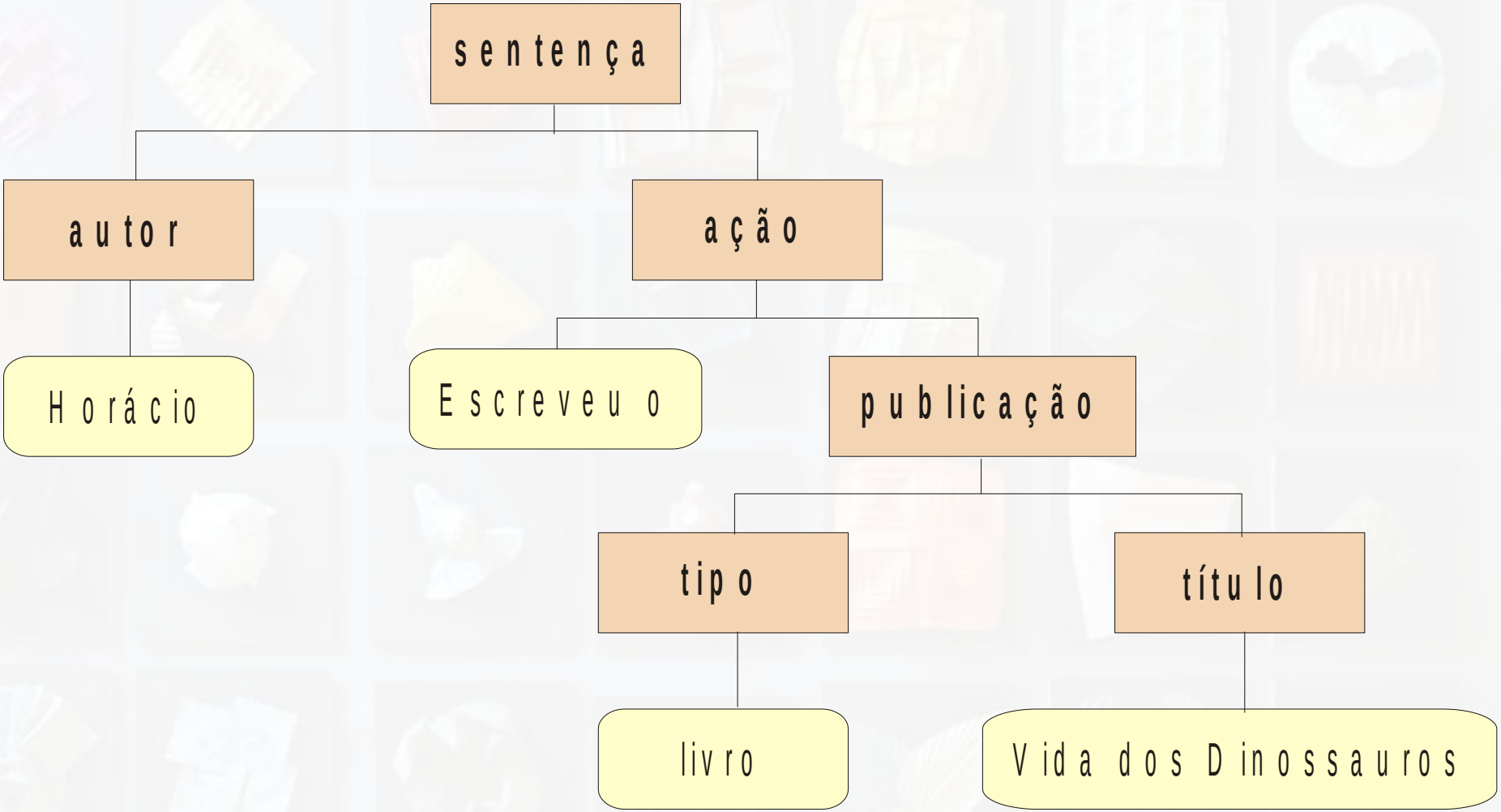
- Os marcadores delimitam unidades estruturais denominadas **elementos**.

Estrutura Hierárquica

- Marcações podem ser agrupadas hierarquicamente.
- A interpretação de cada marcador está subordinada a seu contexto.

```
<sentença>
  <autor>Horácio</autor>
  <ação>escreveu o
    <publicação>
      <tipo>livro</tipo>
      <título>Vida dos Dinossauros</título>
    </publicação>
  </ação>
</sentença>
```

Modelo de Dados XML



Elementos e Atributos

Atributos:

```
<autor cpf="487.526.548-74" nascimento="12/5/1960"> Horácio </autor>
```

- Elementos vazios:

```
<esgotado/>
```

- *Links* para elementos (#):

```
http://www.dominio.org/documento.html#bibliografia
```

- HTML usa esta estratégia em links para fragmentos.

XML e Objetos

- A estrutura hierárquica do XML combina com a estrutura hierárquica dos Objetos

Tarefa 5

- Escreva como o objeto seria representado em DTO da Tarefa 1 seria representado em XML.

Serializando e Deserializando Objetos

Formato XML

- Formato XML
  - XMLEncoder → serialização
  - XMLDecoder → deserialização

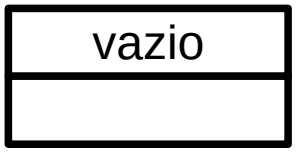
JSON
JavaScript Object Notation

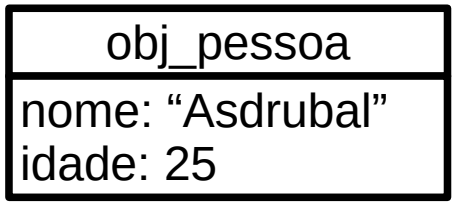
JSON

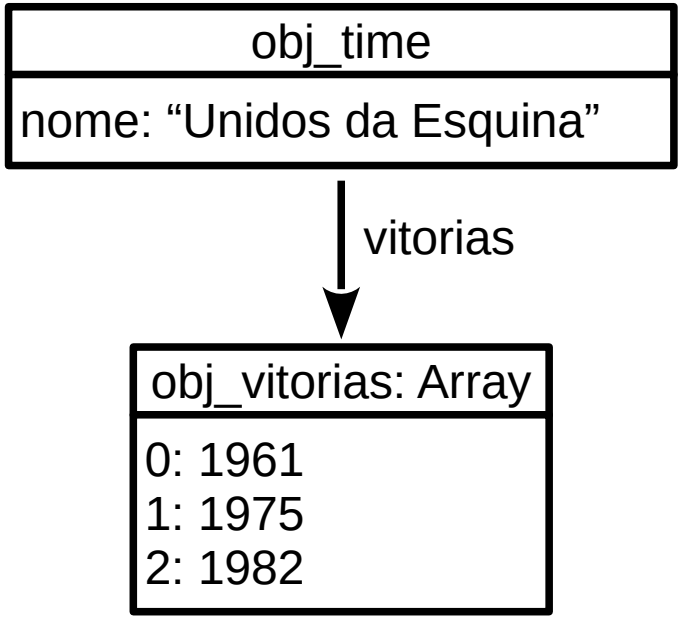
- Padrão aberto de intercâmbio de objetos
- Baseado na notação JavaScript
- Incorporado ao ECMAScript (Ecma, 2011)
- Adotado por diversas linguagens (http://json.org/)

Notação Inline de Objetos JavaScript

Objetos JS

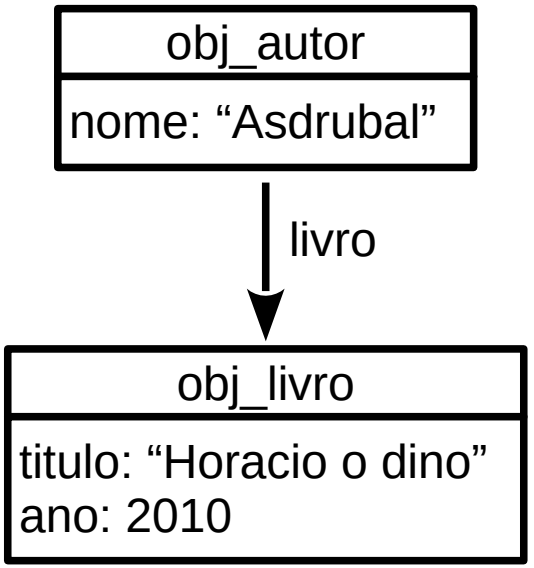
`{ }`	

`{`
`"nome": "Asdrubal",`
`"idade": 25`
`}`
	

`{`
`"nome": "Unidos da Esquina",`
`"vitorias": [1961, 1975, 1982]`
`}`
	

Objetos JS

```
{
 "nome": "Asdrubal",
 "livro": {
   "titulo": "Horacio o
dino",
   "ano": 2010
  }
}
```



Tarefa 6

- Escreva como o objeto seria representado em DTO da Tarefa 1 seria representado em JSON.

Stringify

- Serializando

```
var pessoa = {
   "nome": "Asdrubal",
   "idade": 25
};
var pessoaStr = JSON.stringify(pessoa);
```

- Deserializando

```
var pessoa2 = JSON.parse(pessoaStr);
```

Referências

- Abowd, G. D., Allen, R., Garlan, D. **Formalizing style to understand descriptions of software architecture**. ACM Trans. Softw. Eng. Methodol., ACM Press, 1995, 4, 319-364.
- Bass, L., Clements, P., Kazman, R. **Software Architecture in Practice.** Addison-Wesley, 2003.
- Cheesman, J., & Daniels, J. (2000). **UML Components: A simple process for specifying component-based software**. Addison-Wesley.
- Conway, M. E. (1968). **How Do Committees Invent?** Datamation, 14(5), 28–31.

Referências

- Fowler, M. (2003). **Data Transfer Object**. Retrieved from https://martinfowler.com/eaaCatalog/dataTransferObject.html
- Garlan, D. et al. **Architectural Mismatch (Why It's Hard to Build Systems Out of Existing Parts)**. Proceedings, 17th Int. Conf. on Software Engineering. Seattle, WA, April 23-30, 1995.
- He, H. **What Is Service-Oriented Architecture**. Setembro 2003. Disponível em http://www.xml.com/pub/a/ws/2003/09/30/soa.html
- ISO/IEC/IEEE 24765:2010 **Systems and software engineering — Vocabulary**

Referências

- Jha, P. C., Bali, V., Narula, S., & Kalra, M. (2014). **Optimal component selection based on cohesion & coupling for component based software system under build-or-buy scheme**. Journal of Computational Science, 5(2), 233–242.
- Papazoglou, M. P., Georgakopoulos, D. **Service-oriented computing.** Commun. ACM, 2003, 46, 25-28.
- Parnas, D. **On the Design and Development of Program Families**. IEEE Transactions on Software Engineering SE-2, 1976, 1, 1-9.

Referências

- Software Engineering Standards Committee of the IEEE Computer Society. **Systems and software engineering – Recommended practice for architectural description of software-intensive systems**, ISO/IEC 42010 IEEE Std 1471-2000 First edition 2007-07-15, Julho 2007.
- Sommerville, I. (2007) **Software Engineering**, 8th. ed. Addison Wesley.
- Stevens, W. P., Myers, G. J., & Constantine, L. L. (1974). **Structured design**. IBM Systems Journal, 13(2), 115–139.
- Szyperski, C. **Component Software: Beyond Object-Oriented Programming**. Addison-Wesley Longman Publishing Co., Inc., 2002.
- Taylor, R. N. , et al. **A Component- and Message-Based Architectural Style for GUI Software**. IEEE Trans. Software Engineering, IEEE Press, 1996, 22, 390-406.

Referências

- Comella-Dorda, S. **Component Object Model (COM), DCOM, and Related Capabilities**. Carnegie Mellon University, março de 2001.
- Cook, S., Bock, C., Rivett, P., Rutt, T., Seidewitz, E., Selic, B., & Tolbert, D. (2015). **OMG Unified Modeling Language (OMG UML) - version 2.5**. Needham. Retrieved from http://www.omg.org/spec/UML/2.5/
- Gamma, E. Helm, R. Johnson, R. Vlissides, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1995.
- Krueger, C. W. Software Reuse. ACM Comput. Surv., ACM Press, 1992, 24, 131-183.
- Liskov, B. **Keynote address - data abstraction and hierarchy**. OOPSLA '87: Addendum to the proceedings on Object-oriented programming systems, languages and applications (Addendum), ACM Press, 1987, 17-34.
- Martin, R. C. **Design Principles and Design Patterns**. Object

Referências

- Meyer, B. (1992). **Applying “design by contract.”** Computer, 25(10), 40–51. https://doi.org/10.1109/2.161279
- Meyer, B. (2000) **Object-Oriented Software Construction** (2nd Edition). Prentice Hall.
- Parrish, R. **XPCOM Part 1: An introduction to XPCOM**. DeveloperWorks, fevereiro de 2001, on-line: http://www.ibm.com/developerworks/webservices/library/co-xpcom.html
- Willliams, S. & Kindel, C. **The Component Object Model: A Technical Overview**. Microsoft Corporation, 1994

Bibliografia

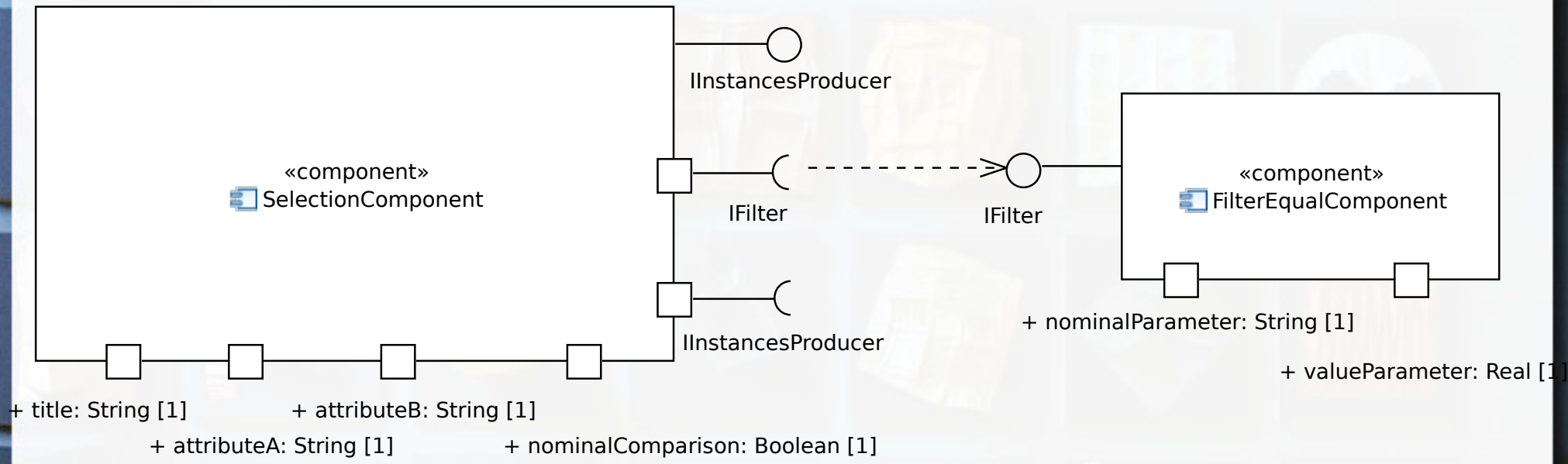
- Bartlett, Dave. **CORBA Component Model (CCM) – Introducing next-generation CORBA**. April 2001.
- OMG – Object Management Group. **CORBA Components – Version 3.0 – formal/02-06-65**. Junho 2002.
- OMG – Object Management Group. **CORBA Componente Model Specification – version 4.0 – formal/06-04-01**. Abril 2006.
- Wang, Nanbor; Schmidt, Douglas C. and O'Ryan, Carlos. **Overview of the CORBA Component Model**, in Component Based Software Engineering: Putting the Pieces Together. Addison-Wesley Pub Co, 2001.

André Santanchè
http://www.ic.unicamp.br/~santanche

Licença

- Estes slides são concedidos sob uma Licença Creative Commons. Sob as seguintes condições: Atribuição, Uso Não-Comercial e Compartilhamento pela mesma Licença.
- Mais detalhes sobre a referida licença Creative Commons veja no link: https://creativecommons.org/licenses/by-nc-sa/4.0/deed.pt_BR
- Agradecimento a Goran Konjevod [https://www.flickr.com/photos/23913057@N05/] por sua fotografia “50-50 Show III” usada na capa e nos fundos, disponível em [https://flic.kr/p/advD33]
vide licença específica da fotografia.

Componente Filter

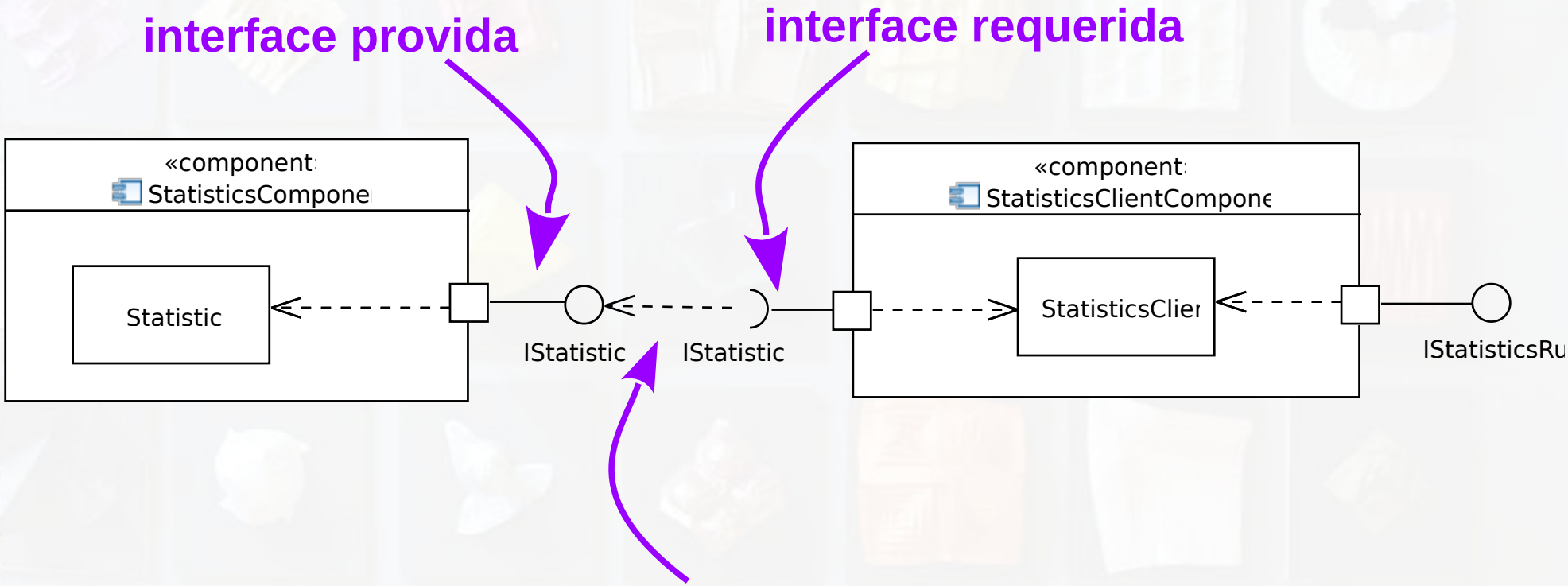


«Interface»
IFilter

+ filter(in toEvaluate: String): Boolean
+ filter(in toEvaluate: Real): Boolean
+ filter(in toCompare1: String, in toCompare2: String): Boolean
+ filter(in toCompare1: Real, in toCompare2: Real): Boolean

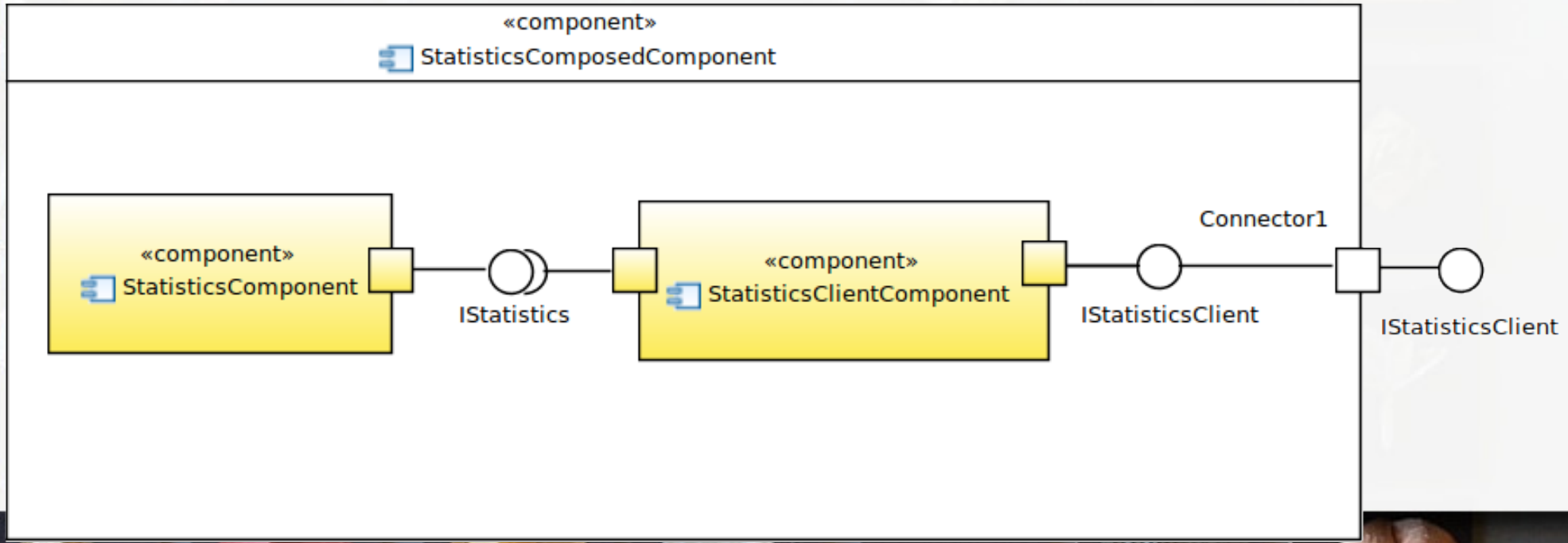
Hierarquia de Composição:
componentes compostos de composição de componentes

Retornando ao Statistics



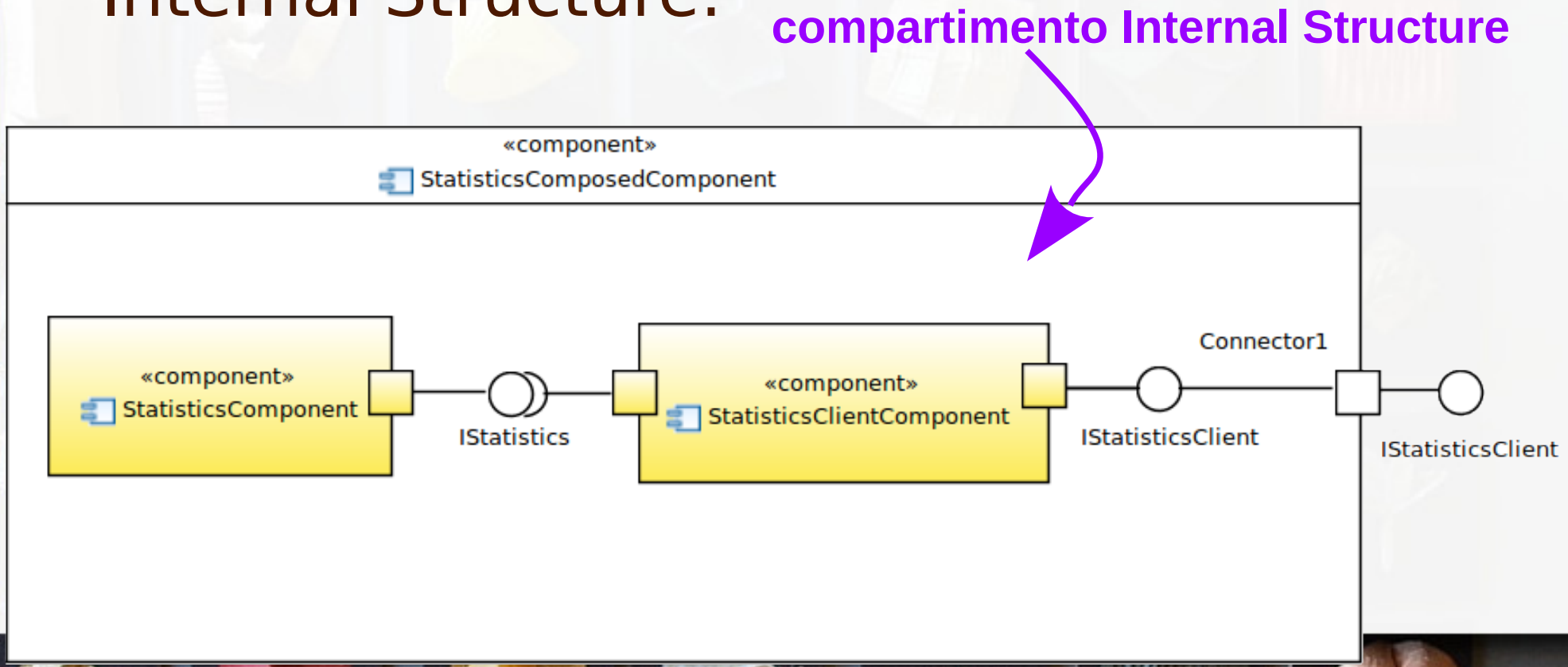
Componentes feitos de Composição

- Componente de granularidade maior (`StatisticsComposedComponent`) construído pela composição de componentes de granularidade menor (`StatisticsComponent` e `StatisticsClientComponent`).



Internal Structure e White-box

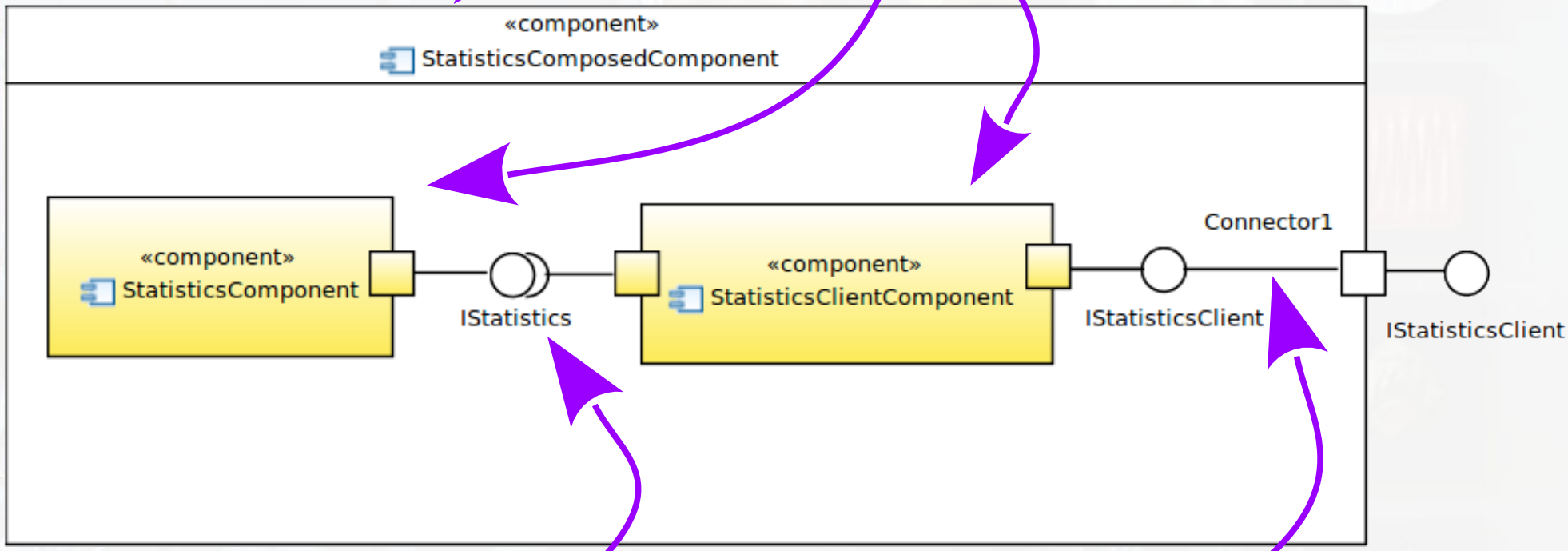
- Visão da estrutura interna do componente no estilo white-box, representado no compartimento de Internal Structure:



Composição de Componentes

componente maior

componentes menores
que fazem parte da composição



interface provida conectada
à interface requerida

conector de delegação:
delega a interface interna
para a externa

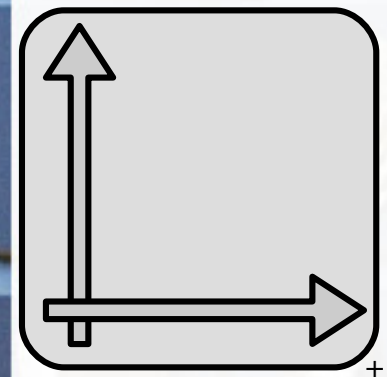
Tarefa 2

- Componha os componentes `SelectionComponent` e `FilterEqualComponent` em um componente de granularidade maior.

Tarefa 3

- Identifique componentes da biblioteca que podem oferecer ou requerer interfaces comuns.
- Modifique a interface desses componentes para minimizar e/ou simplificar o uso de interfaces.

Tarefa 3 (Componentes)

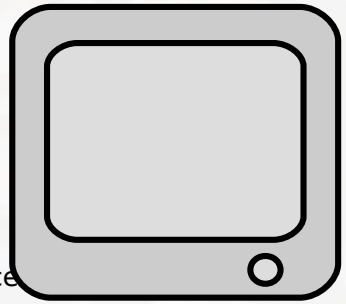


«component»
ChartComponent

IRun

IColumnProducer

+ title: String [1]



«component»
ConsoleComponent

IConsoleUpdate

IInstancesProducer

IColumnProducer

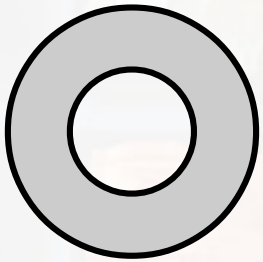
«component»
ProjectionComponent

IColumnProducer

IInstancesProducer

+ title: String [1]

+ attribute: String [1]



«component»
SelectionComponent

IInstancesProducer

IFilter

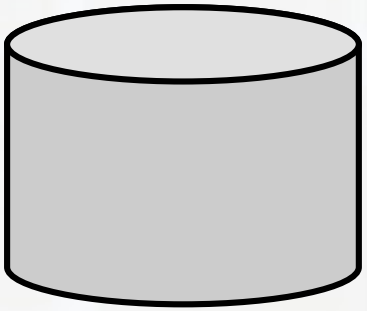
IInstancesProducer

+ title: String [1]

+ attributeB: String [1]

+ attributeA: String [1]

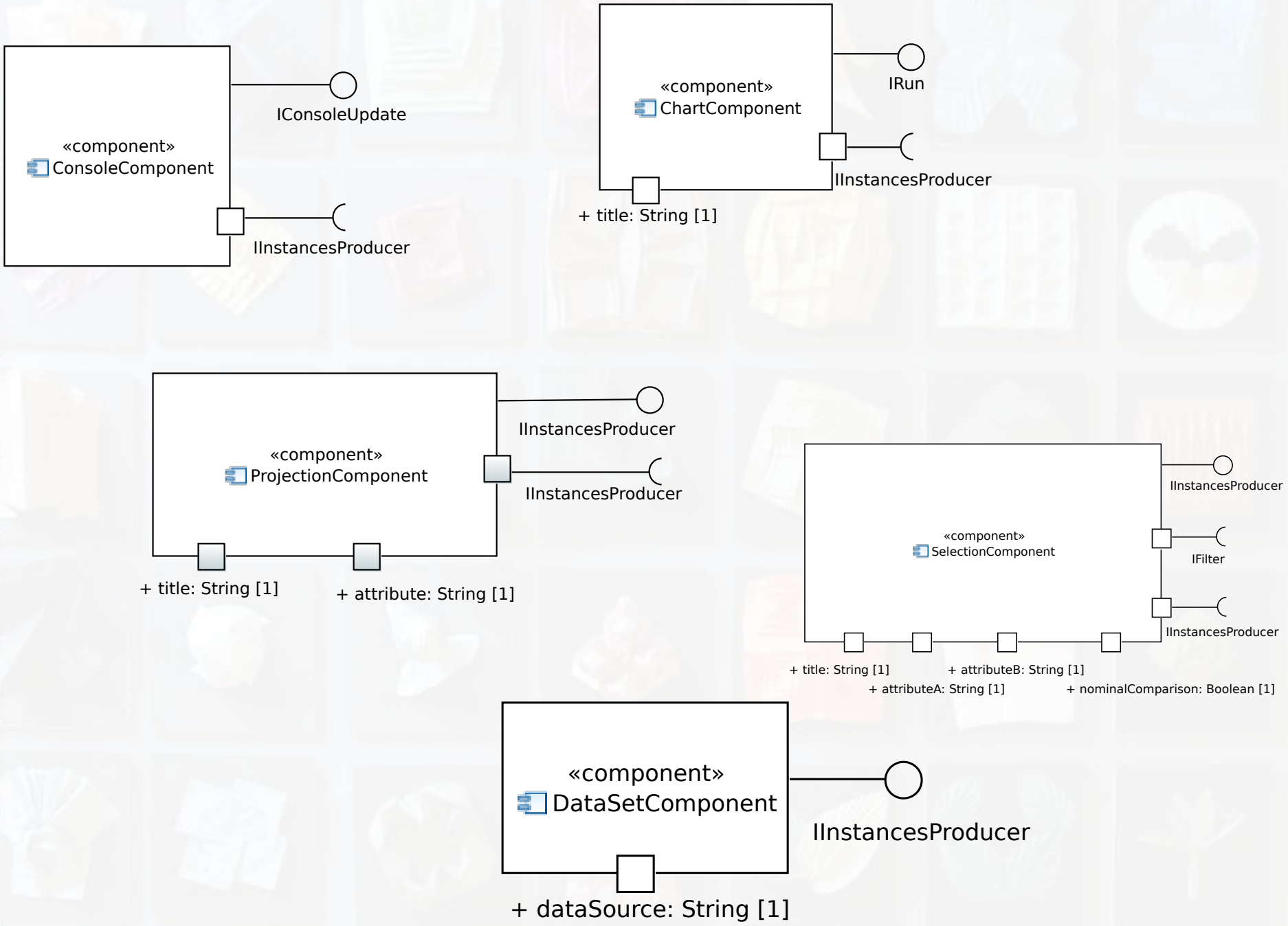
+ nominalComparison: Boolean [1]

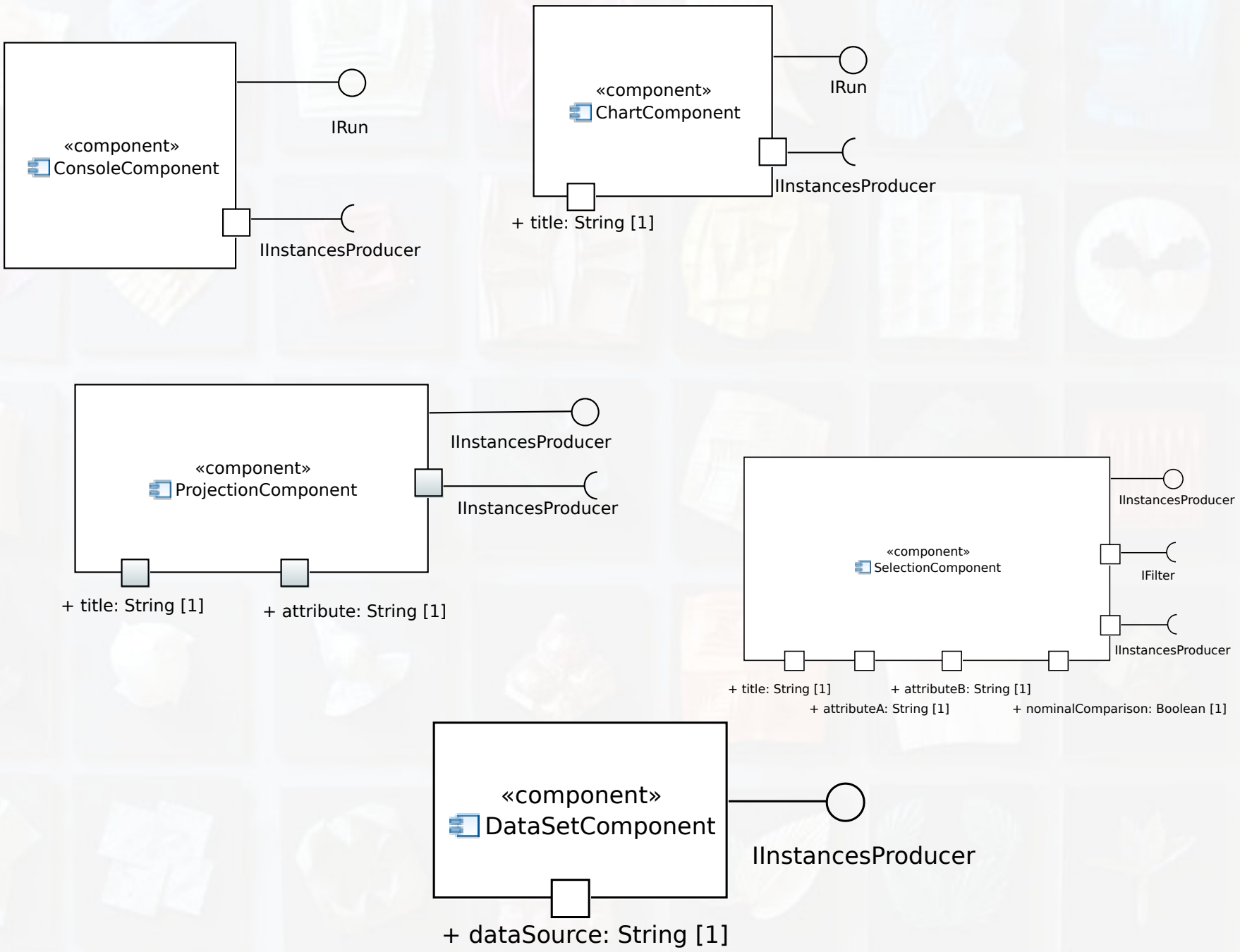


«component»
DataSetComponent

IInstancesProducer

+ dataSource: String [1]


«component»
ConsoleComponent
IConsoleUpdate
IInstancesProducer
«component»
ChartComponent
IRun
IInstancesProducer
+ title: String [1]
«component»
ProjectionComponent
IInstancesProducer
IInstancesProducer
+ title: String [1]
+ attribute: String [1]
«component»
SelectionComponent
IInstancesProducer
IFilter
IInstancesProducer
+ title: String [1]
+ attributeB: String [1]
+ attributeA: String [1]
+ nominalComparison: Boolean [1]
«component»
DataSetComponent
IInstancesProducer
+ dataSource: String [1]


«component»
ConsoleComponent
IRun
IInstancesProducer
«component»
ChartComponent
IRun
IInstancesProducer
+ title: String [1]
«component»
ProjectionComponent
IInstancesProducer
IInstancesProducer
+ title: String [1]
+ attribute: String [1]
«component»
SelectionComponent
IInstancesProducer
IFilter
IInstancesProducer
+ title: String [1]
+ attributeB: String [1]
+ attributeA: String [1]
+ nominalComparison: Boolean [1]
«component»
DataSetComponent
IInstancesProducer
+ dataSource: String [1]

Estilos Arquiteturais

Decomposição Modular

Pipe & Filter

Pipe & Filter

- Bastante popular em sistemas operacionais UNIX-like
- Processo incremental
  - vai gerando os dados de saída, sem esperar que a entrada de dados se complete (Garlan, 1993).
- Invariantes (Garlan, 1993)
  - entidades independentes
  - identidades de entrada e saída desconhecidas
  - especificação local

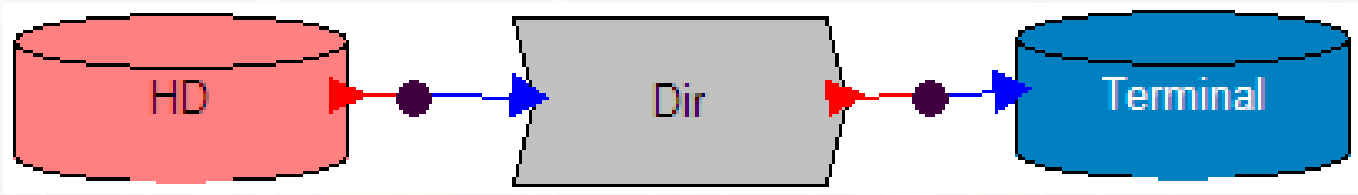
Pipe & Filter

- *Filter* (componente)
  - Lê fluxos de dados de entrada e produz seus resultados como fluxos de dados de saída.

- *Pipe* (conector)
  - Conduzem o fluxo, conectando o fluxo de saída de um filter ao fluxo de entrada de outro filter.

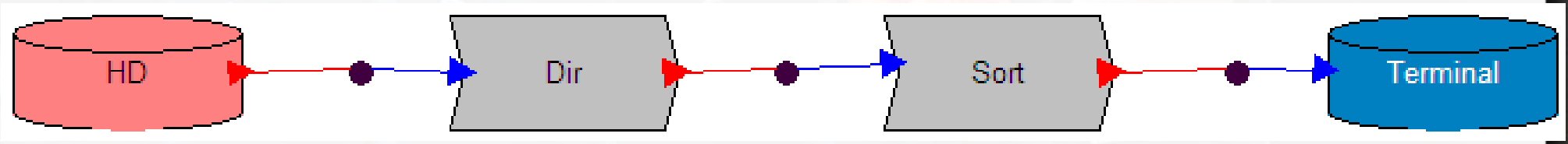
Pipe & Filter UNIX-like

- Lista nome dos arquivos
  - dir /b



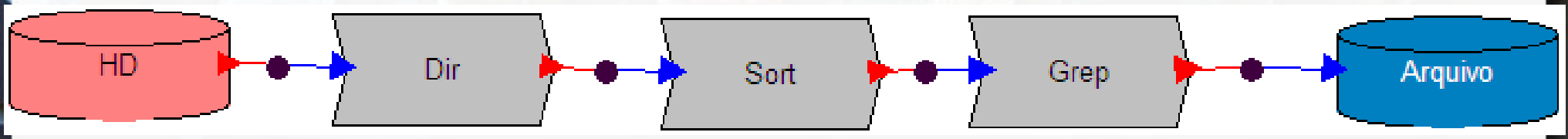
Pipe & Filter UNIX-like

- Operador de pipe no DOS e Unix: |
- Lista nome dos arquivos “pipe” coloca em ordem alfabética
  - dir /b | sort



Pipe & Filter Unix-like

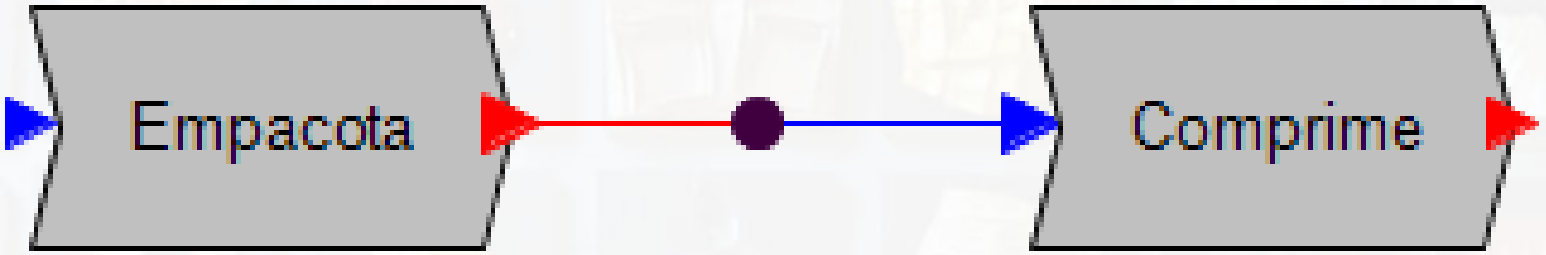
- Lista nome dos arquivos “pipe” coloca em ordem alfabética “pipe” recorta aqueles que têm o trecho “Win”
  - dir /b | sort | grep "Win"
- Redireciona saída (pipe) no DOS: >
- Mesmo anterior com saída para arquivo “resultado.txt”
  - dir /b | sort | grep "Win" >resultado.txt



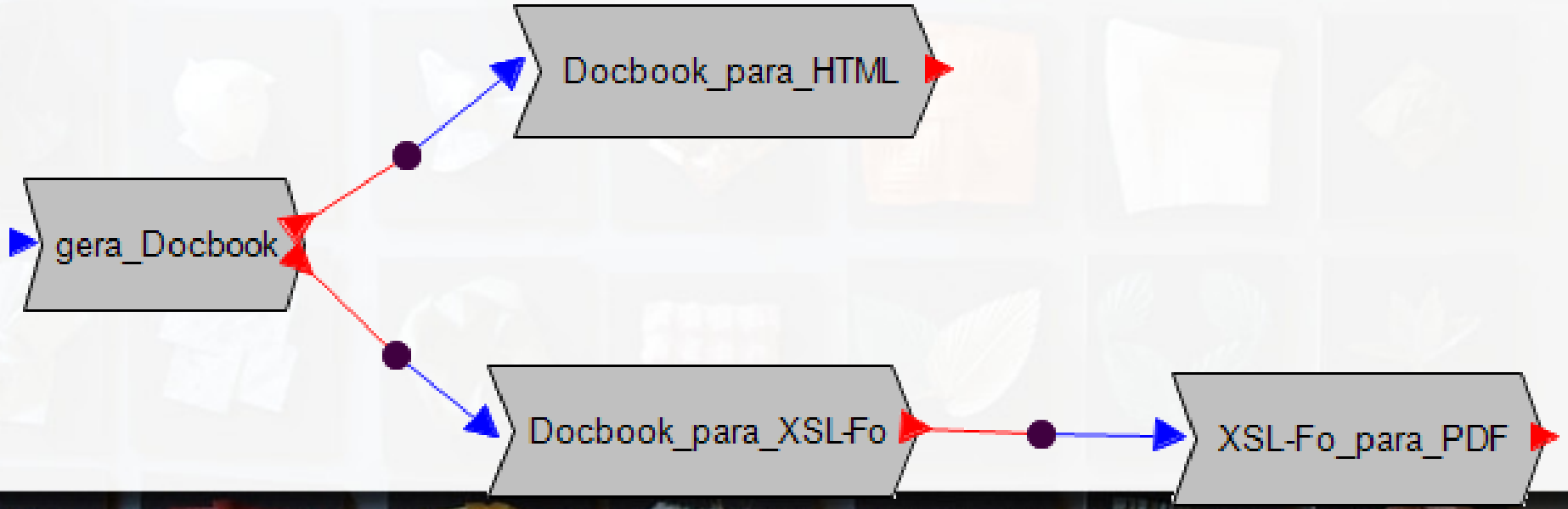
Pipe & Filter

Exemplos

- Empacotando e comprimindo



- Docbook



Vantagens do Pipe & Filter

- Suporta reúso de transformações
- Organização intuitiva para a comunicação das partes
- Fácil de adicionar novas transformações
- Relativamente simples de implementar tanto em sistemas concorrentes como sequenciais."[1]

(Sommerville, 2007)

1. "- Supports transformation reuse.
   - Intuitive organisation for stakeholder communication.
   - Easy to add new transformations.
   - Relatively simple to implement as either a concurrent or sequential system." (Sommerville,

Pipe & Filter – Implementação Java

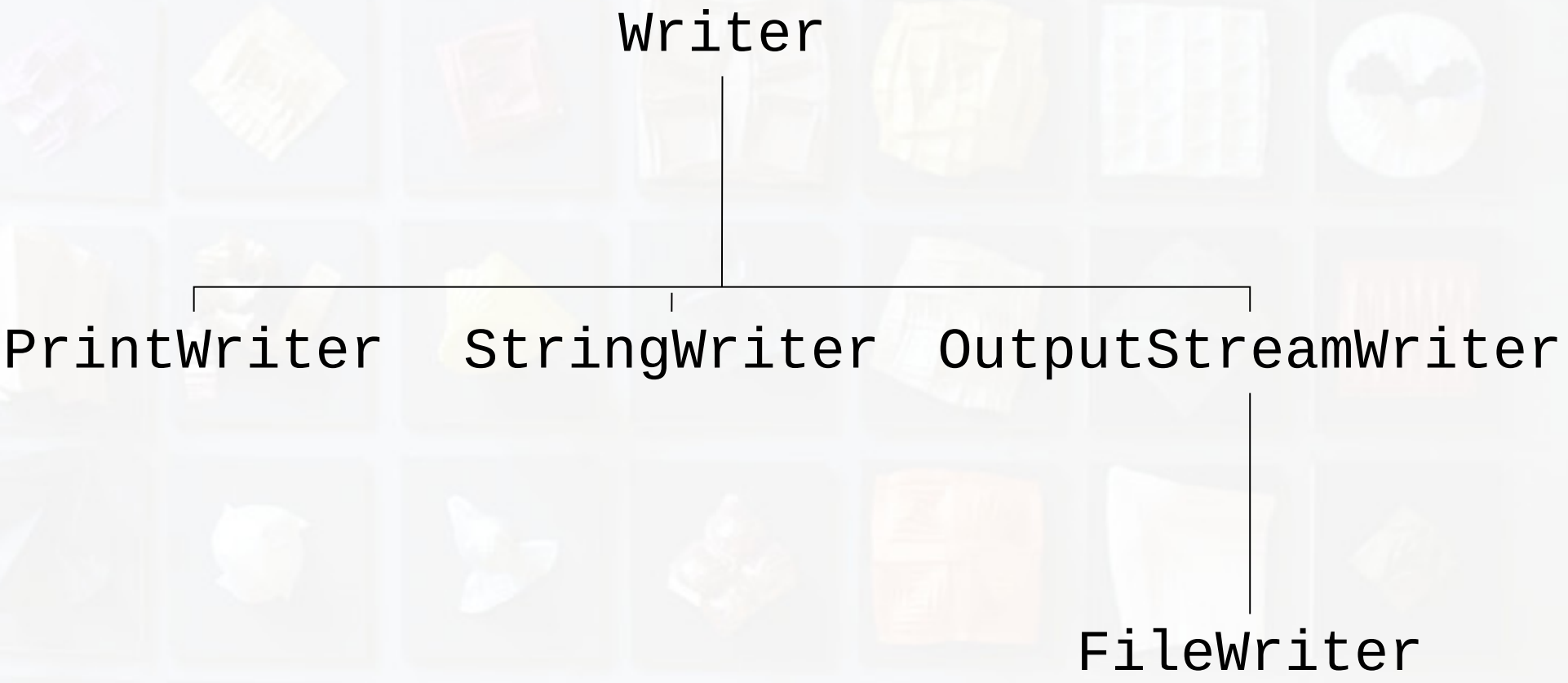
Streams

- Envio e recuperação de dados para/de fontes externas (arquivos, dados pela rede etc.)
- Streams representam fluxos de informação de entrada ou saída
- As Streams são representadas genericamente por duas classes abstratas:

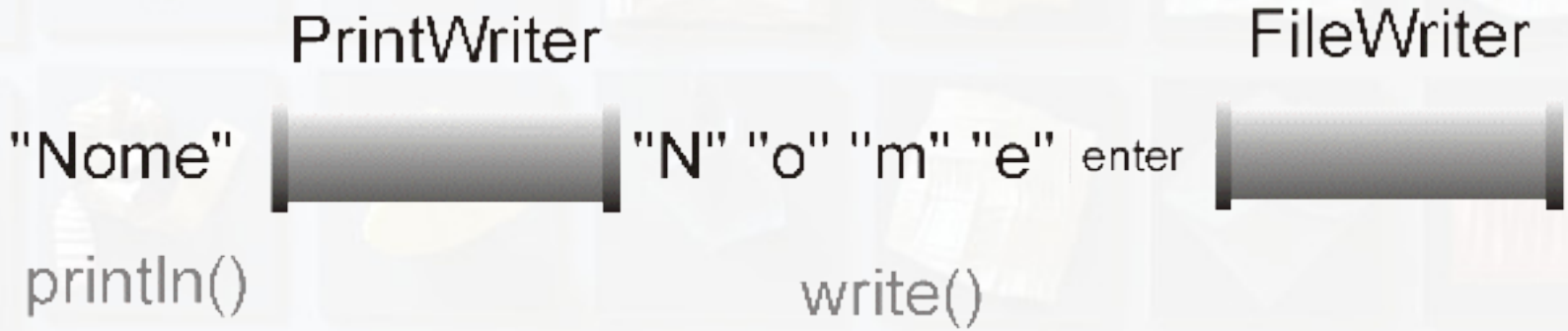
`Reader` - stream de entrada

`Writer` - stream de saída

Hierarquia de Streams `Writer`



Writer → Pipe & Filter



Pipe & Filter em Java

1. Crie o Writer final

2. Crie o Writer inicial ligado ao final

3. Ao chamar o método do Writer inicial ele automaticamente canalizará para o final

4. Feche o Writer no final

```
FileWriter arquivo;
arquivo = new FileWriter("saida2.txt");
```

```
PrintWriter formatado;
formatado = new PrintWriter(arquivo);
```

```
formatado.println("Tecodonte");
```

```
formatado.close();
```

Tarefa 4

- Ajuste a proposta da Tarefa 3 para que se encaixe em uma arquitetura Pipe & Filter.

Estilos Arquiteturais

Model-View-Controller (MVC)

Model-View-Controller

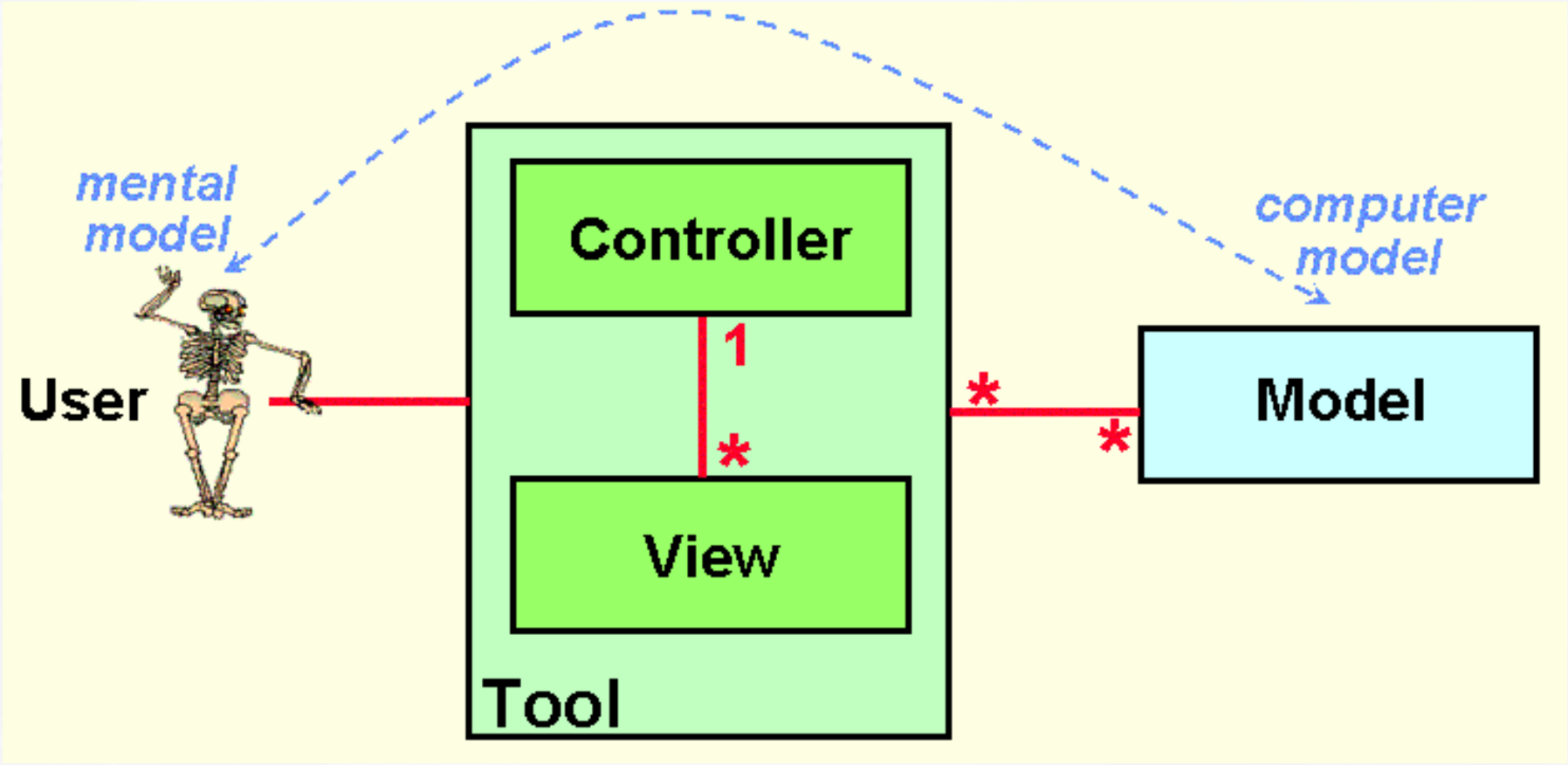
- Observações no contexto do Smalltalk demonstraram as vantagens de dividir uma aplicação em três partes:
  - modelo subjacente do domínio da aplicação
  - forma como o modelo é apresentado ao usuário
  - forma como o usuário interage com o modelo

(Krasner, 1988)

MVC
Model-View-Controller

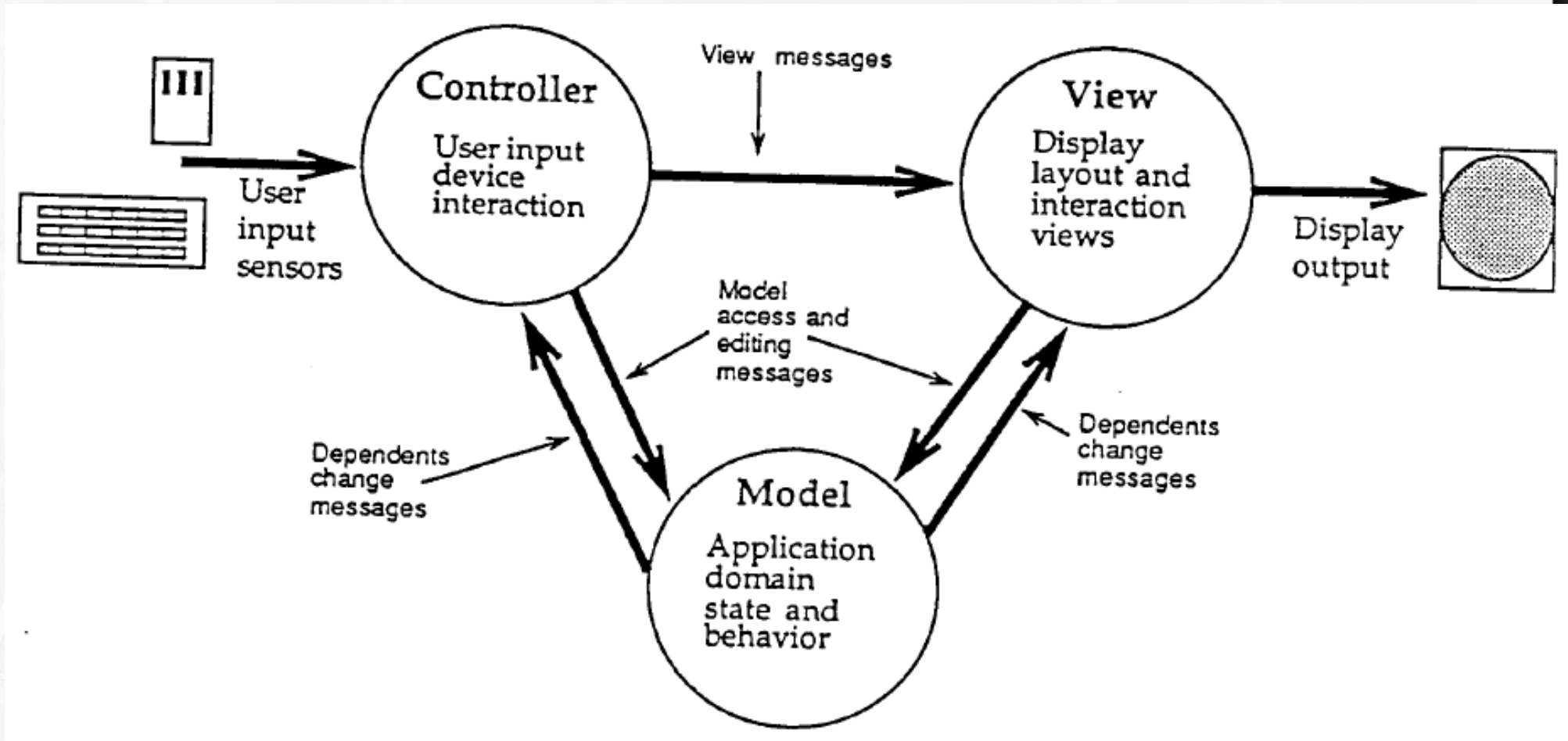
- Estilo arquitetural cujos componentes são divididos em três grupos:
  - *Model*: modelo subjacente da aplicação
    - representada como estruturas de dados ou de classes
  - *View*: lida com a parte de apresentação visual (gráfica)
  - *Controller*: interface entre *Model* e *View* e com os dispositivos de entrada

Model-View-Controller



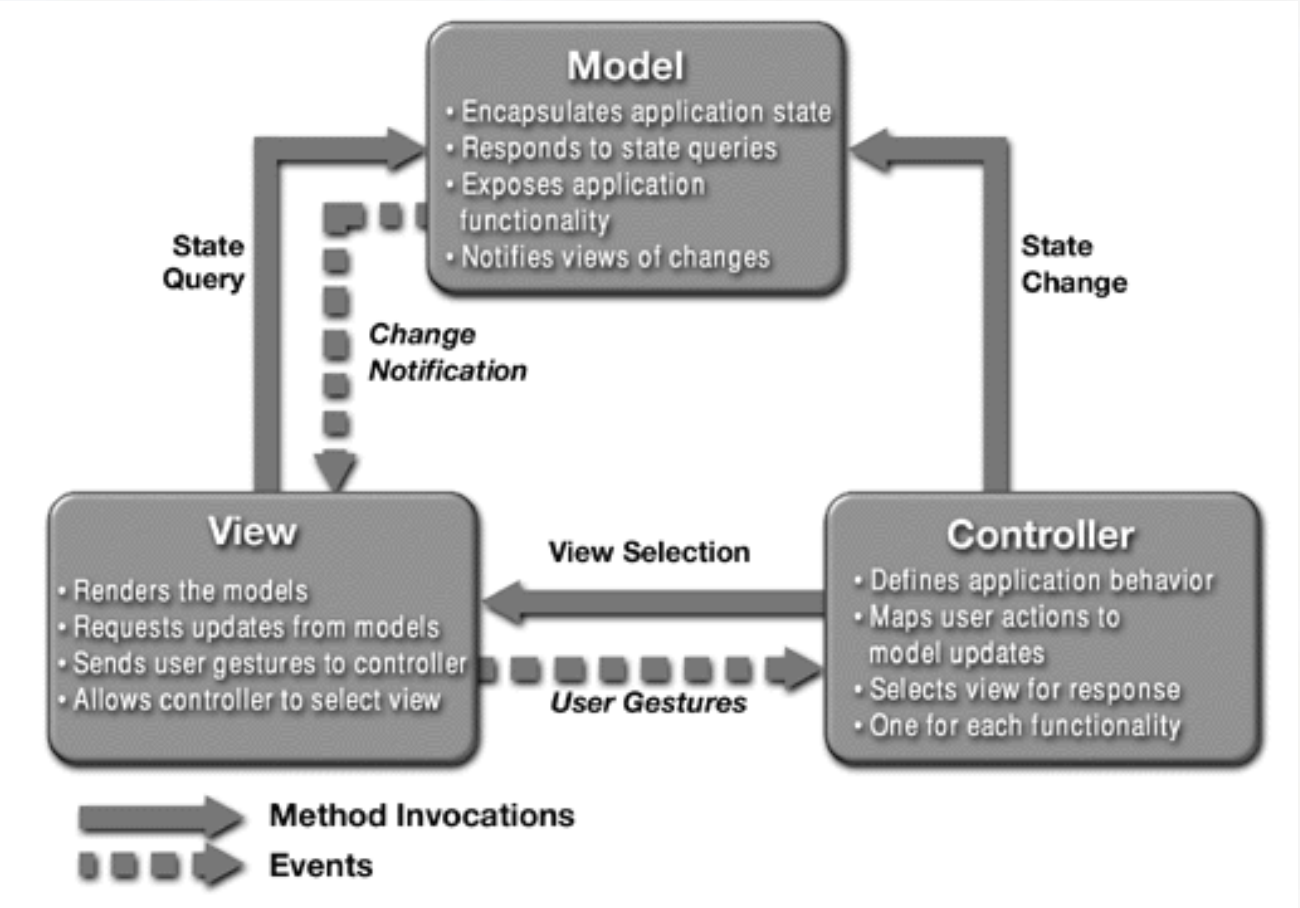
(?)

Model-View-Controller



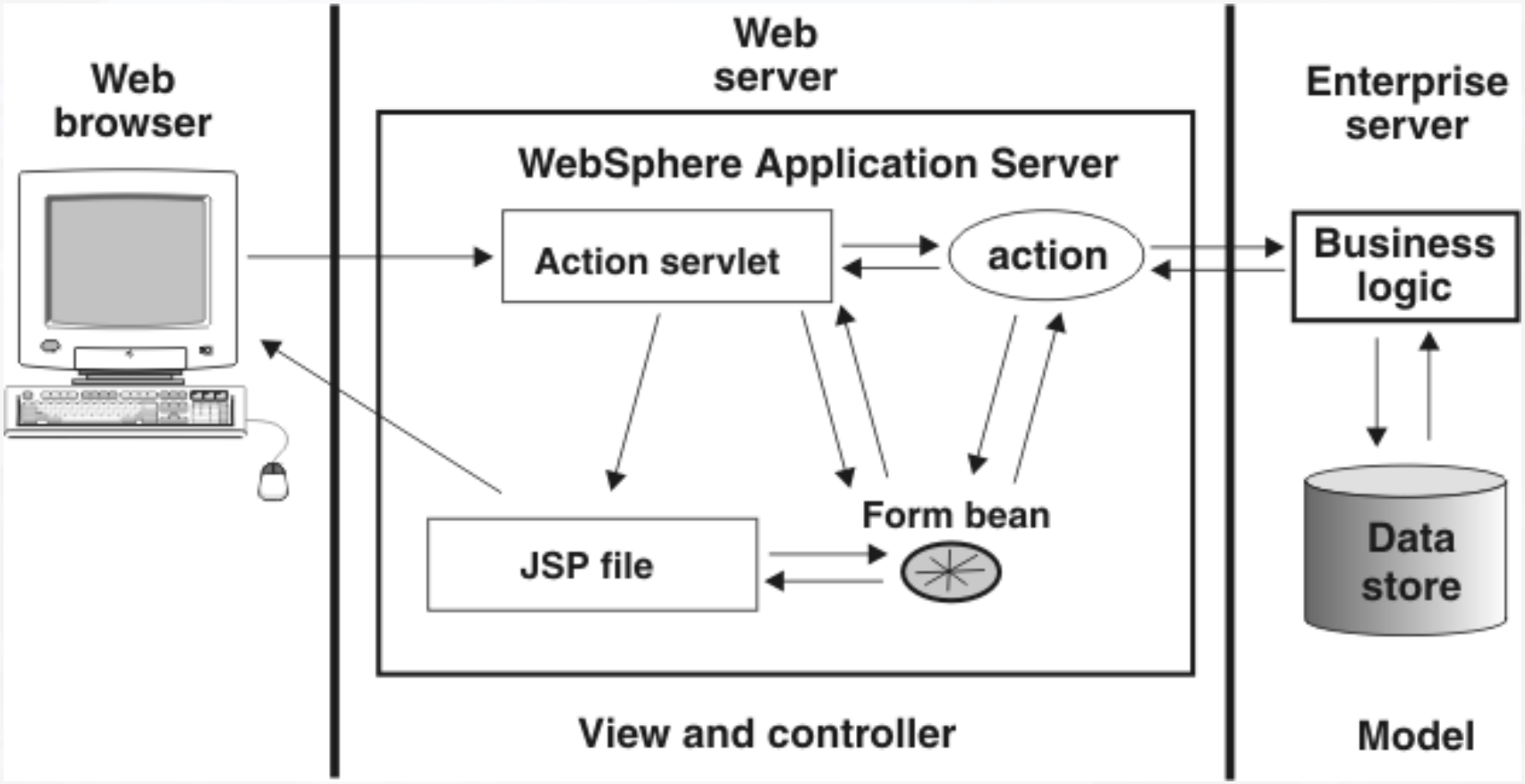
(Krasner, 1988)

MVC
J2EE MVC


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
Change Notification
State Change
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

(?)

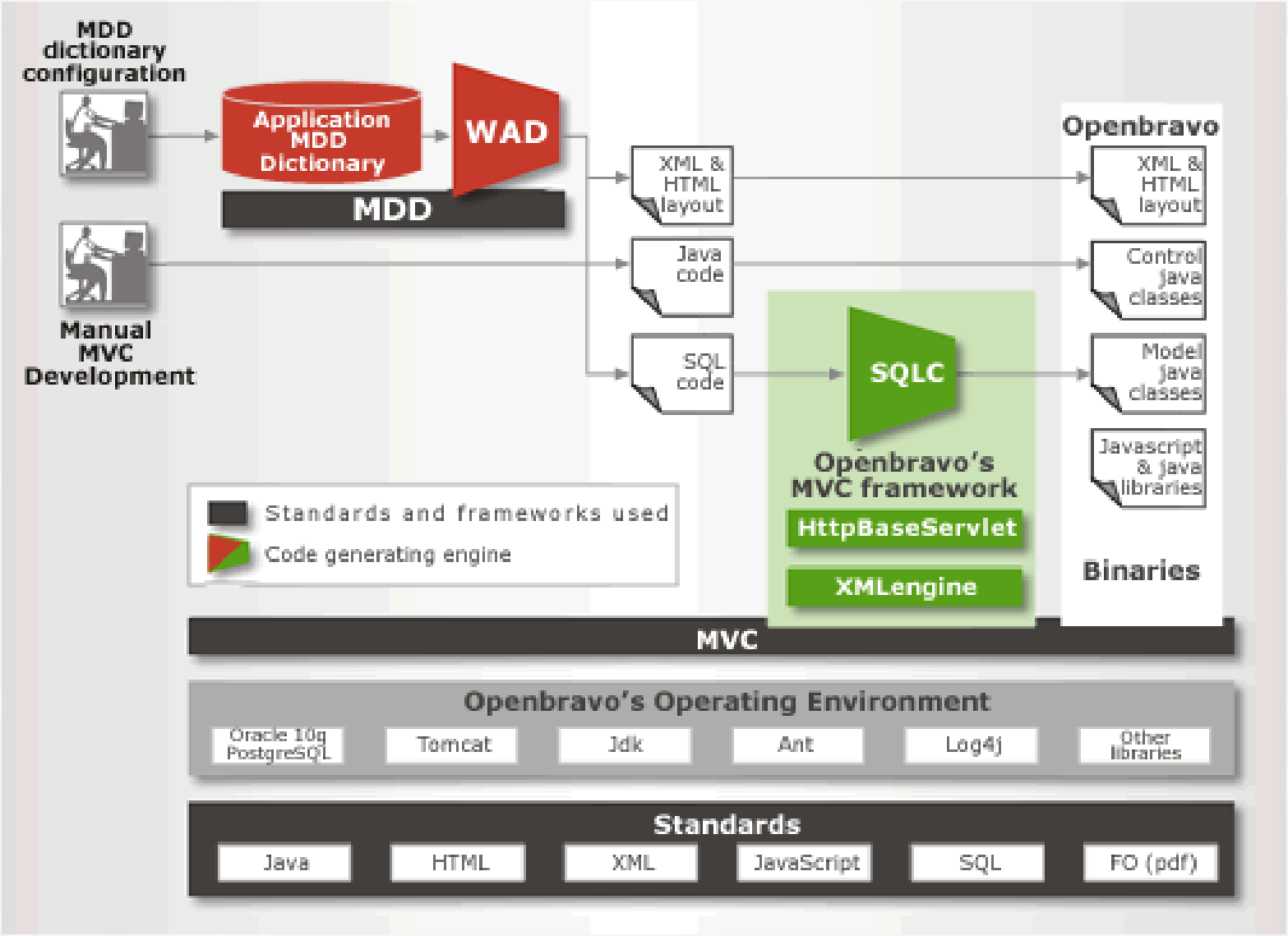
MVC
Apache Struts



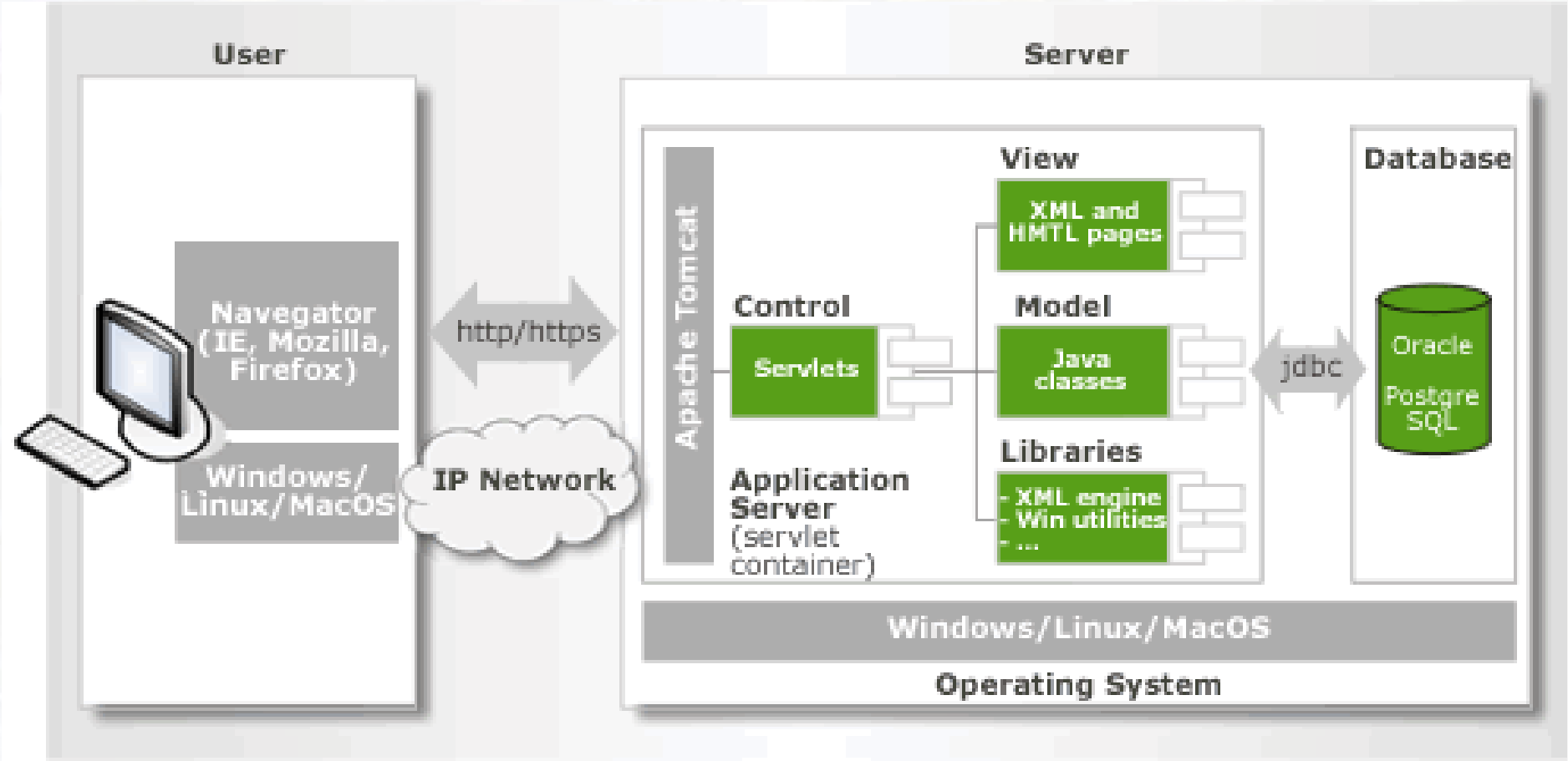
(?)

Arquitetura na Prática

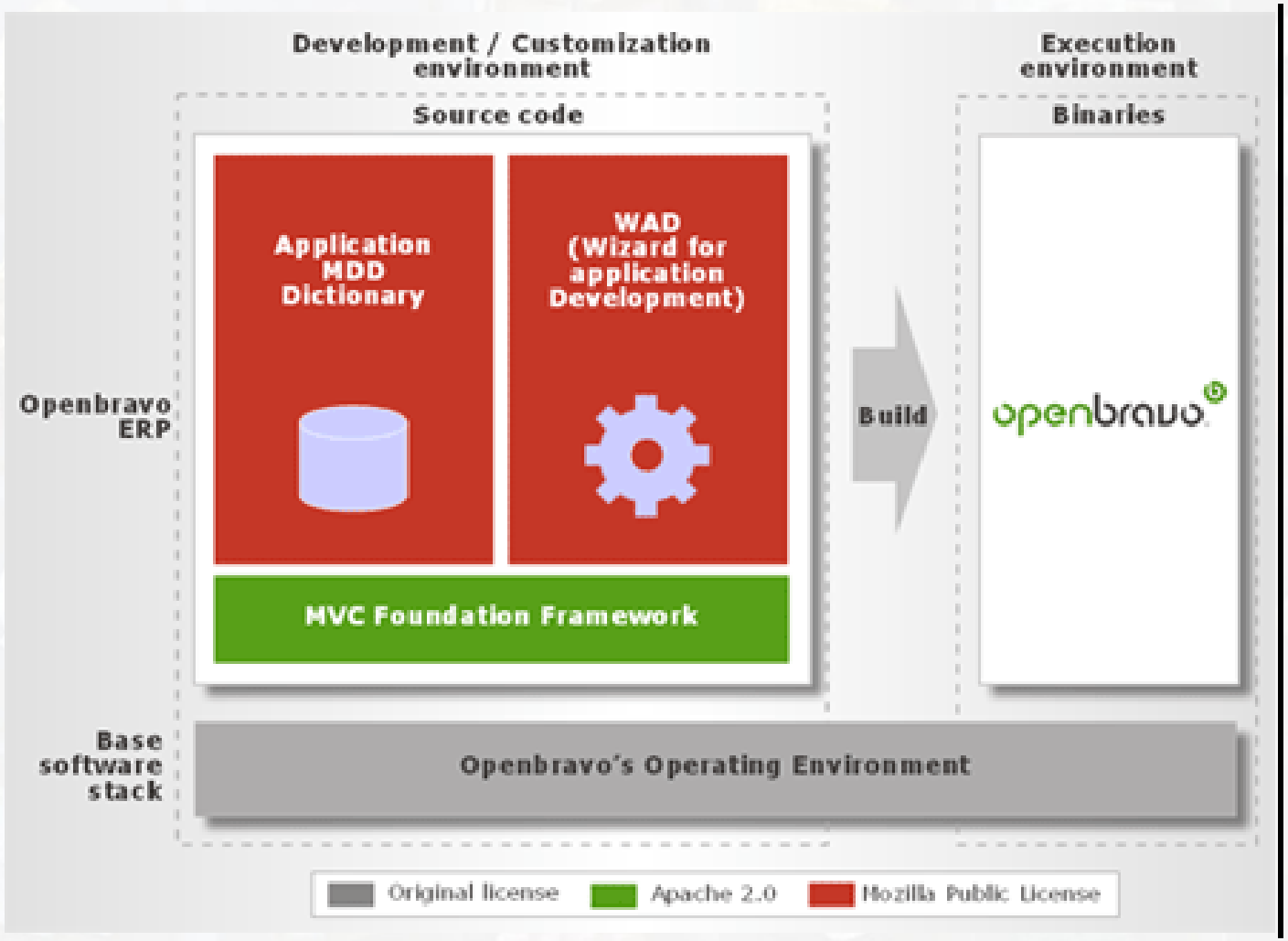
Openbravo
Arquitetura



Openbravo
Arquitetura



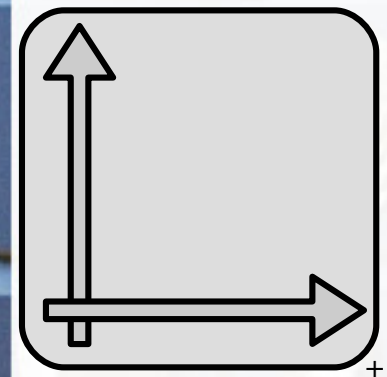
Openbravo
Arquitetura



Tarefa 5

- Organize os componentes da biblioteca conforme a arquitetura MVC.

Tarefa 3 (Componentes)

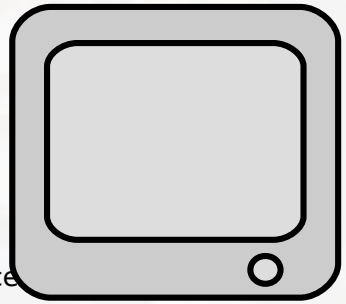


«component»
ChartComponent

IRun

IColumnProducer

+ title: String [1]



«component»
ConsoleComponent

IConsoleUpdate

IInstancesProducer

IColumnProducer

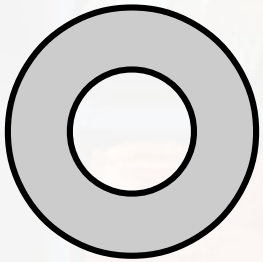
«component»
ProjectionComponent

IColumnProducer

IInstancesProducer

+ title: String [1]

+ attribute: String [1]



«component»
SelectionComponent

IInstancesProducer

IFilter

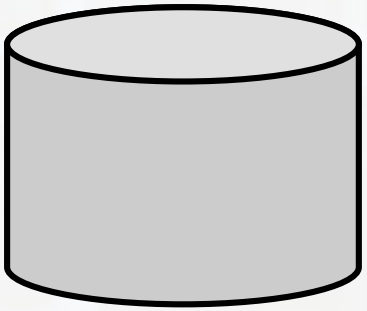
IInstancesProducer

+ title: String [1]

+ attributeB: String [1]

+ attributeA: String [1]

+ nominalComparison: Boolean [1]



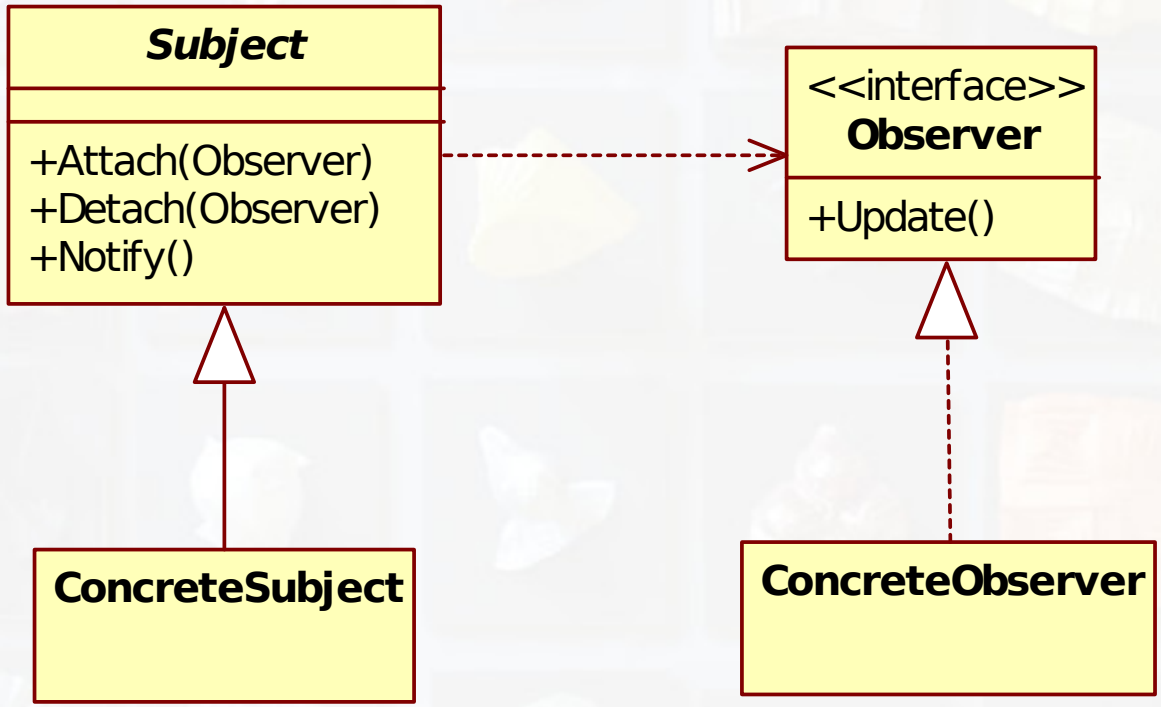
«component»
DataSetComponent

IInstancesProducer

+ dataSource: String [1]

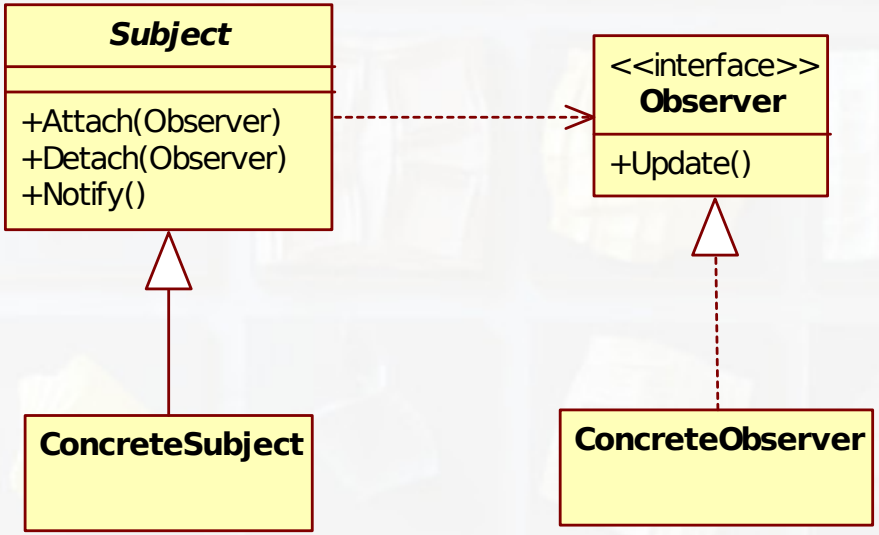
Pattern Observer

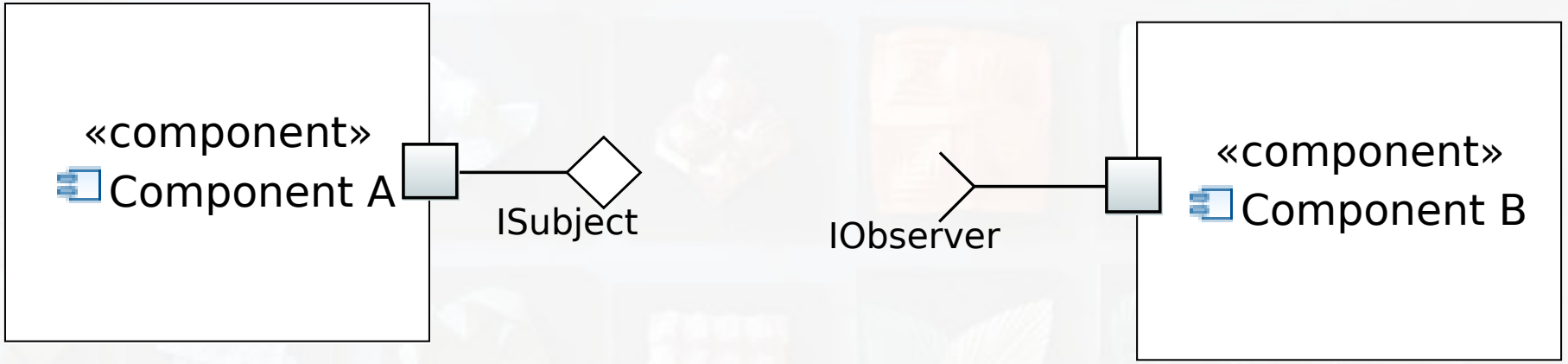
Eventos
Pattern *Observer*



Eventos (notação CORBA Component Model)

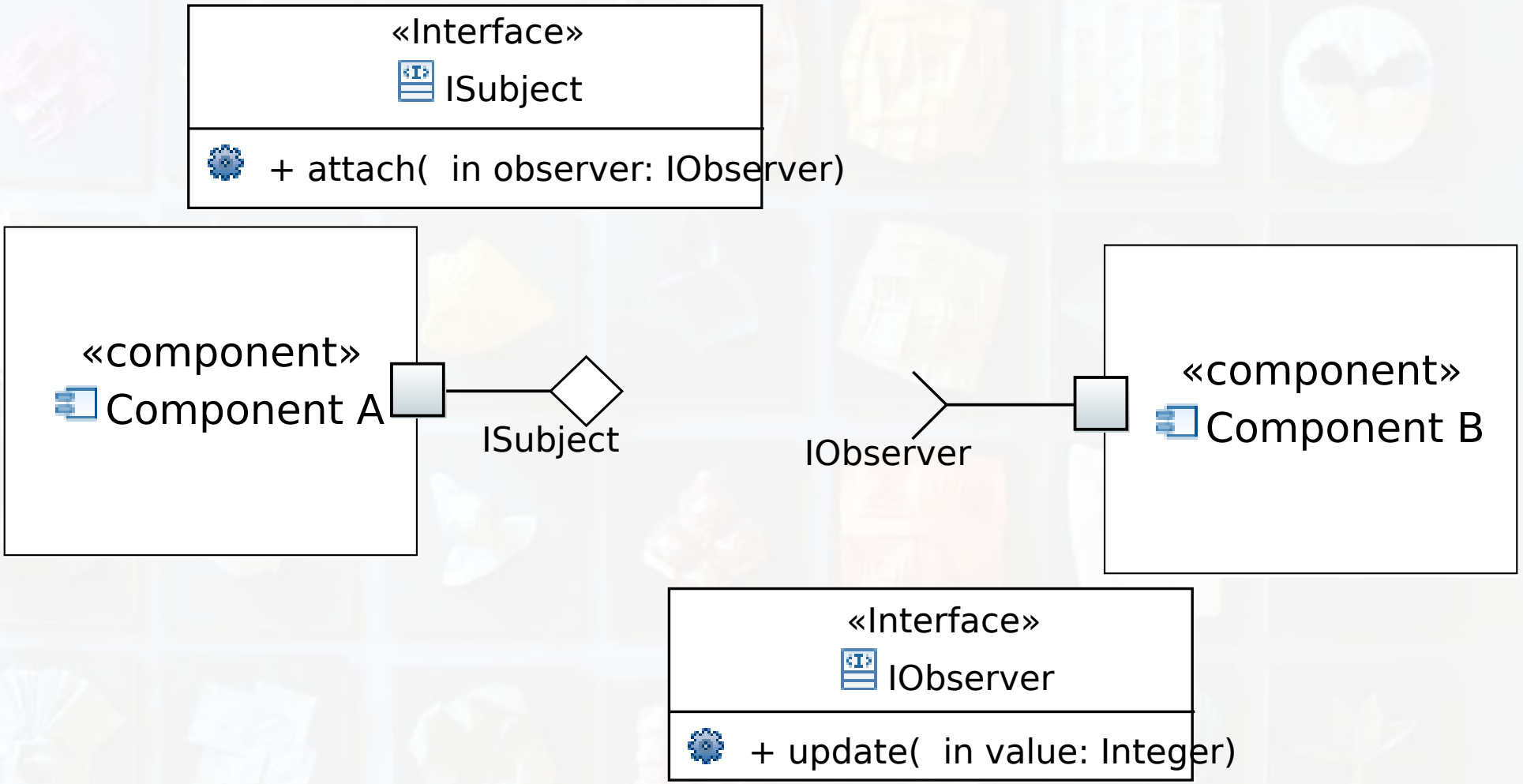
Pattern *Observer*





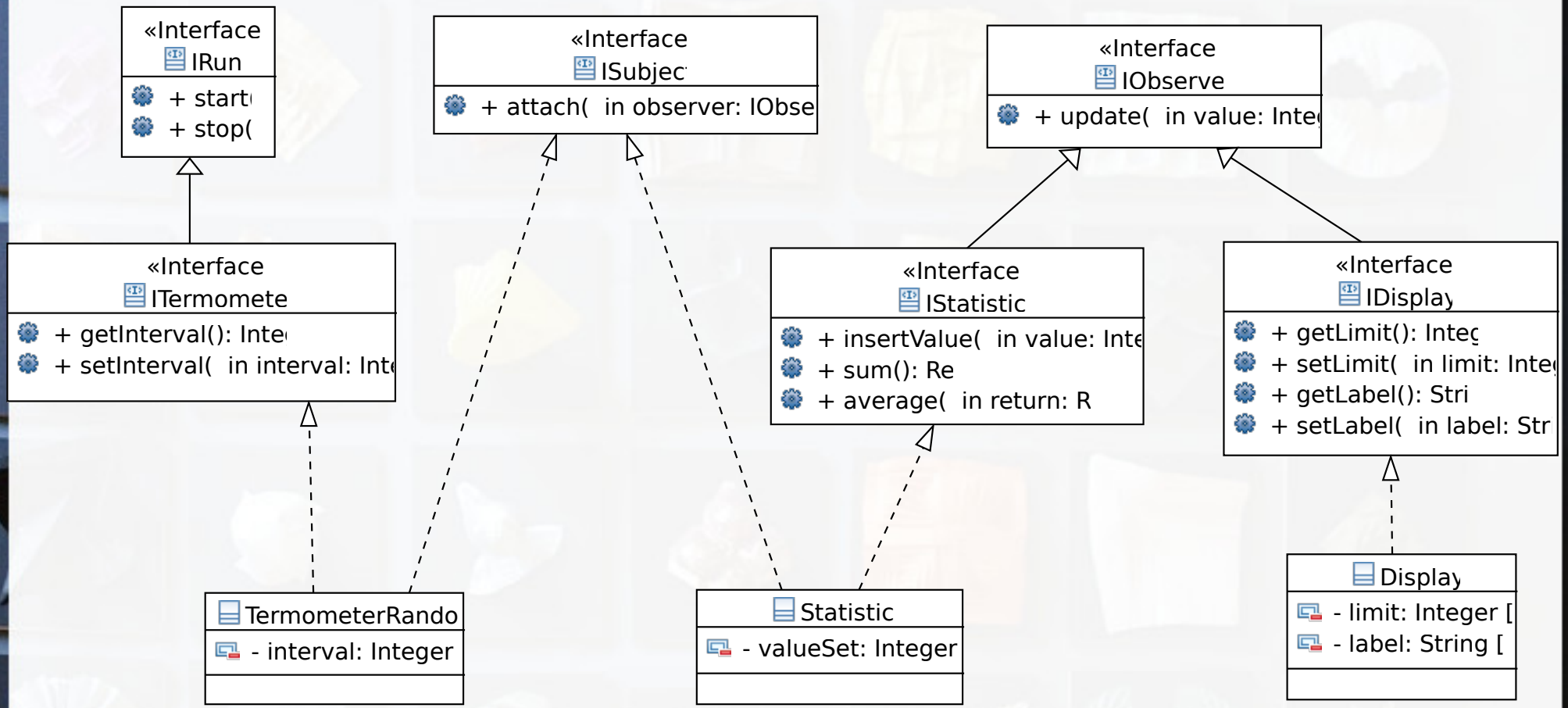
Eventos (notação CORBA Component Model)

Pattern *Observer*

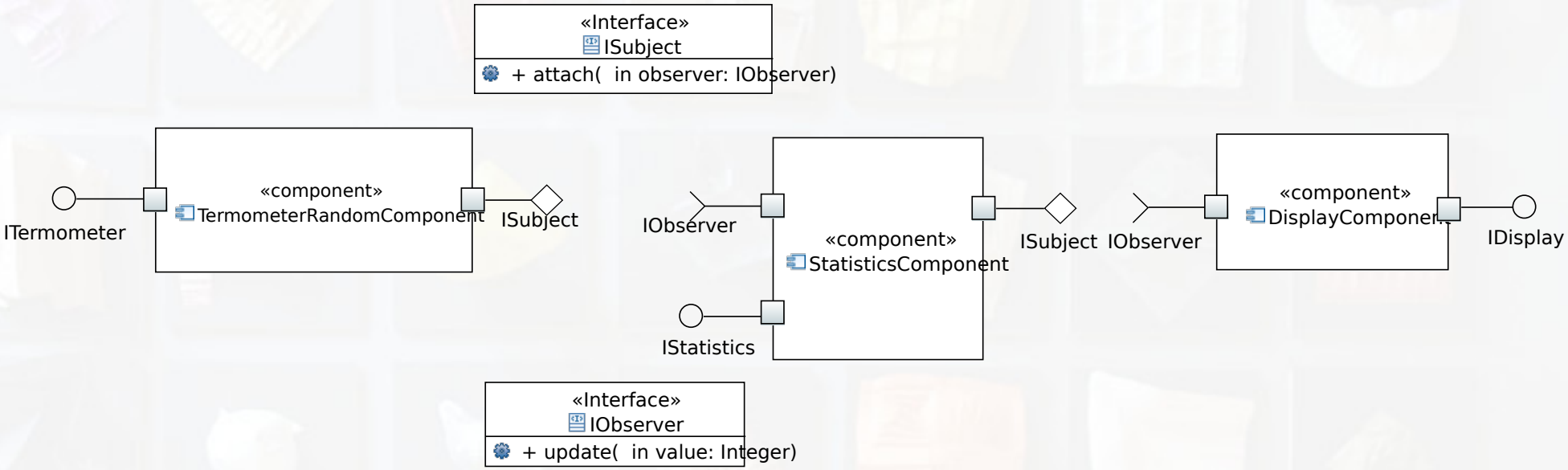


O Caso da Indústria

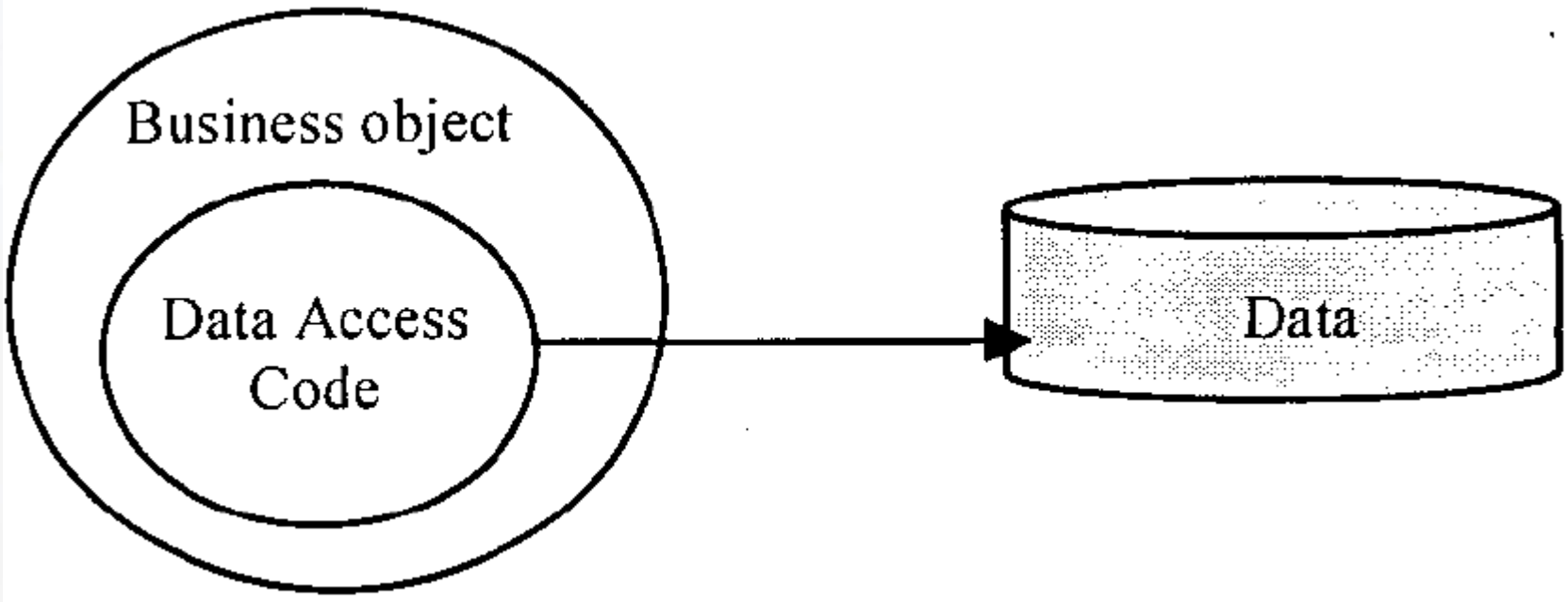
Caso da Indústria
Observer



Caso da Indústria
Observer



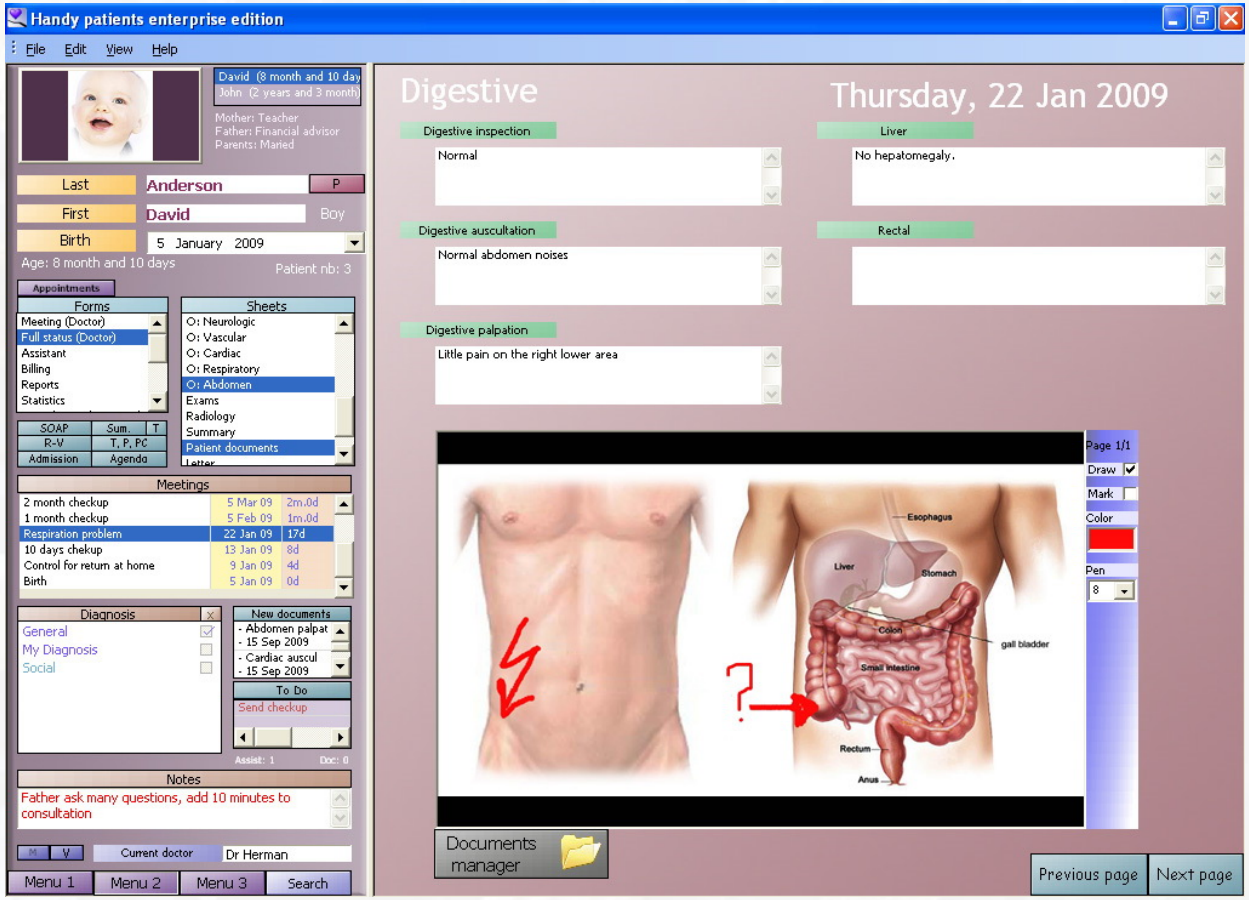
Data Access Objet (DAO) Pattern



(Matic, 2004)

Retomando o Desafio

- Montar componentes para suporte à decisão em um Prontuário Eletrônico.



By Oguntoye patients electronic medical record (free open source version), GPL, https://commons.wikimedia.org/w/index.php?curid=8894074

Organizando o Projeto da Estrutura

- Componentes de Negócio
- Componentes Técnicos

Componentes de Negócio
Lei de Conway

- “[...] uma organização que projeta sistemas [...] são levadas a produzir projetos que são cópias da estrutura de comunicação dessas organizações.” (Conway, 1968)

“[...] organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.” (Conway, 1968)

Fluxo de um Diagnóstico
Tarefa 1

- Considere que um paciente aparece em um hospital com sintomas.
- Projete o fluxo de atividades simples (em três passos, sem exames laboratoriais e intervenções) para emissão desse diagnóstico.
- Considere que não há sistema computacional.

Fluxo de um Diagnóstico
Tarefa 1

Paciente

sintomas

Médico

Fluxo de um Diagnóstico
Tarefa 1

sintomas

diagnóstico

Médico

Paciente

Fluxo de um Diagnóstico

Tarefa 1

Paciente

sintomas

diagnóstico

Médico

dianóstico

diagnóstico

Cirurgião

Enfermeira

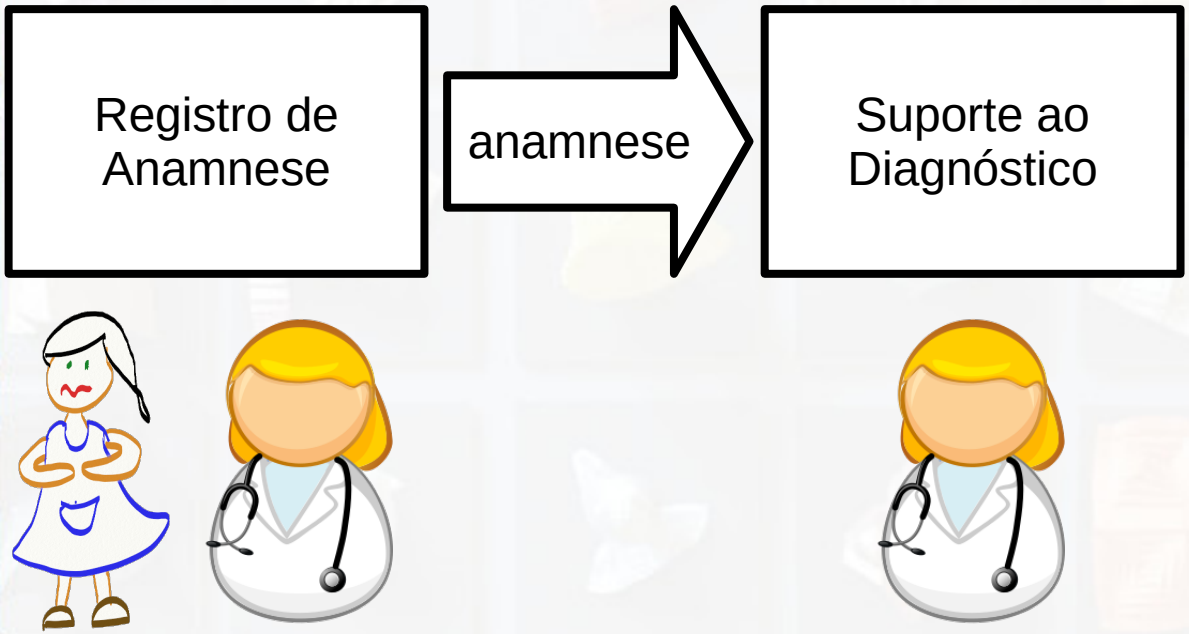
Fluxo de um Diagnóstico
Tarefa 2

- Considerando que você está projetando um sistema computacional para o fluxo da Tarefa 1 com suporte computacional ao diagnóstico.
- Desenhe um diagrama com os módulos envolvidos e como eles se comunicam.

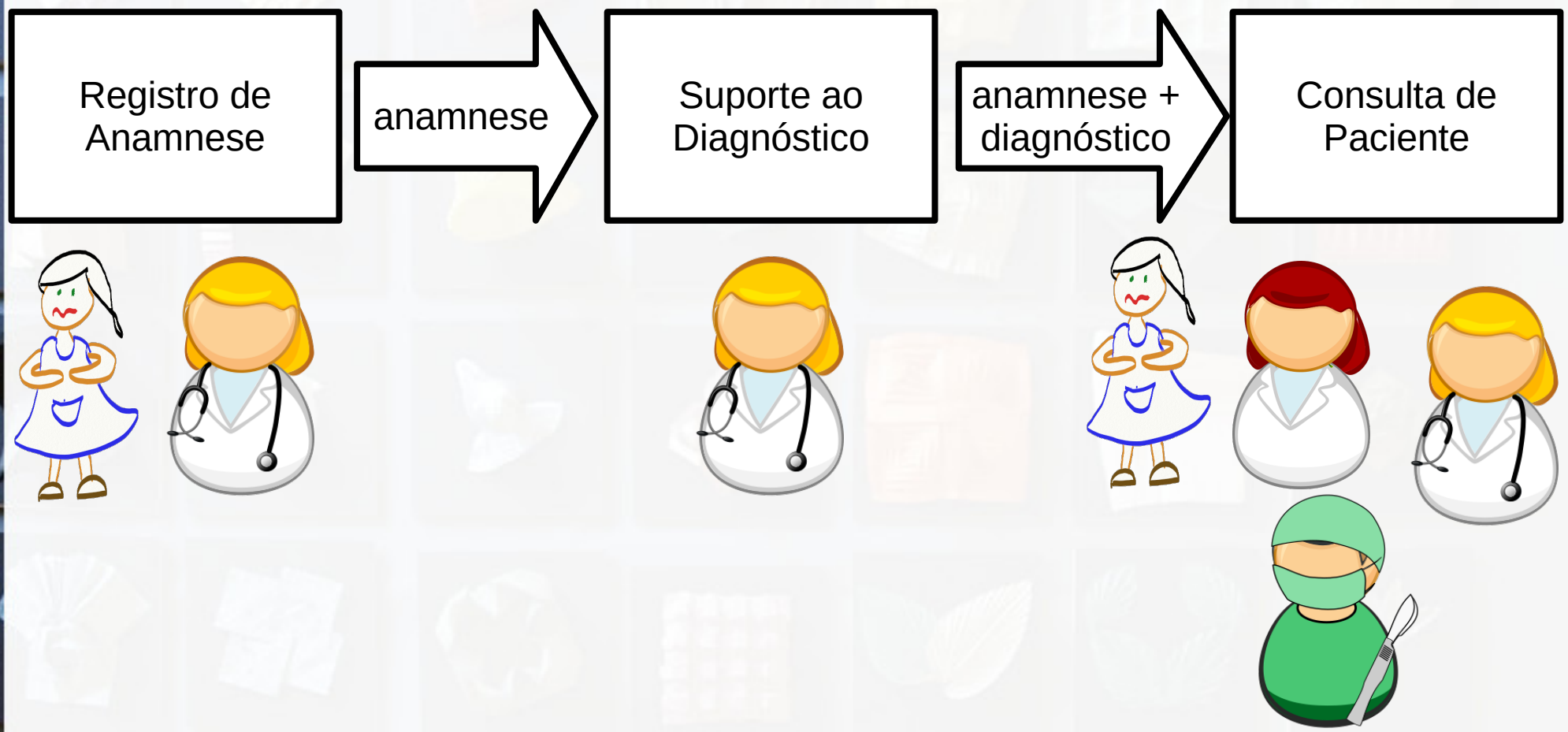
Fluxo de um Diagnóstico
Tarefa 2

Registro de Anamnese

Fluxo de um Diagnóstico
Tarefa 2



Fluxo de um Diagnóstico
Tarefa 2



Modelando Componentes como Serviços

Estilos Arquiteturais

Orientado a Serviços

Arquitetura Orientada a Serviços

- Arquitetura Orientada a Serviços – *Service Oriented Architecture* (SOA)
- Computação Orientada a Serviços
  - “Computação orientada a serviços é o paradigma da computação que utiliza serviços como elementos fundamentais para o desenvolvimento de aplicações.” (Papazoglou, 2003)

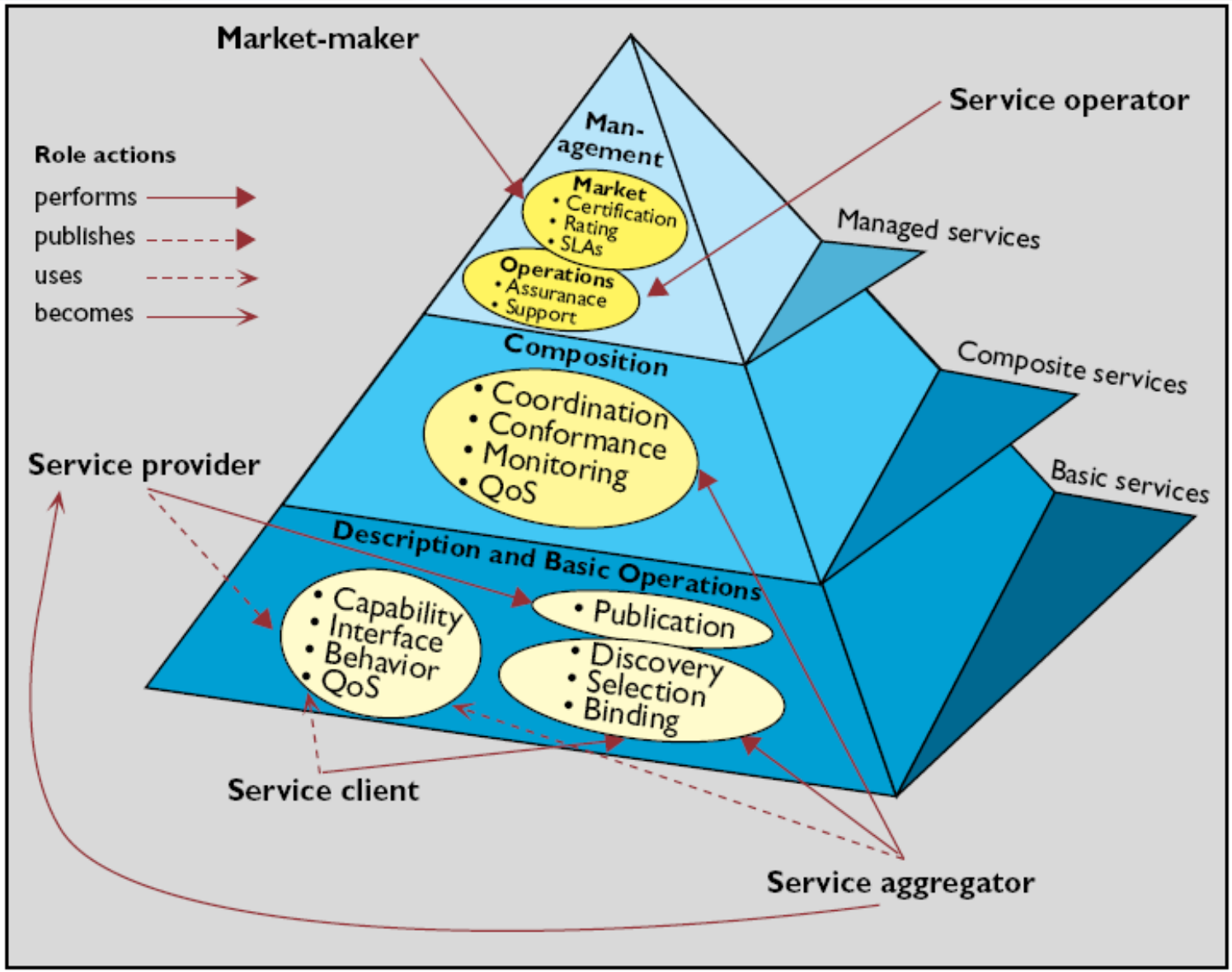
Service Oriented Architecture (SOA)

- Componentes auto-descritivos
- Abertos
- Possibilitam composição rápida e a baixo custo
- São providos por provedores de serviços

(Papazoglou, 2003)

- “SOA é um estilo arquitetural cujo objetivo é alcançar baixo acoplamento entre agentes de software em interação.” (He, 2003)

Service Oriented Architecture (SOA)



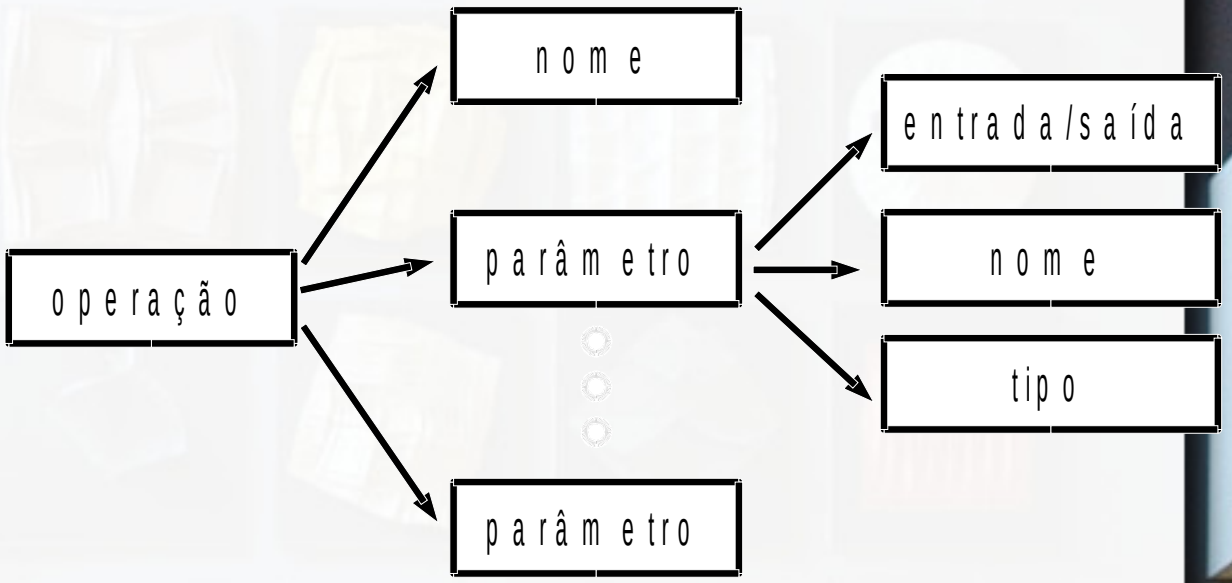
(Papazoglou, 2003)


O que diferencia um
Componente de um Serviço?

Componente como Serviço

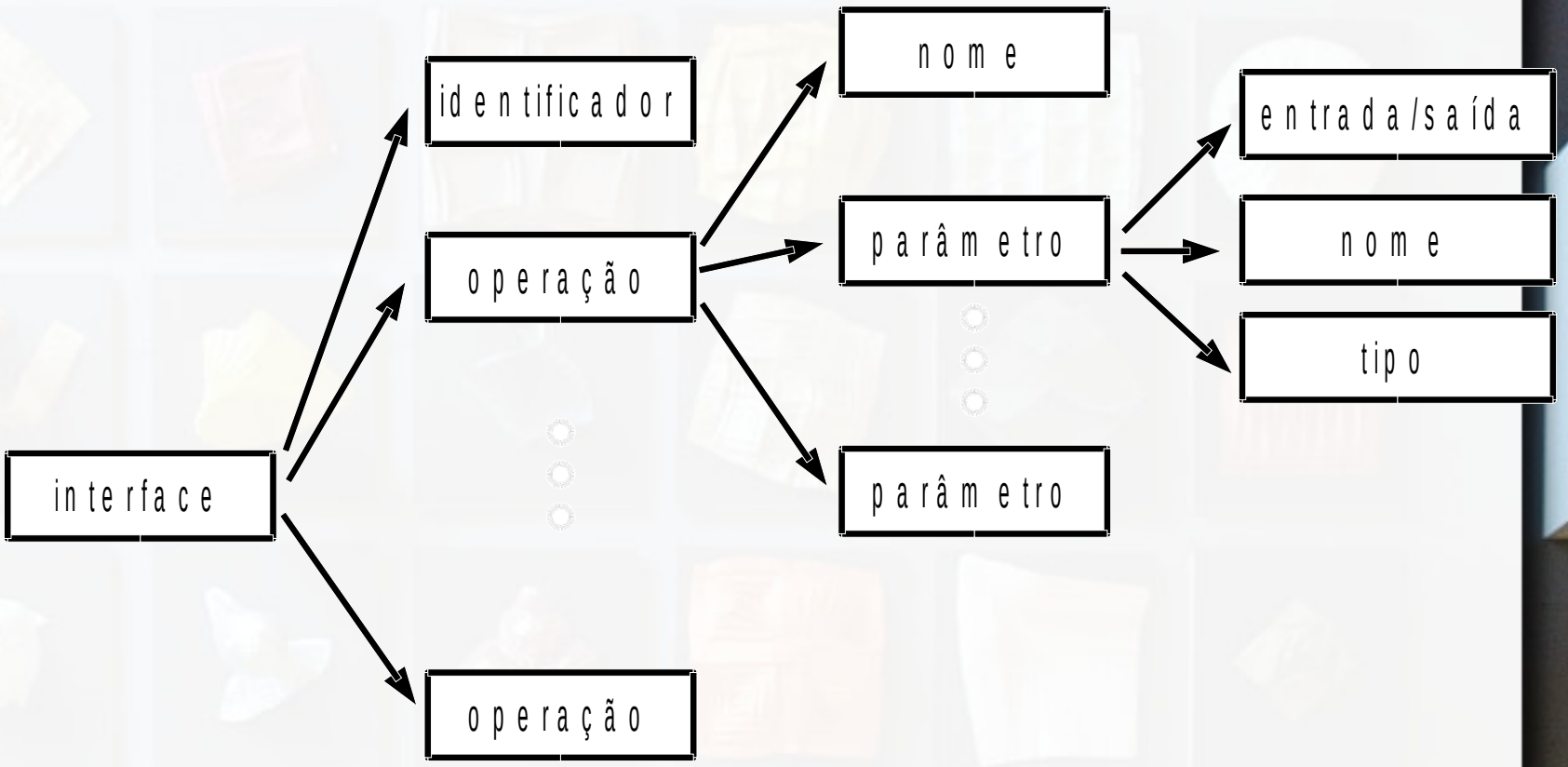
Componente como Serviço

Operações



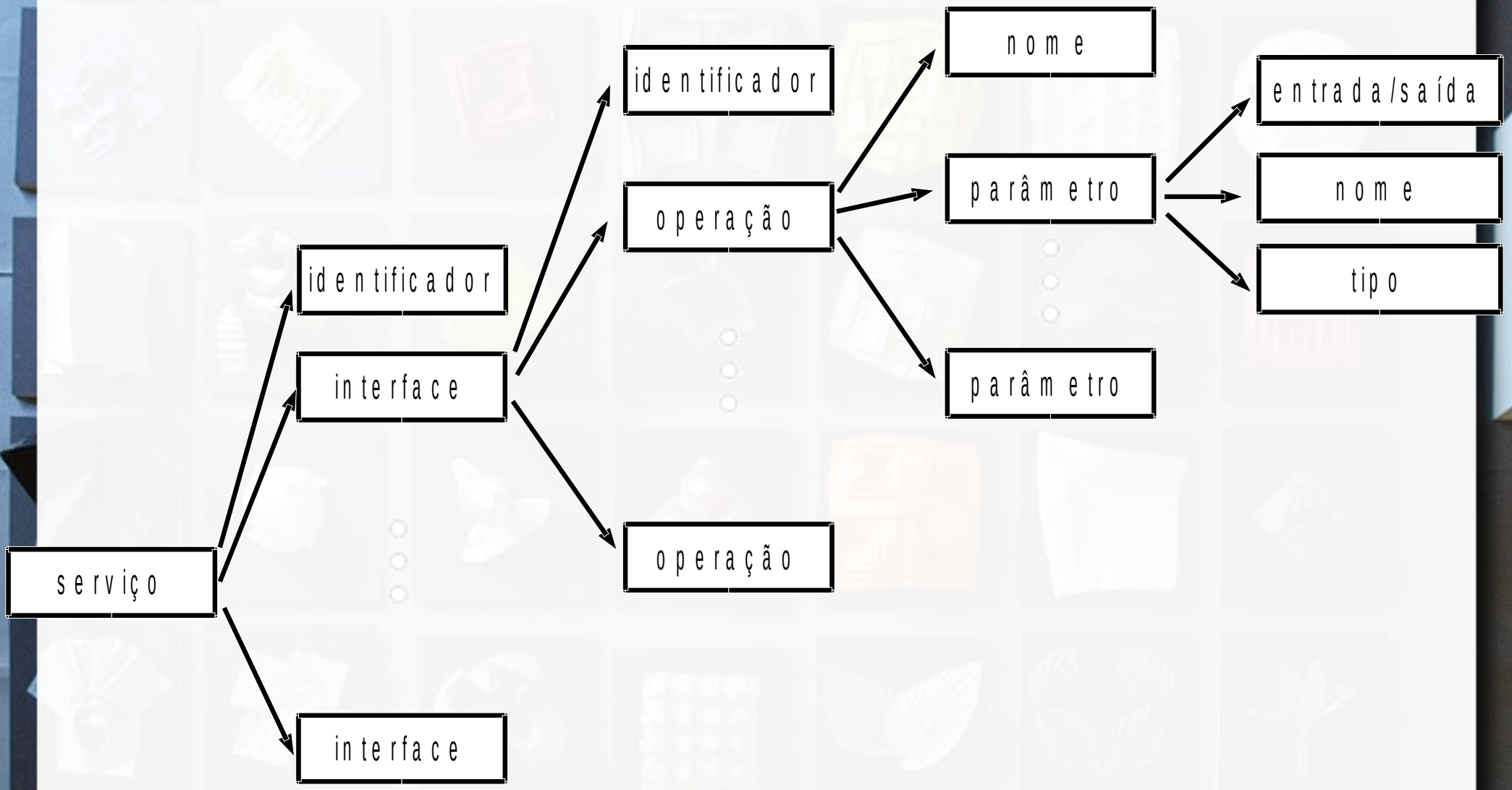
Componente como Serviço

Interface

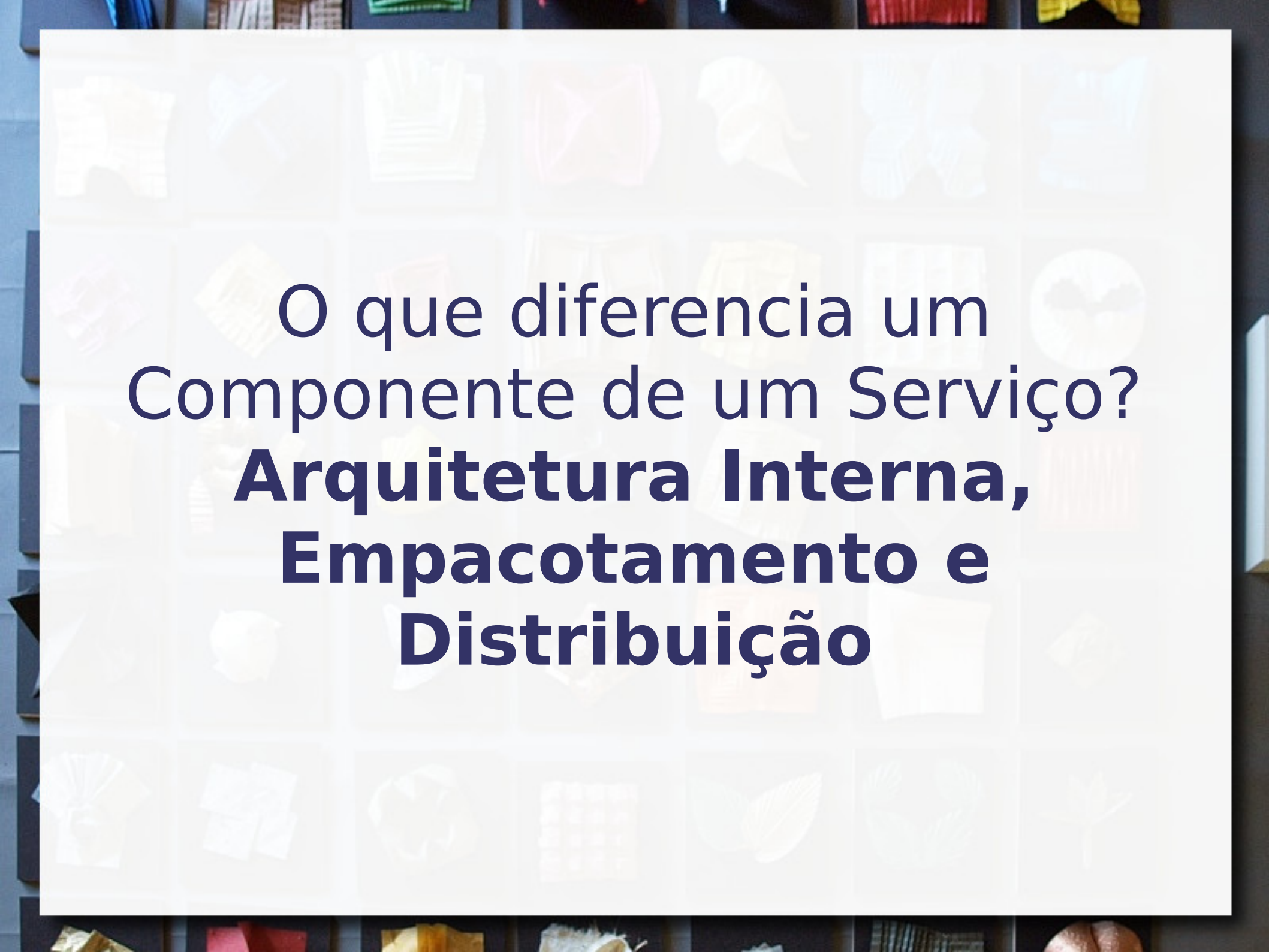


Componente como Serviço

Serviço



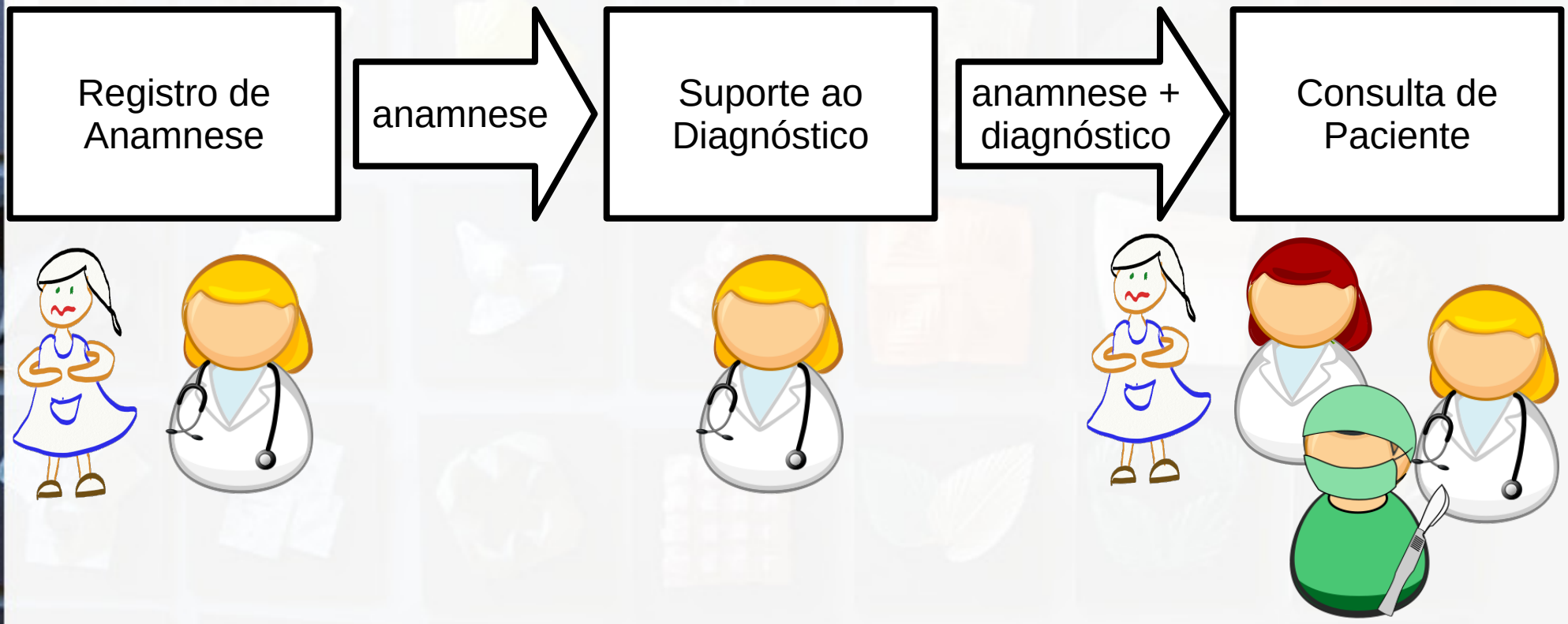

O que diferencia um
Componente de um Serviço?


O que diferencia um
Componente de um Serviço?
Arquitetura Interna,
Empacotamento e
Distribuição

Componente de Suporte ao Diagnóstico

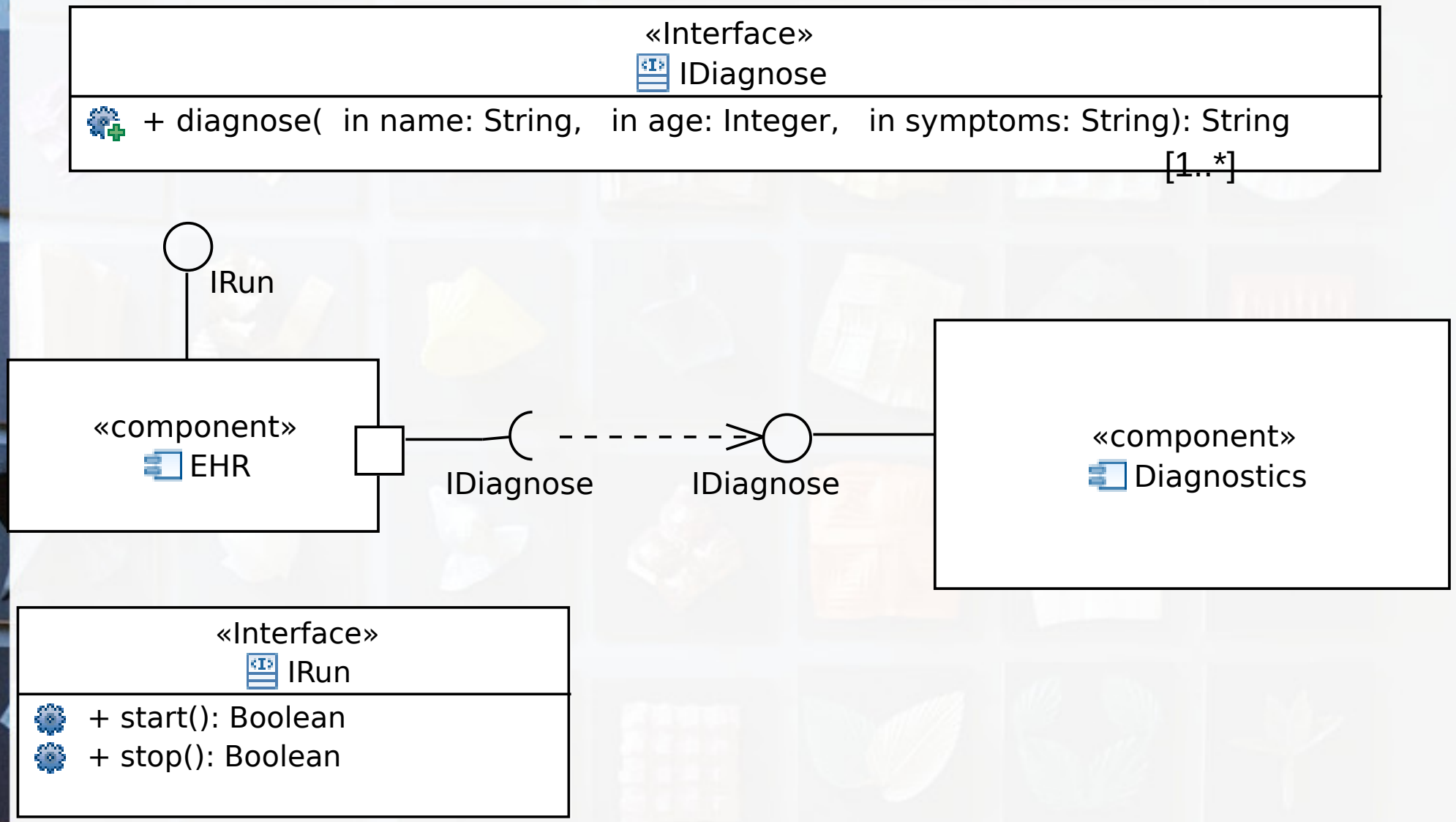
Tarefa 3

- Considerando que o módulo de Suporte ao Diagnóstico é um componente e um serviço, projete a sua interface.



Abordagem 1 – Grande Serviço

Tarefa 3



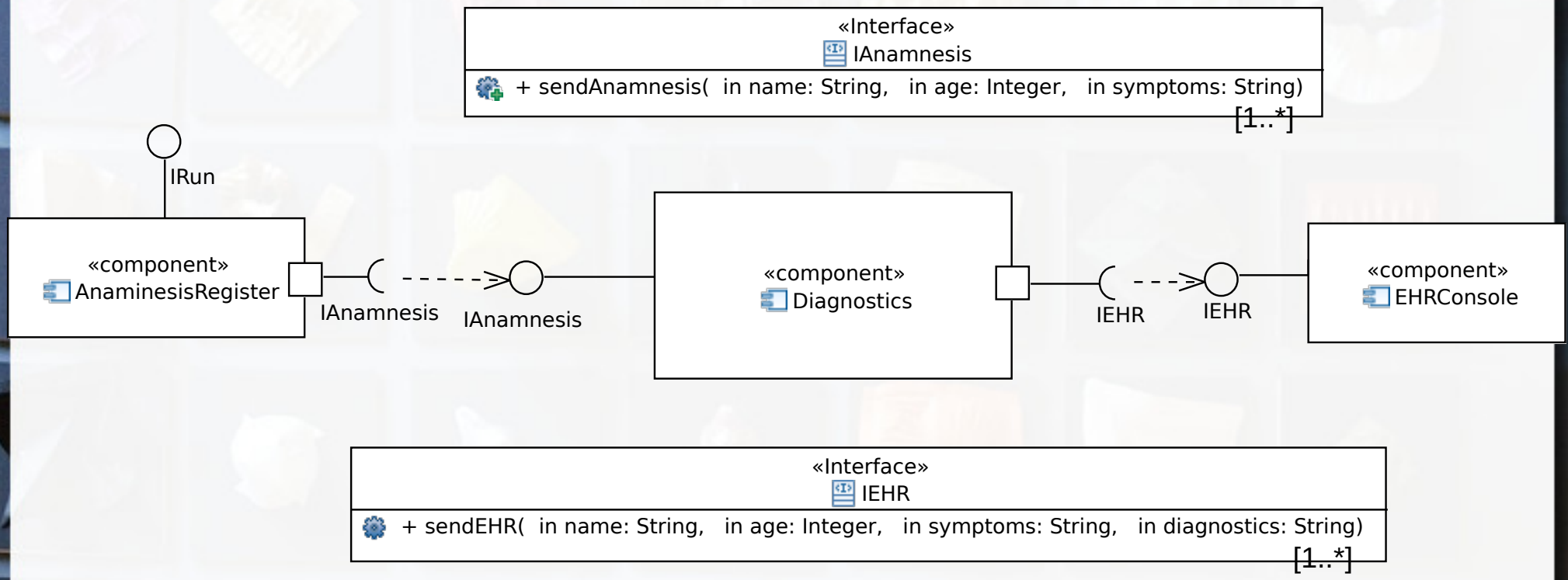

Possível projetar serviços menores?

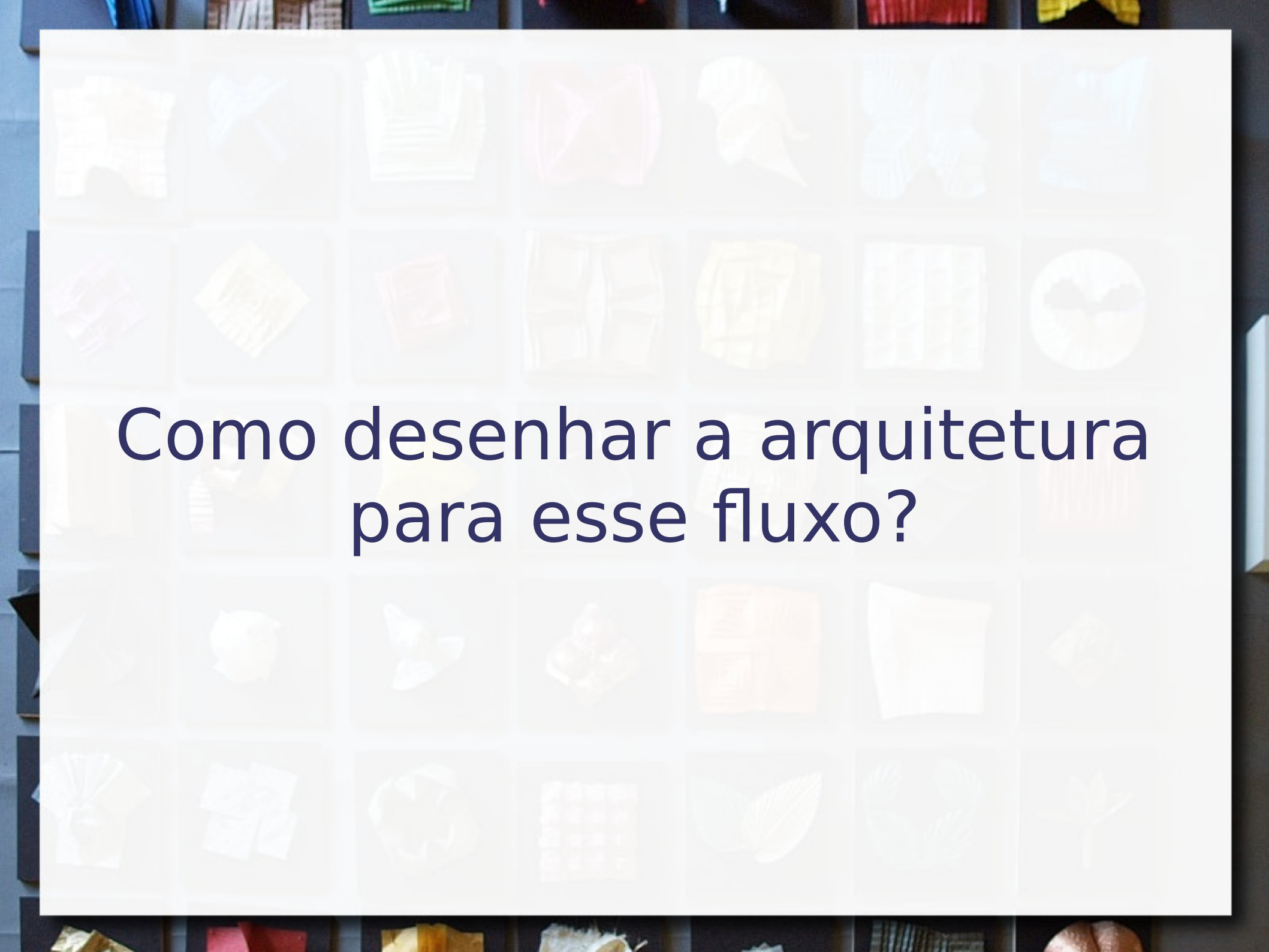
Regra Importante

- Na descrição do serviço de um componente evitar:
  - Componente faz serviço 1 e serviço 2
- Nos casos em que os serviços sejam independentes

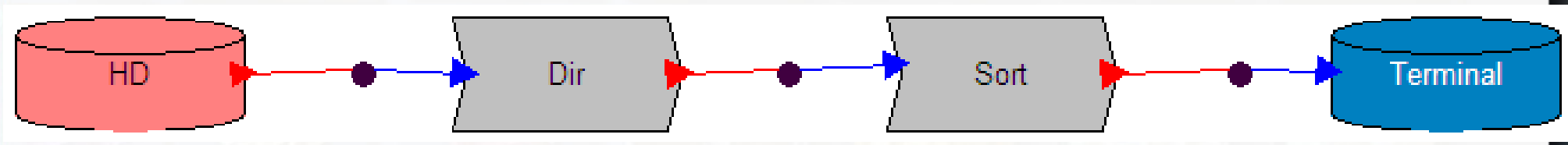
Abordagem 2 – Serviços Menores

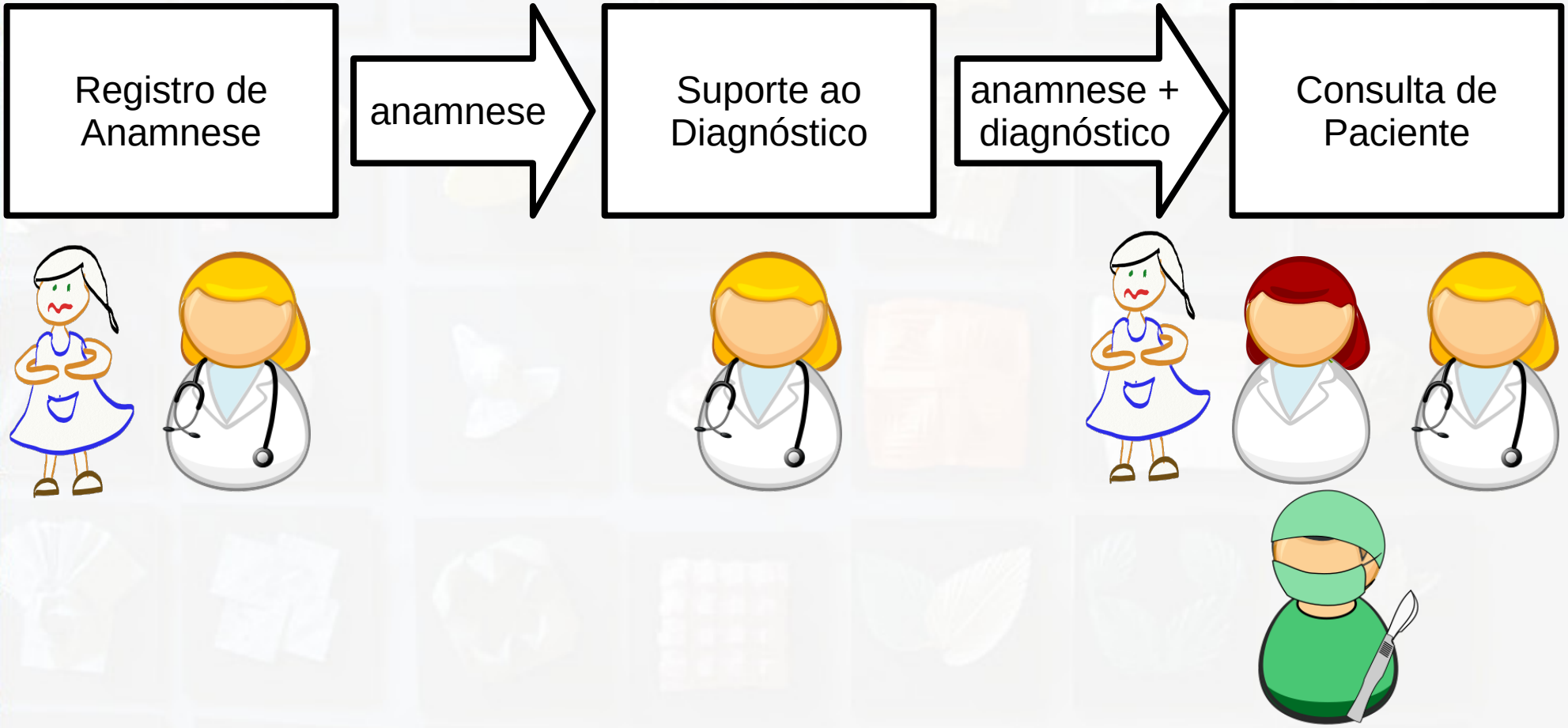
Tarefa 3




Como desenhar a arquitetura
para esse fluxo?

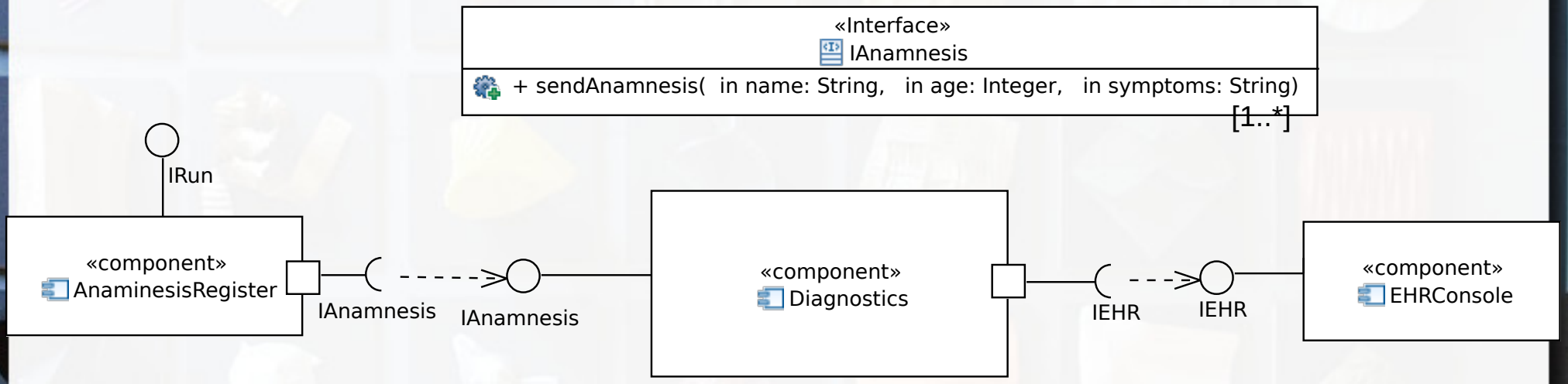
Fluxo de um Diagnóstico
Pipe & Filter?

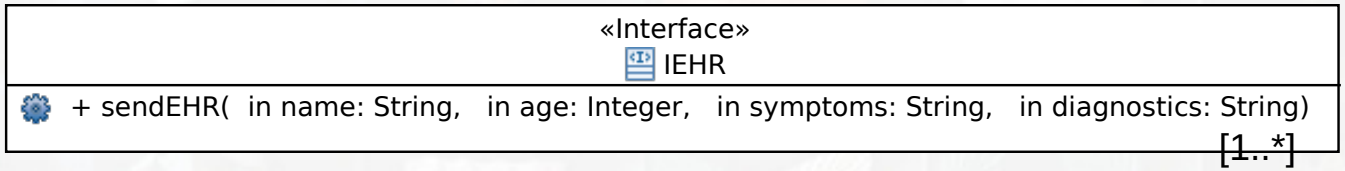


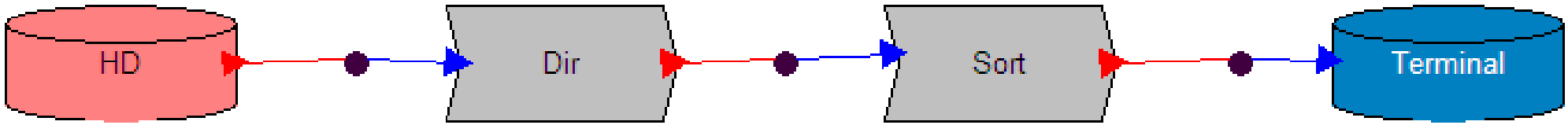


Pipe & Filter
Tarefa 4

- Modifique os componentes considerando um fluxo Pipe & Filter.

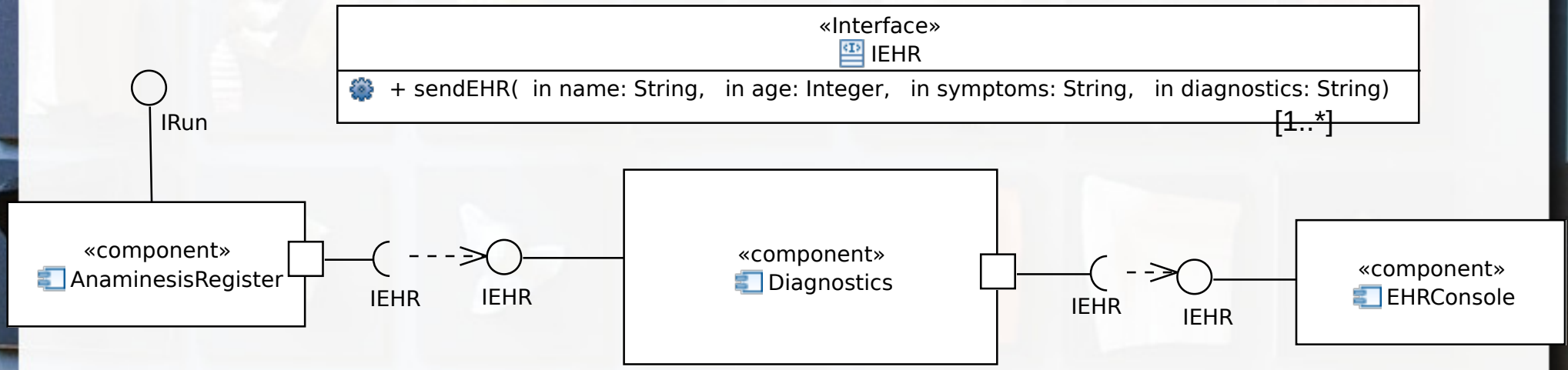






Pipe & Filter
Tarefa 4

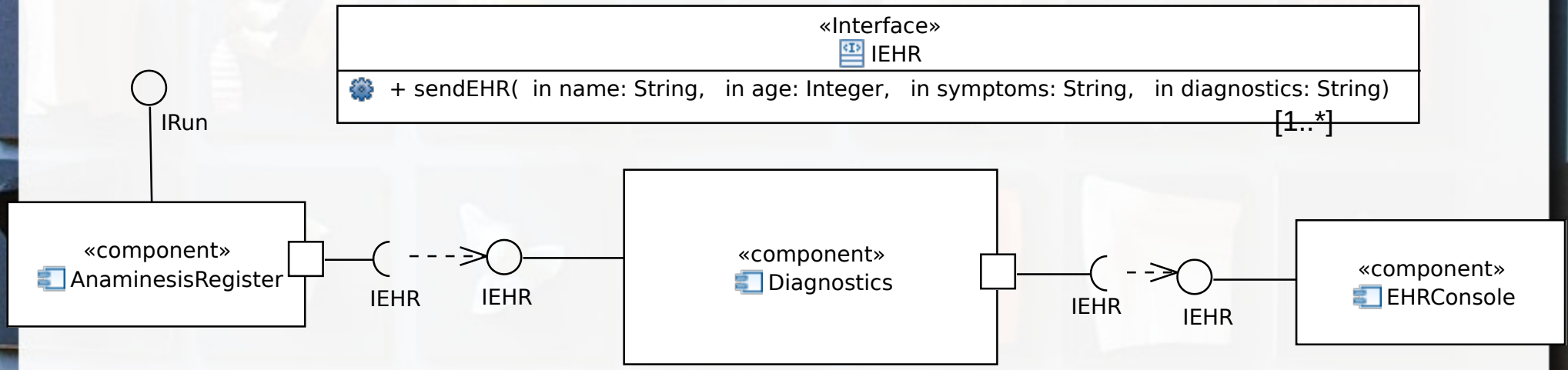
- Modifique os componentes considerando um fluxo Pipe & Filter.



Reduzindo a complexidade da Interface

Reduzindo a Complexidade
Tarefa 5

- Reduza a complexidade da Interface, que tem muitos parâmetros.

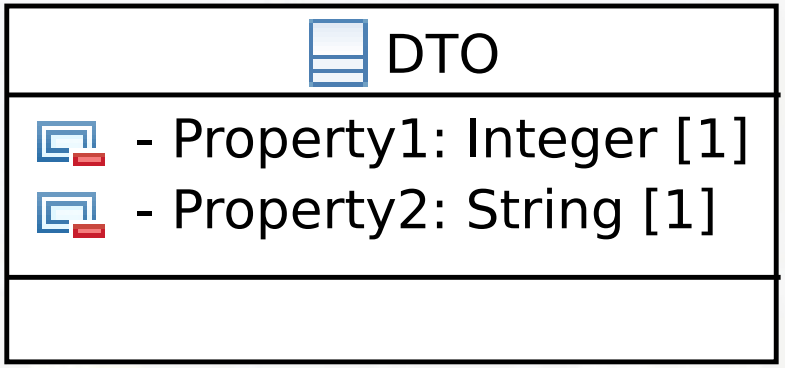


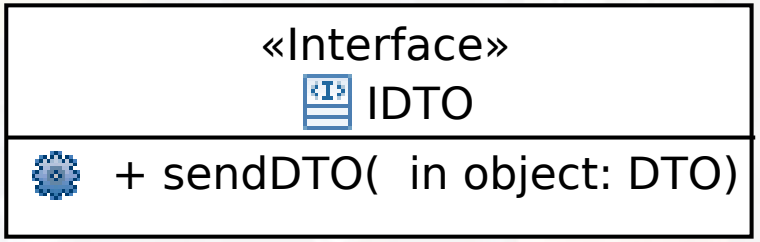
Data Transfer Object (DTO)
Enterprise Design Pattern

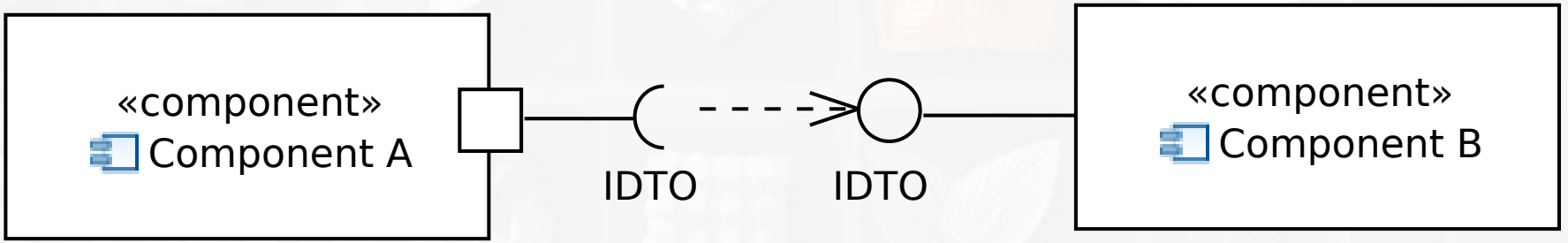
- “Um objeto que carrega dados entre processos para reduzir o número de chamadas de métodos.” (Fowler,2003)

“An object that carries data between processes in order to reduce the number of method calls.” (Fowler,2003)

Data Transfer Object (DTO)
Enterprise Design Pattern

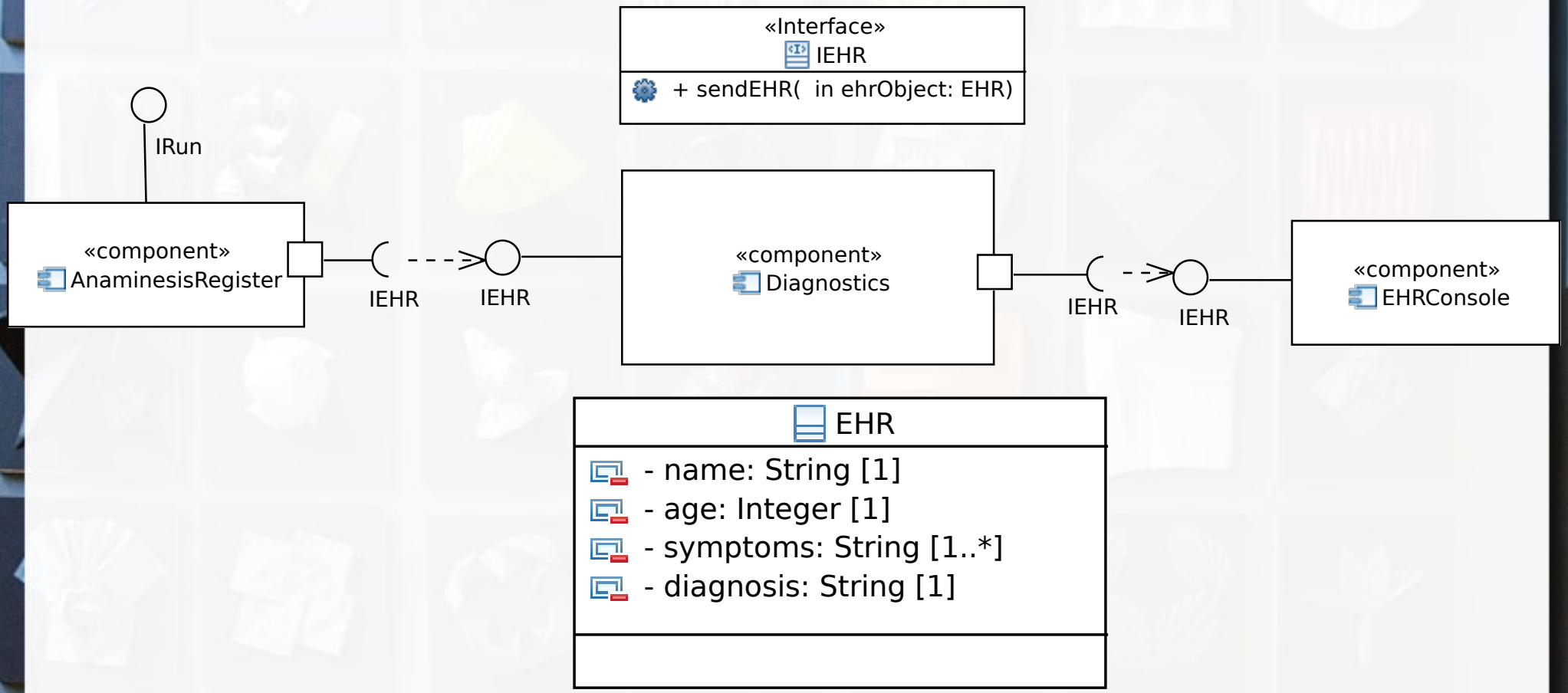




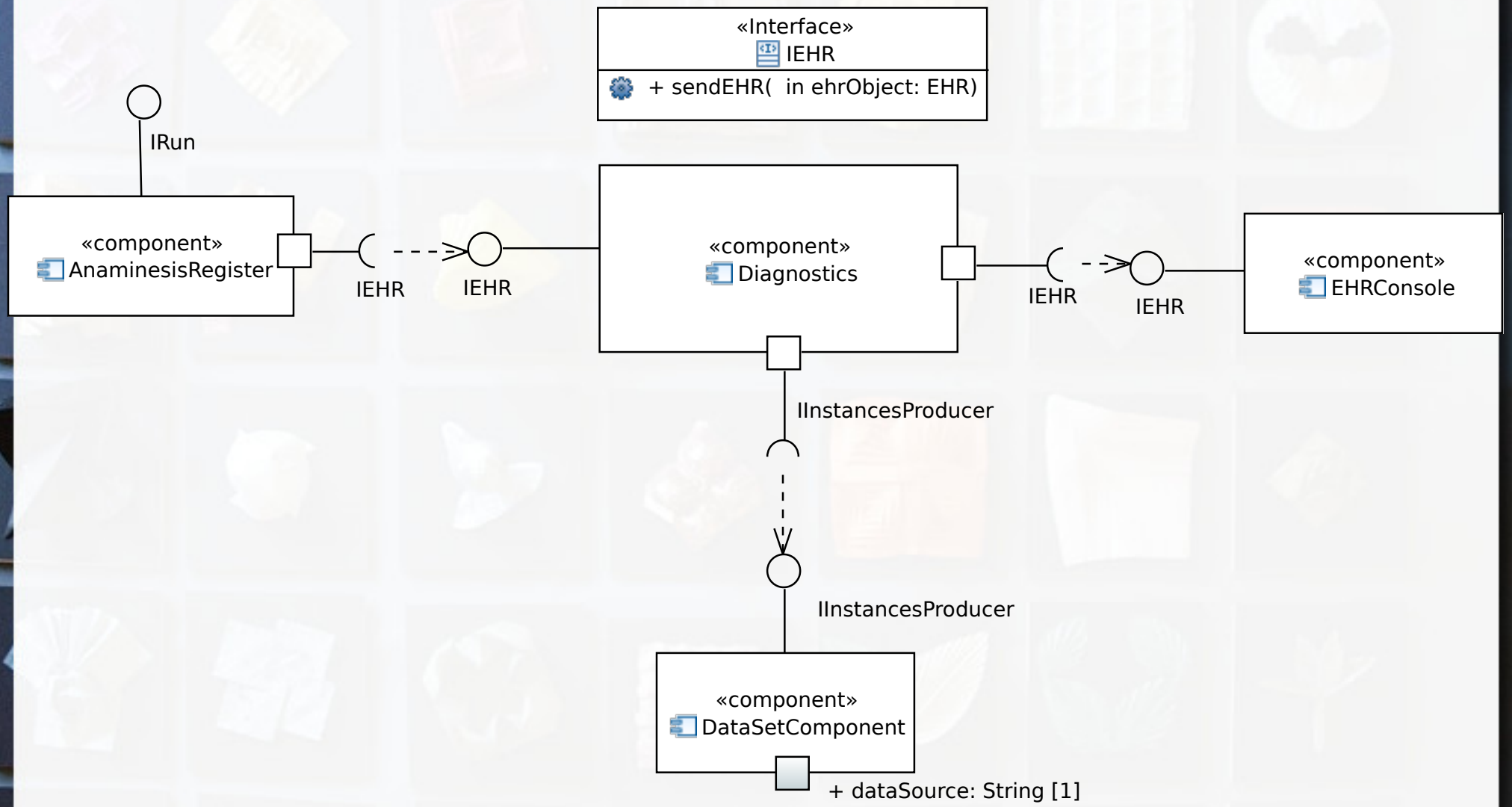


Tarefa 5
Data Transfer Object (DTO)

- Reduza a complexidade da Interface, que tem muitos parâmetros.



Tarefa 5
Versão Completa



Composição Orientada a Dados
Data-driven Composition

- “[...] várias formas de composição orientadas a dados, onde a chegada de uma mensagem de um certo tipo ou com um certo conteúdo determina o próximo passo de processamento.” (Szyperski, 2002)

“[...] various forms of data-driven composition, where the arrival of a message of a certain type or with certain contents determines the next processing step.” (Szyperski, 2002)

Eventos

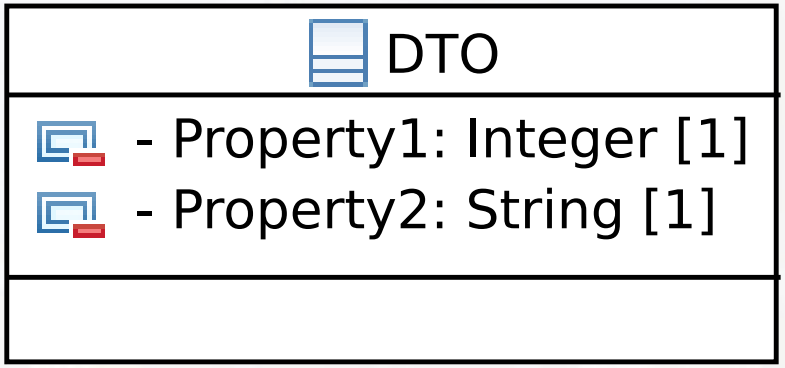
Eventos

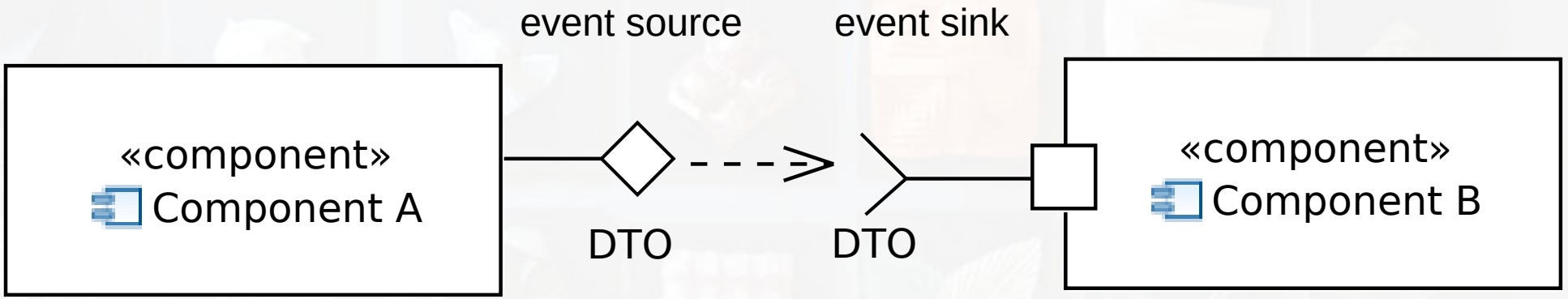
- Componentes podem interagir através da difusão (broadcast) de eventos
- Ação inicia com um componente que 'anuncia' um evento
- Evento anunciado pode disparar operações em outros componentes

(Abowd, 1995)

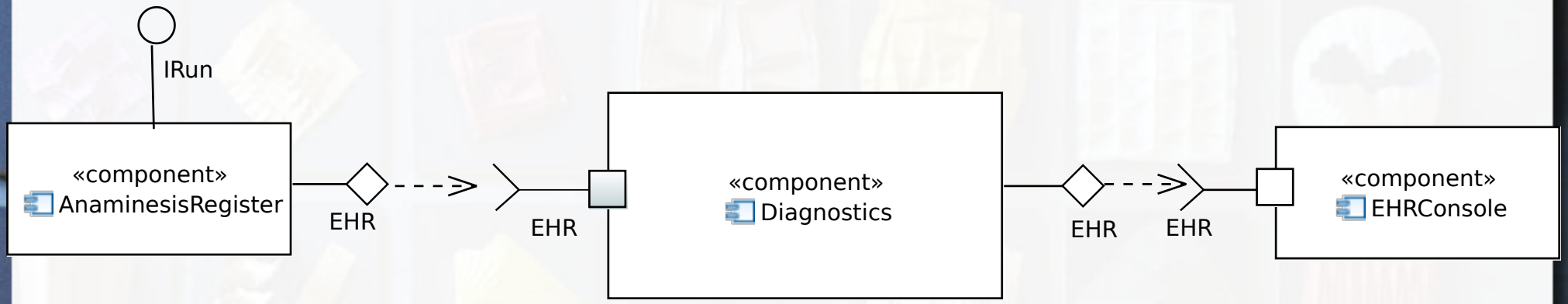
- Exemplo: Publish-Subscribe

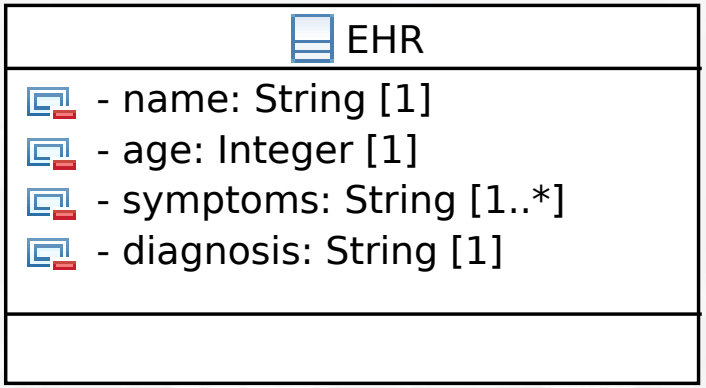
Eventos
(notação CORBA Component Model)



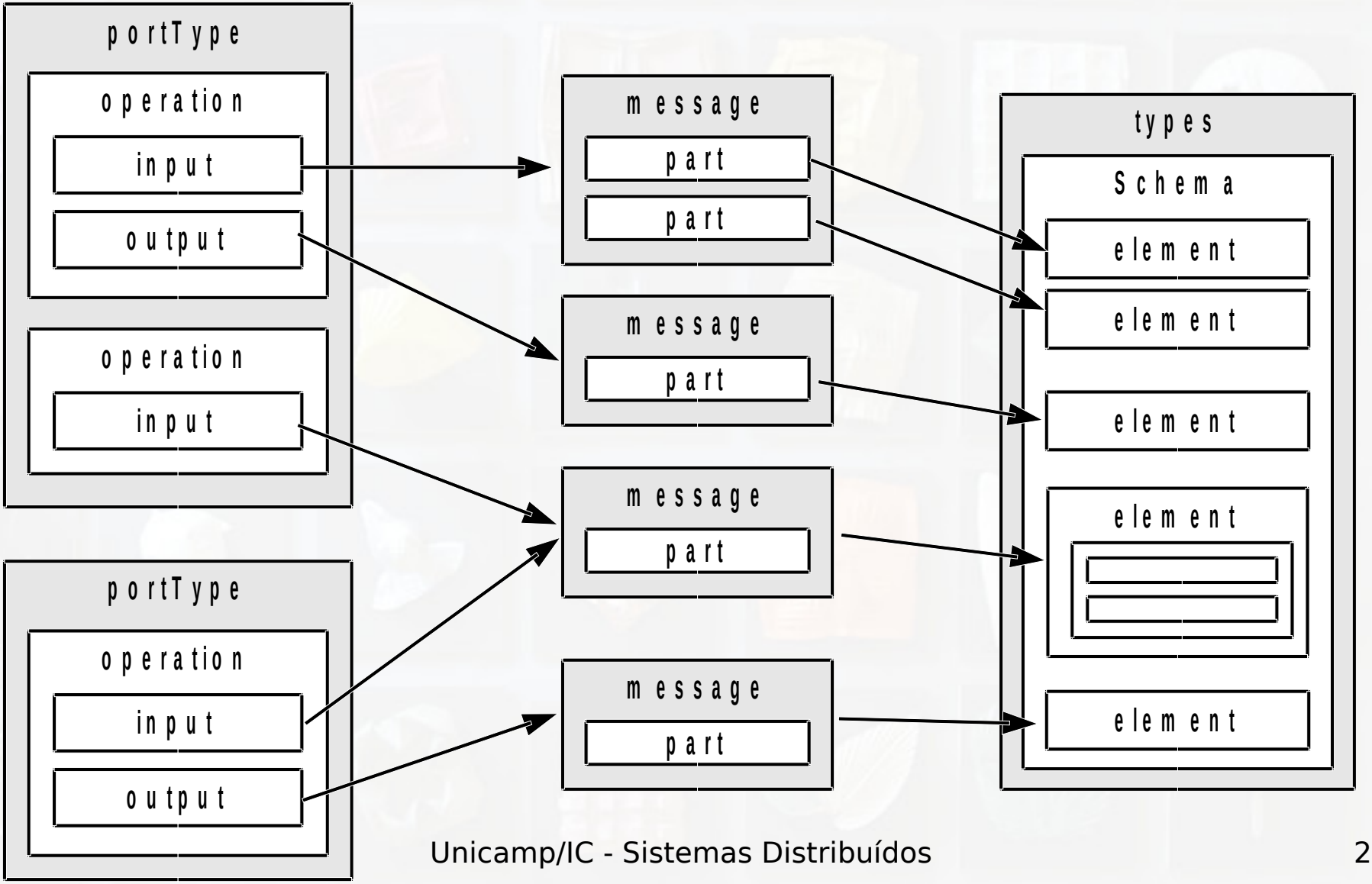


Tarefa 5 - Eventos
(notação CORBA Component Model)

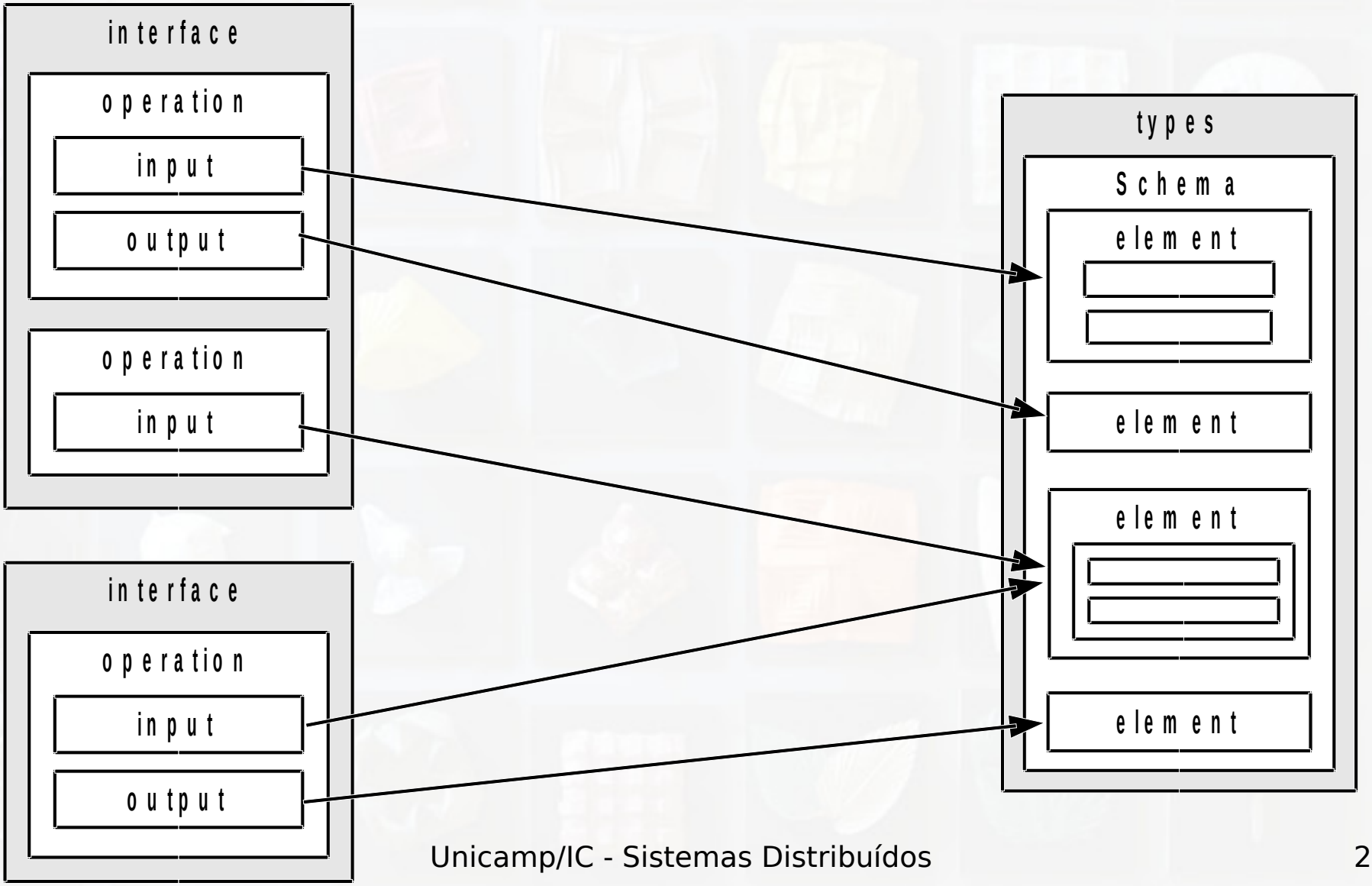




Web Services
WSDL 1

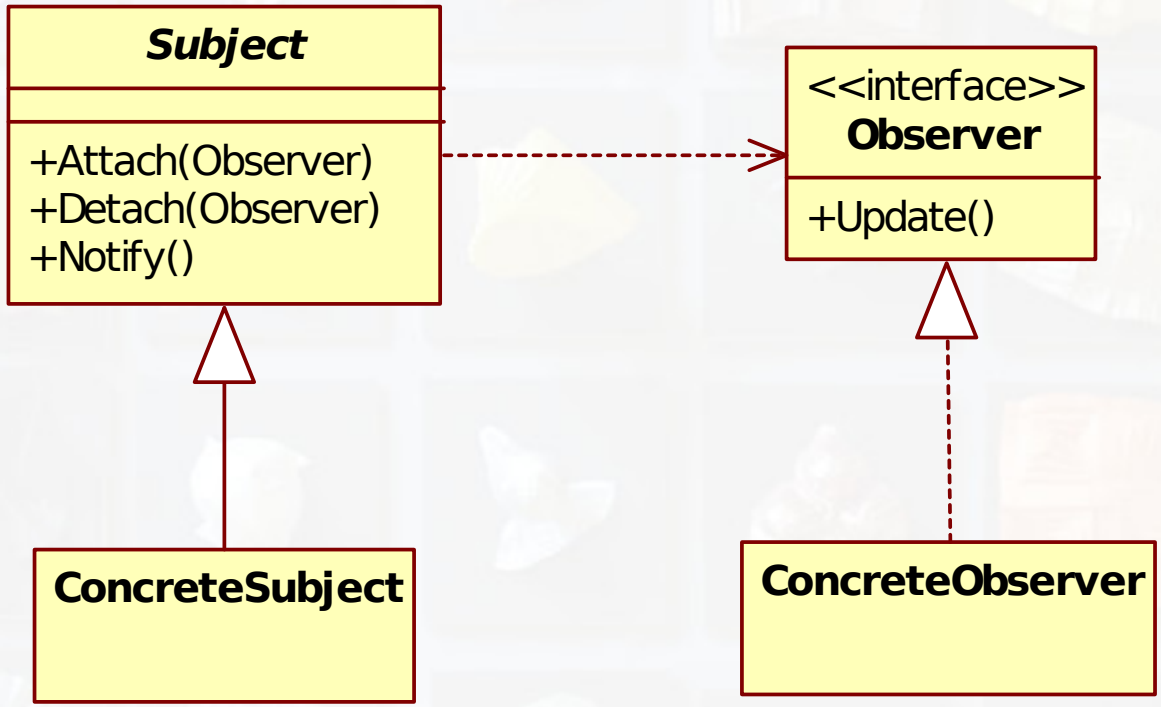


Web Services
WSDL 2



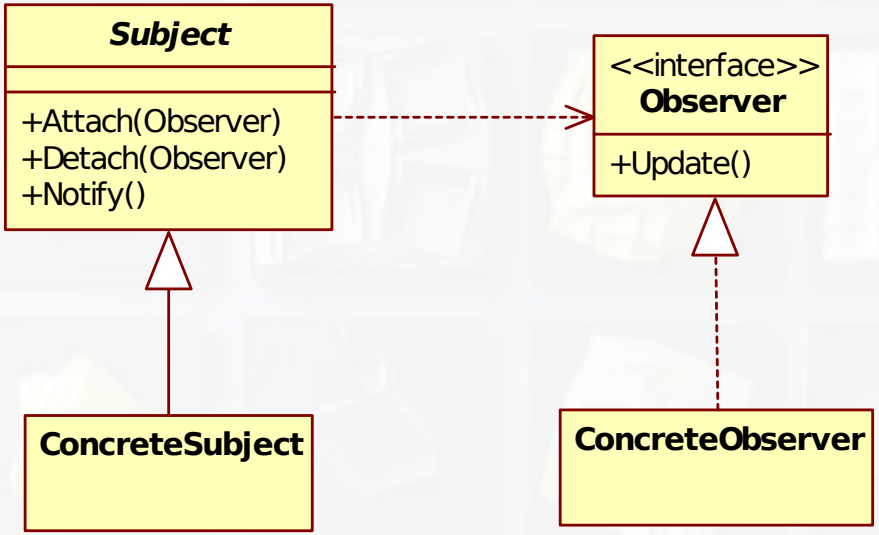
Pattern Observer

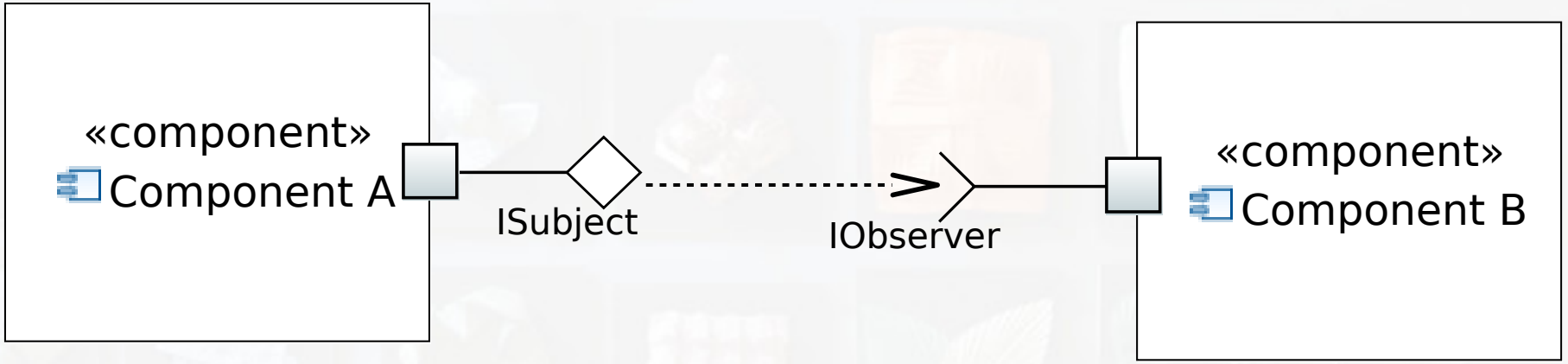
Eventos
Pattern *Observer*



Eventos (notação CORBA Component Model)

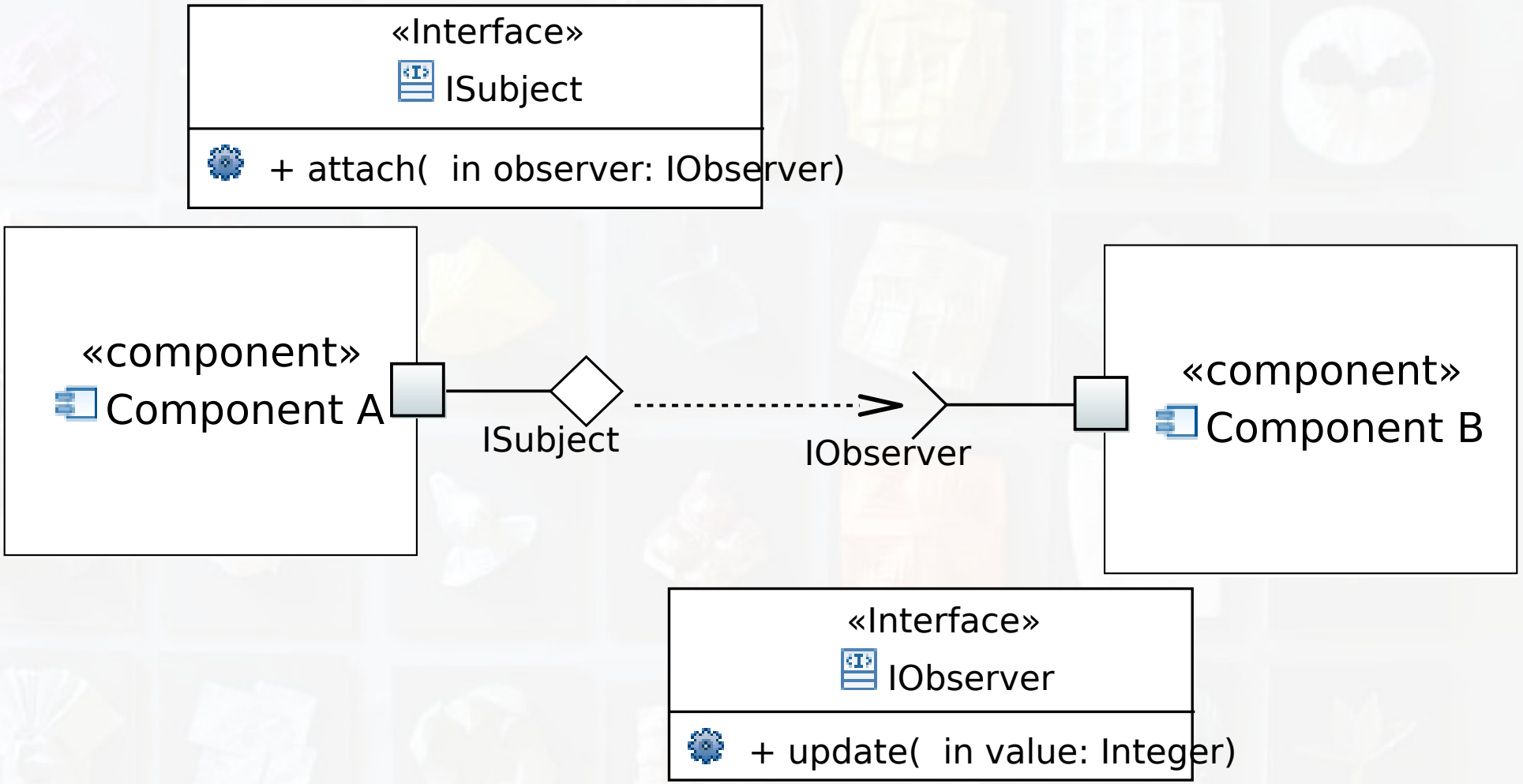
Pattern *Observer*



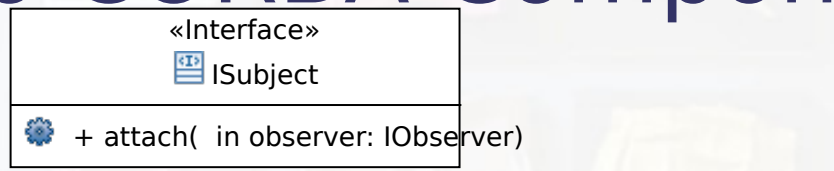


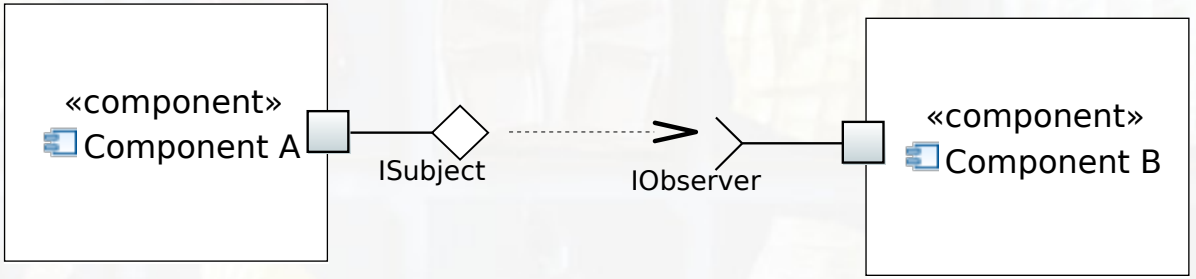
Eventos (notação CORBA Component Model)

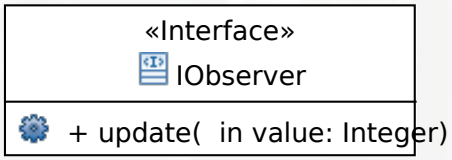
Pattern *Observer*

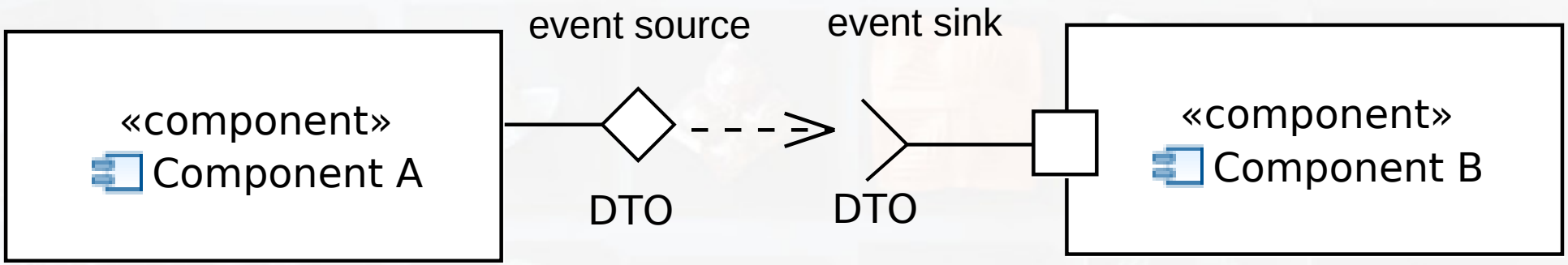


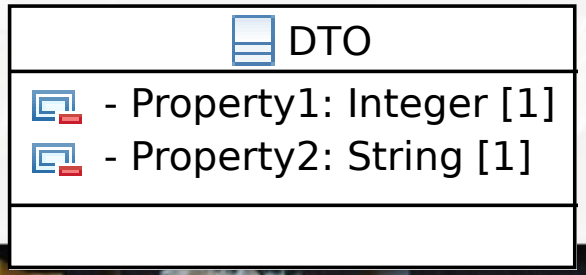
Foco no DTO
(notação CORBA Component Model)



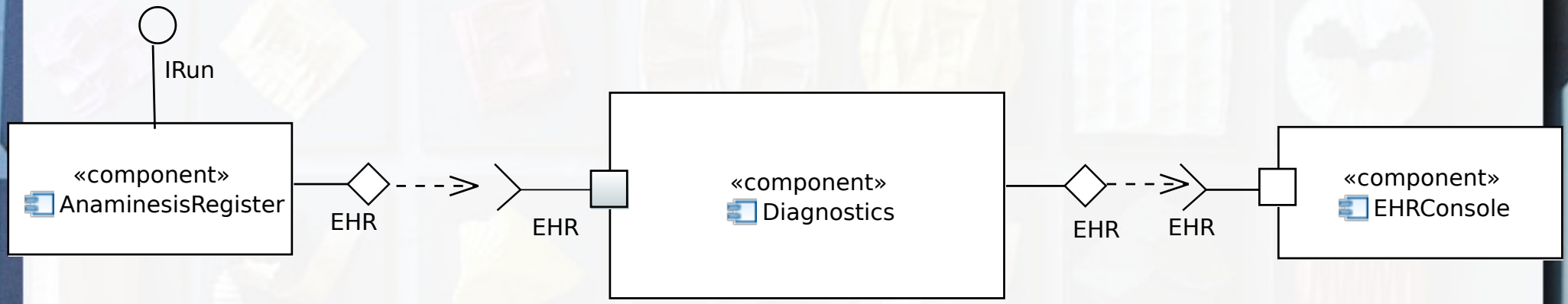


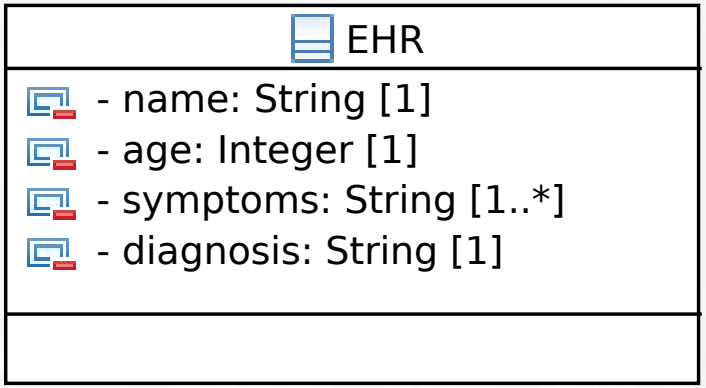






Tarefa 5 - Eventos
(notação CORBA Component Model)





Estilos Arquiteturais

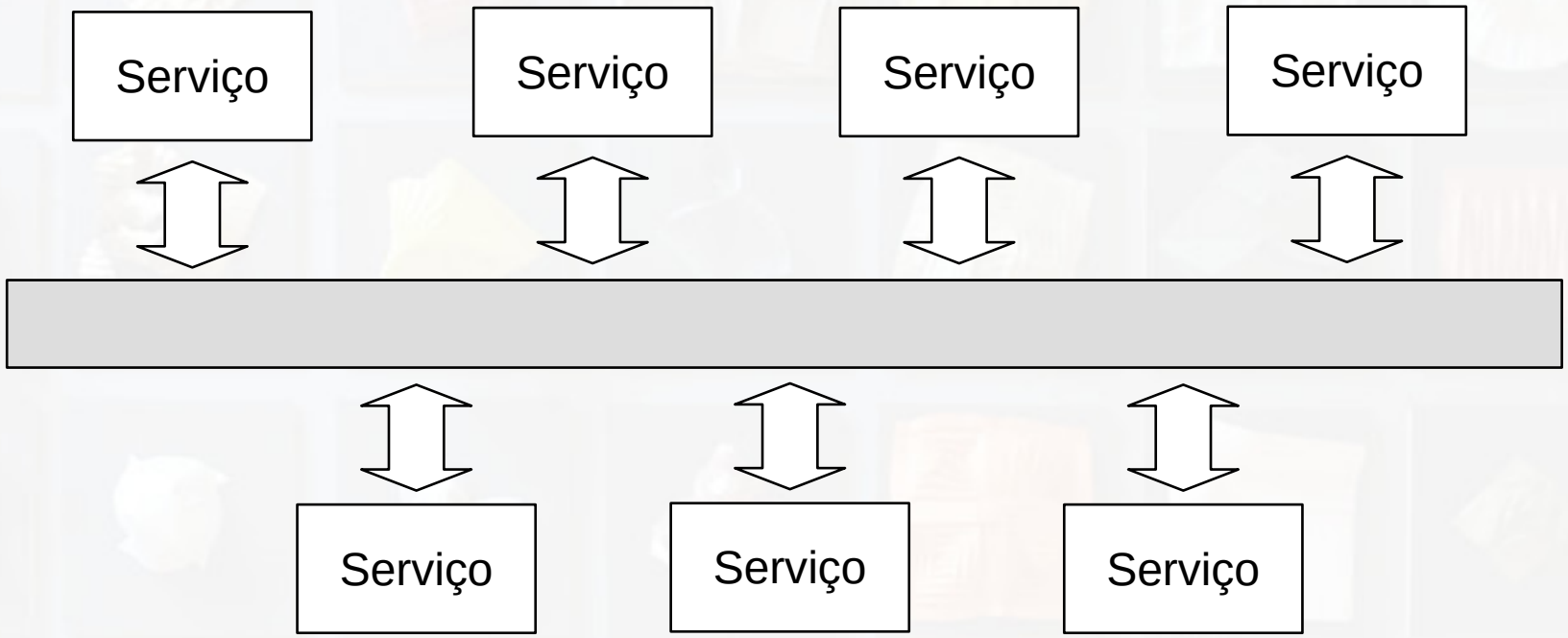
Baseado em Mensagens

Baseado em Mensagens

- Comunicação realizada exclusivamente pela troca de mensagens
- Tipicamente usada por aplicações assíncronas

(Taylor, 1992)

Barramento de Mensagens
Message Bus



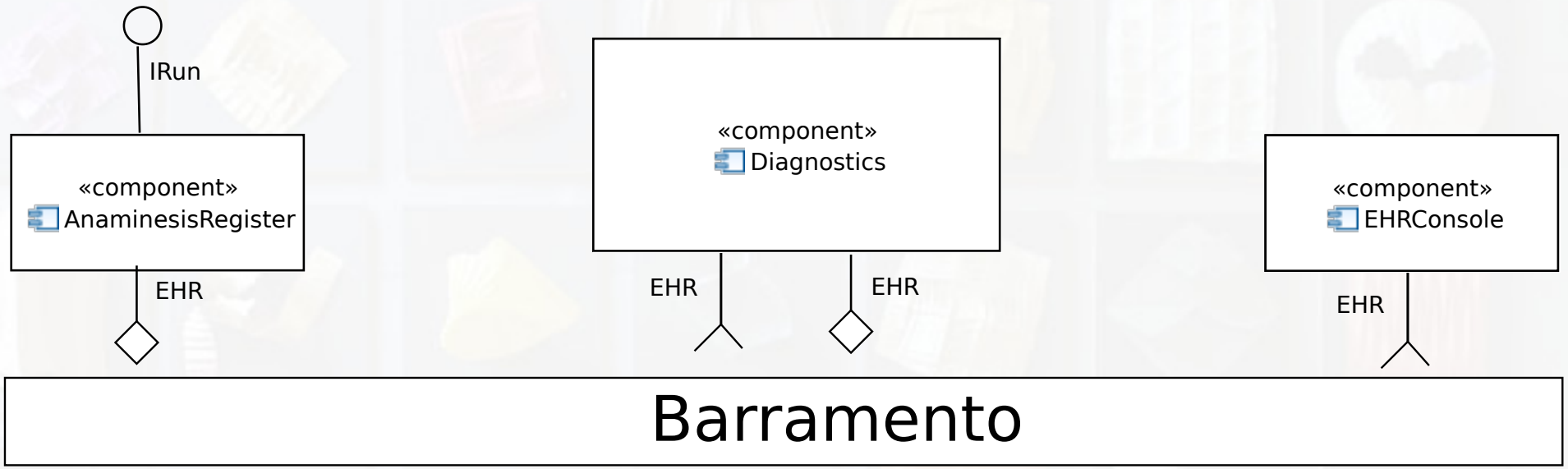
Modelo Broadcast

- Efetivo na integração subsistemas em computadores diferentes em rede
- Subsistemas registram interesse em evento. Quando ele ocorre, o controle é transferido para o sistema que pode tratá-lo.
- Política não está embutida no manipulador eventos ou mensagens. Subsistemas decidem os eventos que os interessam.
- Contudo, subsistemas não sabem quando um evento será tratado."[1]

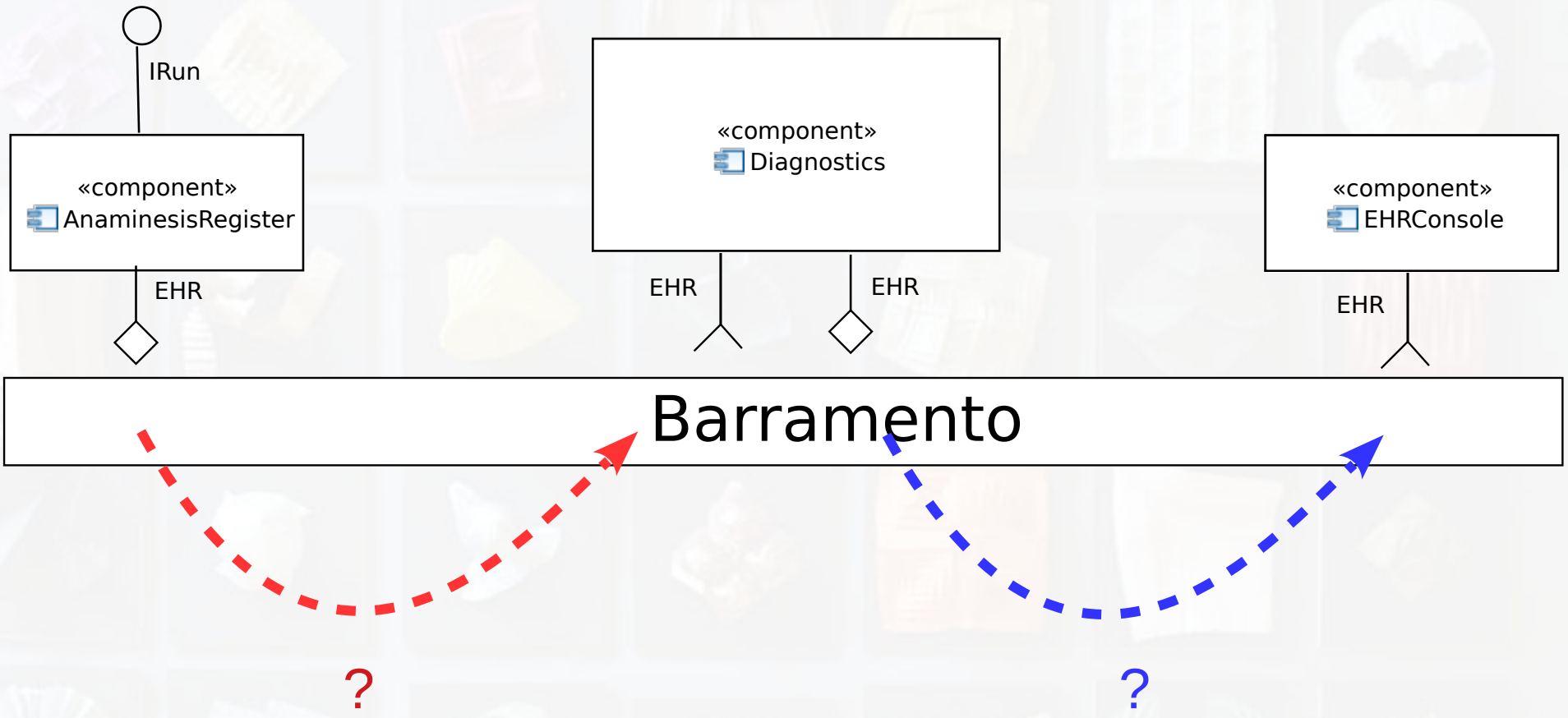
(Sommerville, 2007)

"- Effective in integrating sub-systems on different computers in a network.
- Sub-systems register an interest in specific events. When these occur, control is transferred to the sub-system which can handle the event.
- Control policy is not embedded in the event and message handler. Sub-systems decide on events

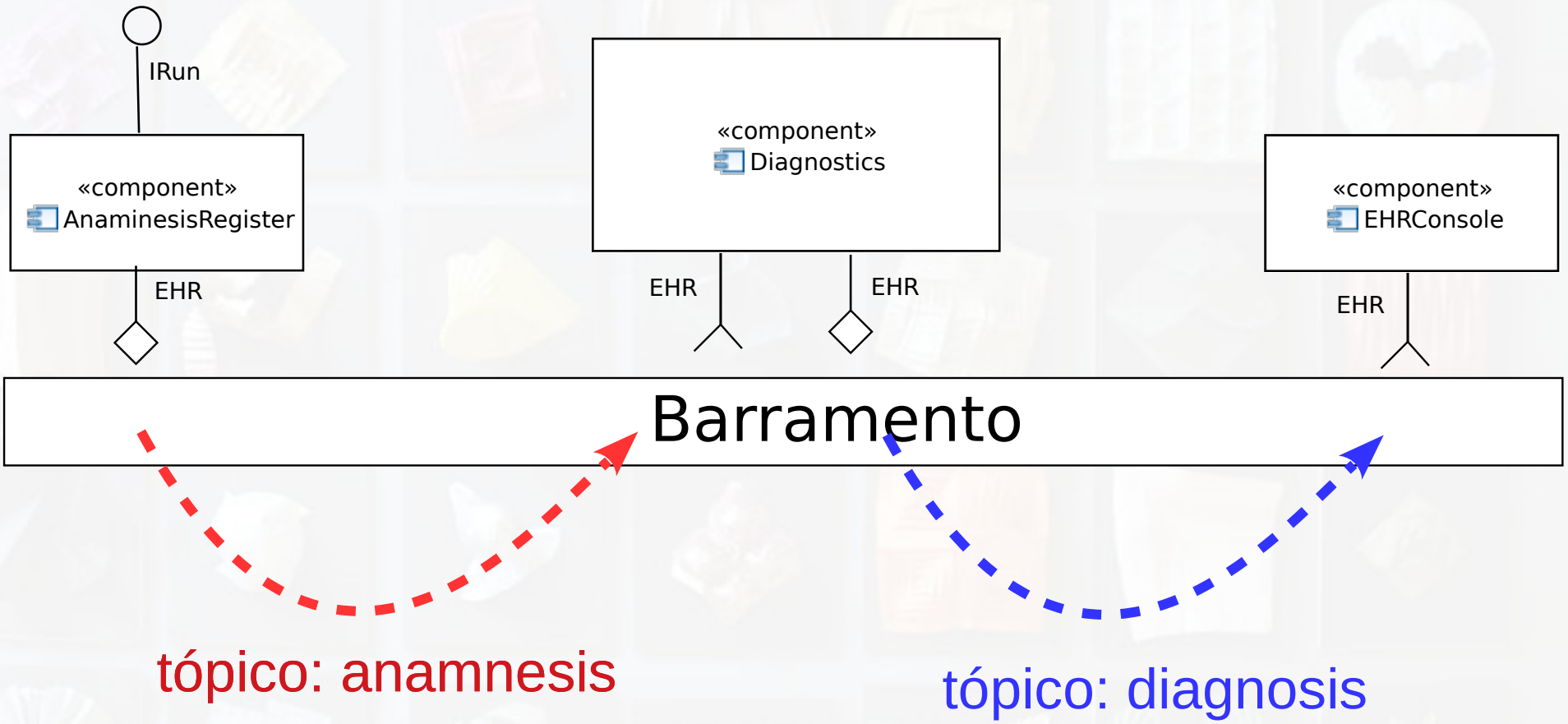
Modelo Broadcast



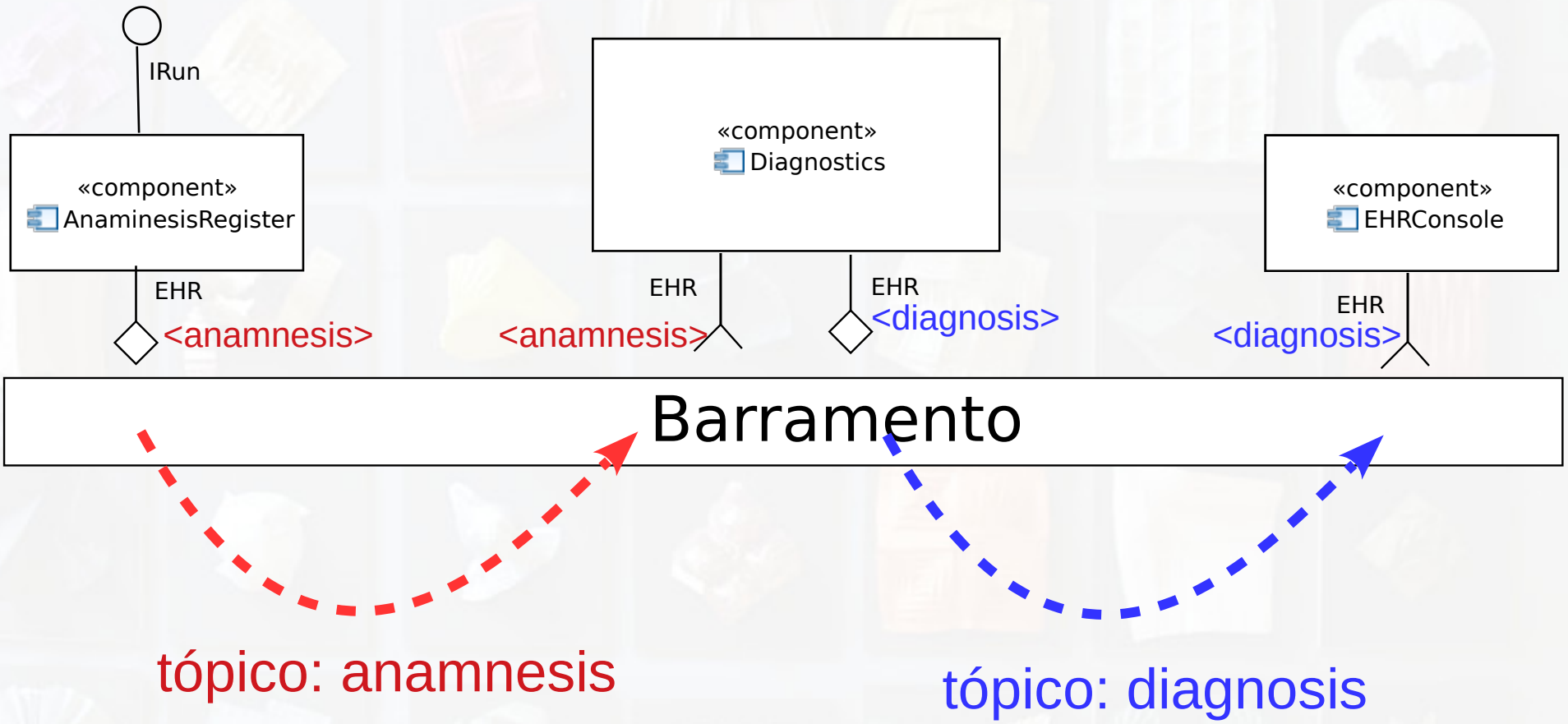
Modelo Broadcast



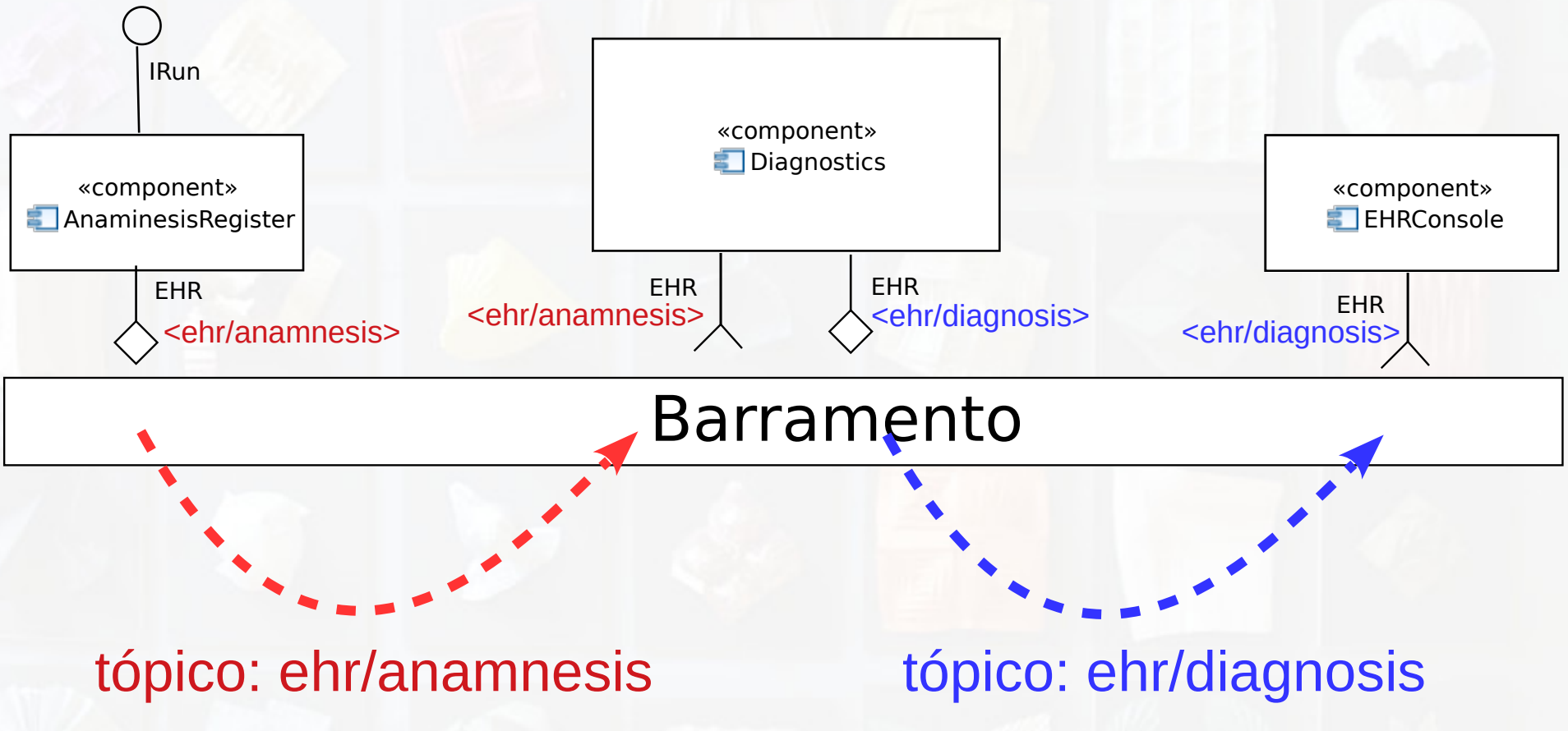
Modelo Broadcast
Publish/Subscribe



Modelo Broadcast
Publish/Subscribe



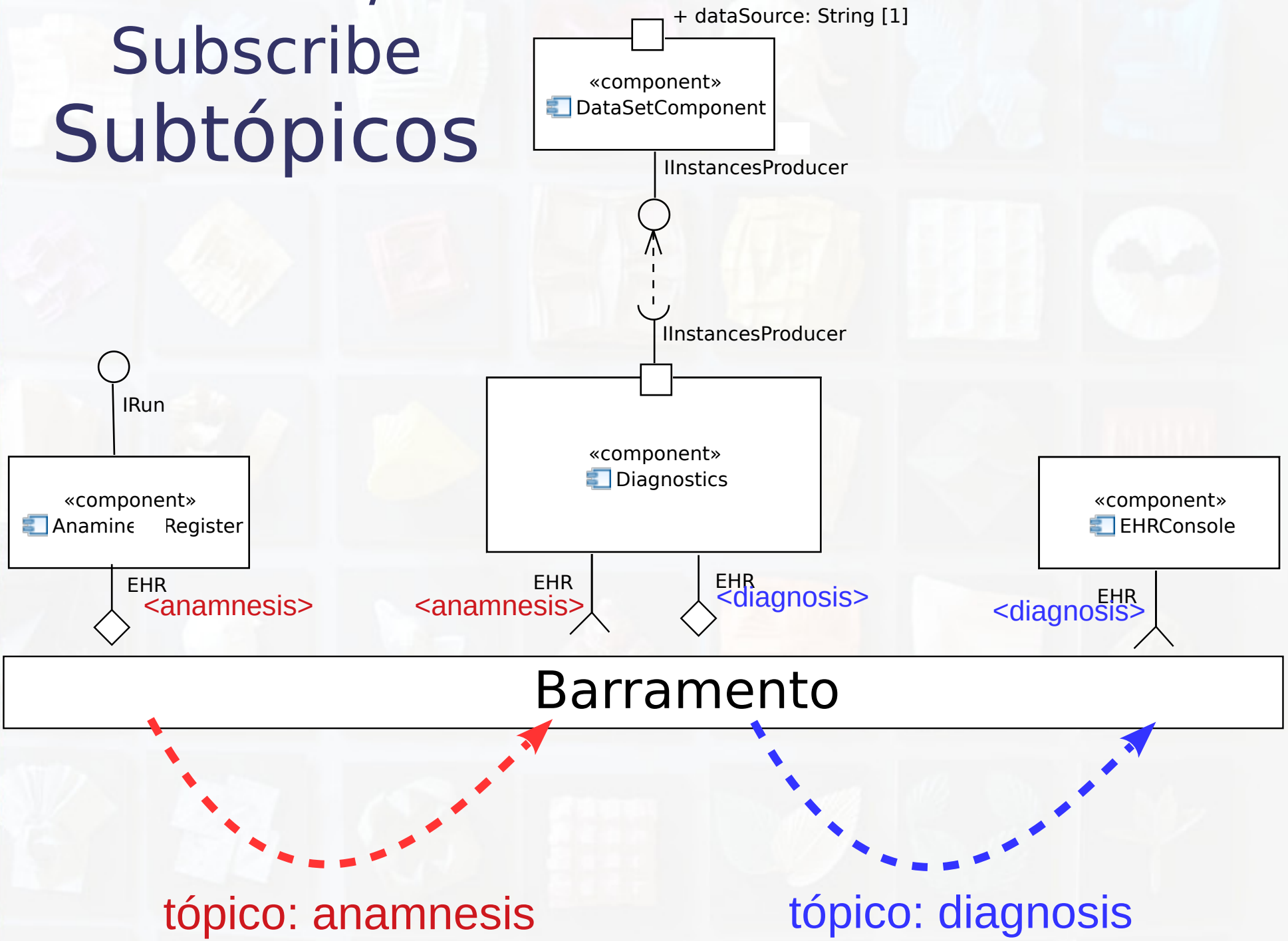
Publish/Subscribe
Subtópicos



tópico: ehr/anamnesis

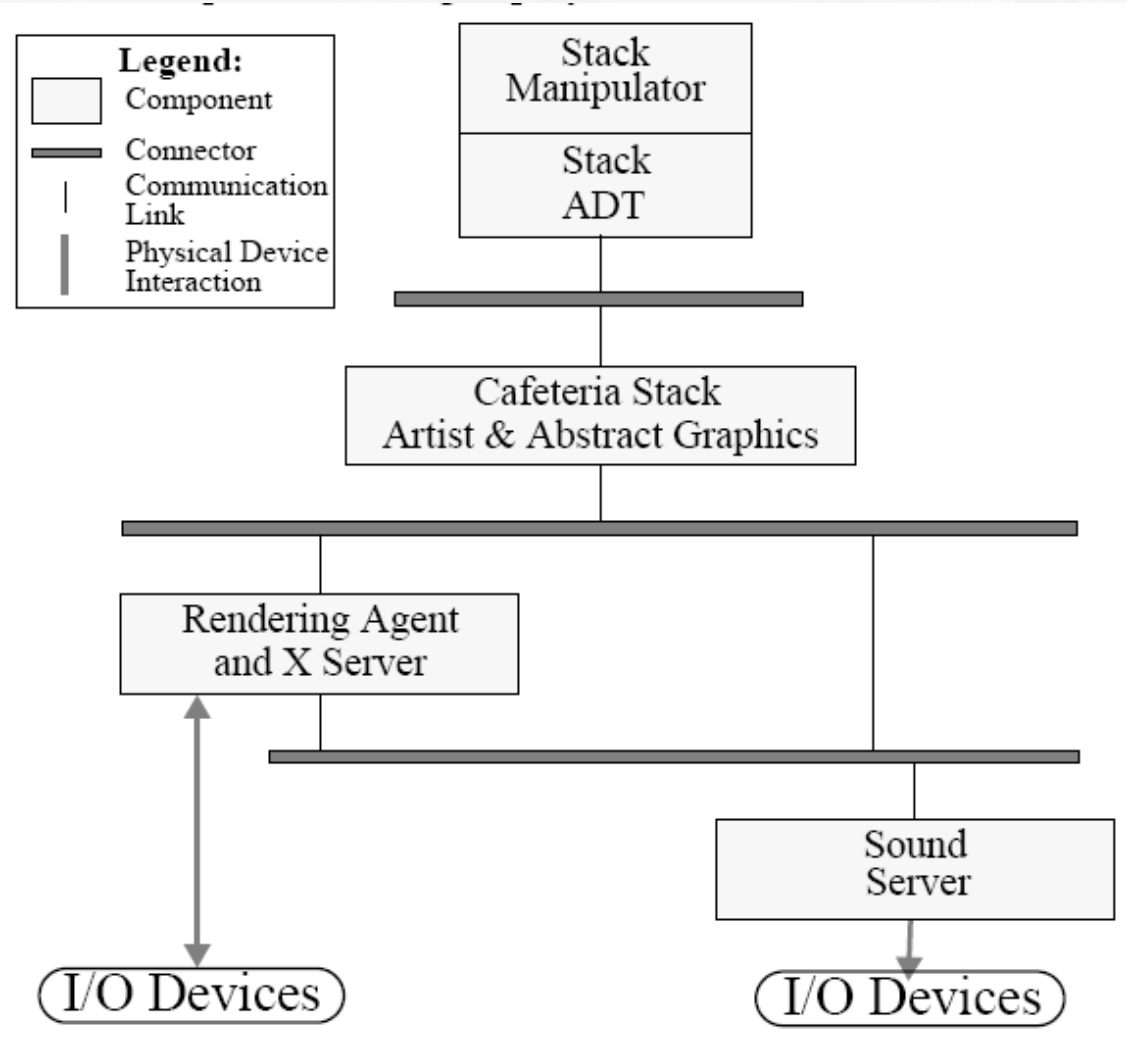
tópico: ehr/diagnosis

Publish/ Subscribe Subtópicos



Múltiplos Barramentos

C2



(Taylor, 1992)