
LOOK MOMMY, SYSTEM ENGINEERS!
BASEBALL
DOES NOT MATTER, THEY ARE JUST REGULAR ...
RICKY....
SYSTEMS, REQUIREMENTS, INTERFACES, EMERGENCT BEHAVIOUR...
imgflip.com
- EXTRA FABULOUS COMICS -
ZMS

REVISÃO

- Existem diagramas que são **clássicos**. E é importante que consigamos usá-los e entendê-los pois são a base da linguagem da Engenharia de Sistemas.
  - **eFFBD** – sequencializa as funções de um Sistema
  - **DFD** – inclui que é possível indicar atores e o que é transitado de uma função para outra
  - **N2** – permite indicar as interelações através de uma matriz
  - **IDEF0** - permite indicar a sequencia de um processo
  - **Diagrama de Blocos – interligação básica entre blocos representando as relações (trocas) entre os blocos e decomposição interna dos blocos.**

ARQUITETURA

[2024]

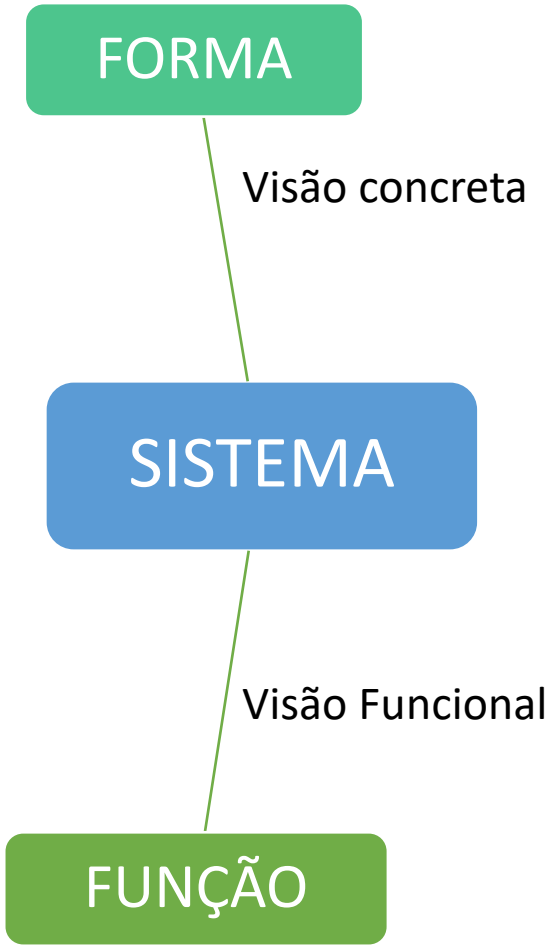
Prof. Dr. Christopher S. Cerqueira

	SEMANA	TEORIA	INDIVIDUAL	PESO	GRUPO	PESO
Teoria, Linguagem e Lógica	1	Estrutura do Curso e Apresentações Pessoais	AI-01 - Leitura/Resumo Cap 1 - HB INCOSE	5%	AG-01 - Entrega dos grupos.	0%
	04-Mar	O que é Engenharia de Sistemas? INCOSE				
	08-Mar	Contexto: Aeronave x Sistemas do Dom. Aéreo				
	2	Gramática: Representações clássicas	AI-02 - Leitura/Resumo paper sobre representações clássicas.	5%	AG-02 -	0%
	11-Mar					
	15-Mar	Lógica: Diagrama de Blocos				
	3	Gramática: Arquitetura e Visões	AI-03 - Exercícios sobre arquitetura funcional e diagrama de blocos.	5%	AG-03 - Resumo sobre arquitetura funcional.	10%
	18-Mar	Gramática: Funções				
	22-Mar	Lógica: Diagrama de Classes				
	4*	Gramática: Stakeholders	AI-04 - Exercícios sobre diagrama de casos de uso	5%	AG-04 - Resumo sobre stk, ciclo de vida e CONOPs	10%
	25-Mar	Gramática: Ciclo de Vida e CONOPS				
	29-Mar	Lógica: Diagrama de Casos de Uso				
	5	Gramática: Requisitos	AI-05 - Exercício de escrita de requisitos.	10%	AG-05 - Resumo sobre requisitos vindos da parte de segurança (safety) do RBAC.	10%
	01-Apr	Gramática: Verificação e Validação				
	05-Apr	Contexto: ANAC (RBACs)				
	6	Lógica: Diagrama de Estados	AI-06 - Exercício sobre diagrama de estados.	10%	AG-06 - Revisita do CONOPs do DECEA usando storyboards	15%
	08-Apr					
	12-Apr	Contexto: DECEA (ICAs/DCAs)				
	7	Lógica: Diagrama de Sequência	AI-07 - Exercício sobre diagrama de sequencia.	0%	AG-07 - Representação do CONOPs usando Diagrama de Sequencia.	20%
	15-Apr					
	19-Apr	Temas do projeto do segundo semestre e P1				
	8	P1	AI-08(P1) - Teoria e linguagem (Pres/Consulta - sem chatGPT)	60%	AG-01(P1) - Apresentação Gravada: 20min	40%
	22-Apr					
	26-Apr					
				100%		**105%**

OBJETIVOS DA AULA DE HOJE

- Apresentar o conceito de arquitetura
- Visões diferentes do Sistema
- Definição de Forma e Função
- Diagrama de Classes

ARQUITETURA



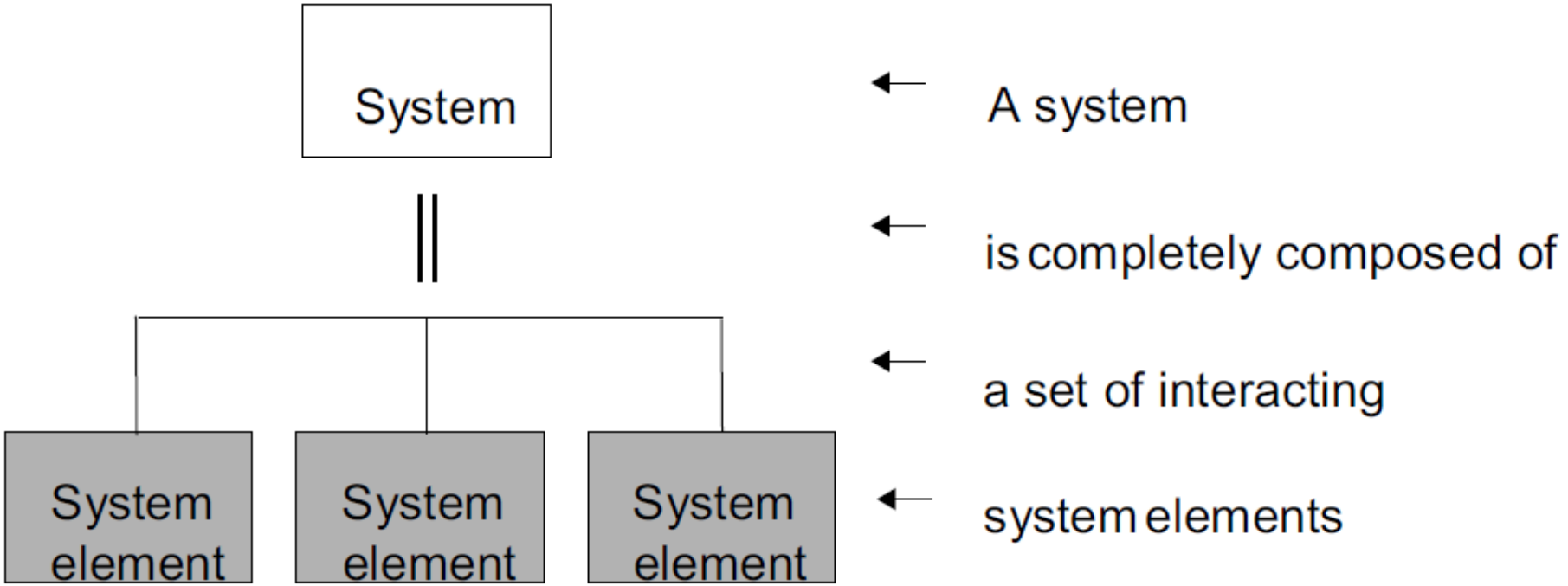
- Um relacionamento funcional requer um relacionamento entre formas.
- A relação da forma é o instrumento da relação funcional.

FILOSOFIA

- **Forma é o que foi ou é eventualmente implementado.**
- **Forma é sobre existência.**
- **Forma é o que o sistema é**. É a manifestação concreta e muitas vezes visível do sistema.
-
- A forma é a personificação física ou informacional de um sistema que existe ou tem potencial para existir, por algum período, e é instrumental na execução da função. As formas incluem as entidades de forma e as relações entre as entidades. A forma existe antes da execução da função.
- Forma é um atributo produto/sistema.

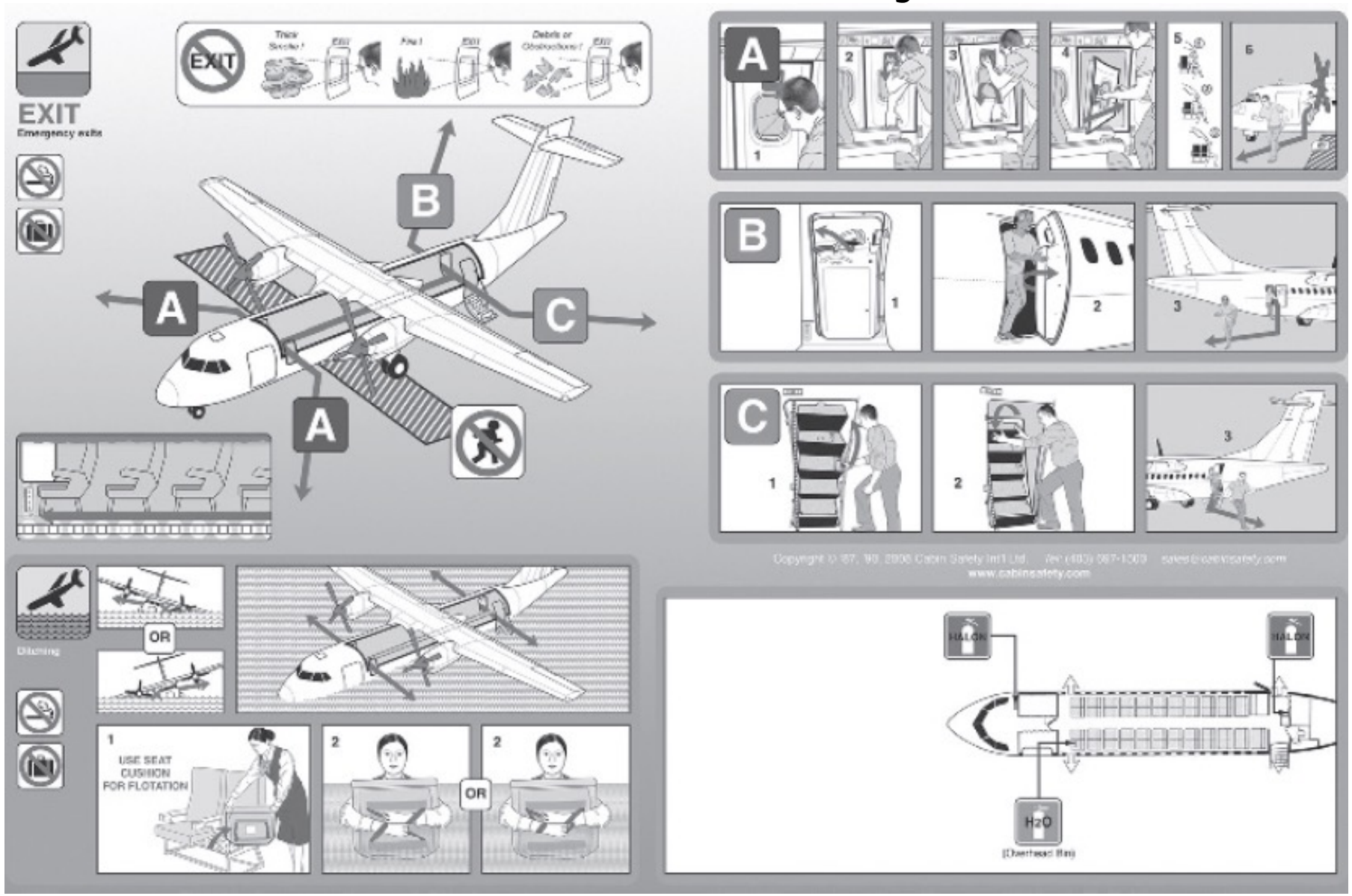
HIERARQUIA DO SISTEMA

- Relação entre sistema e elemento do sistema

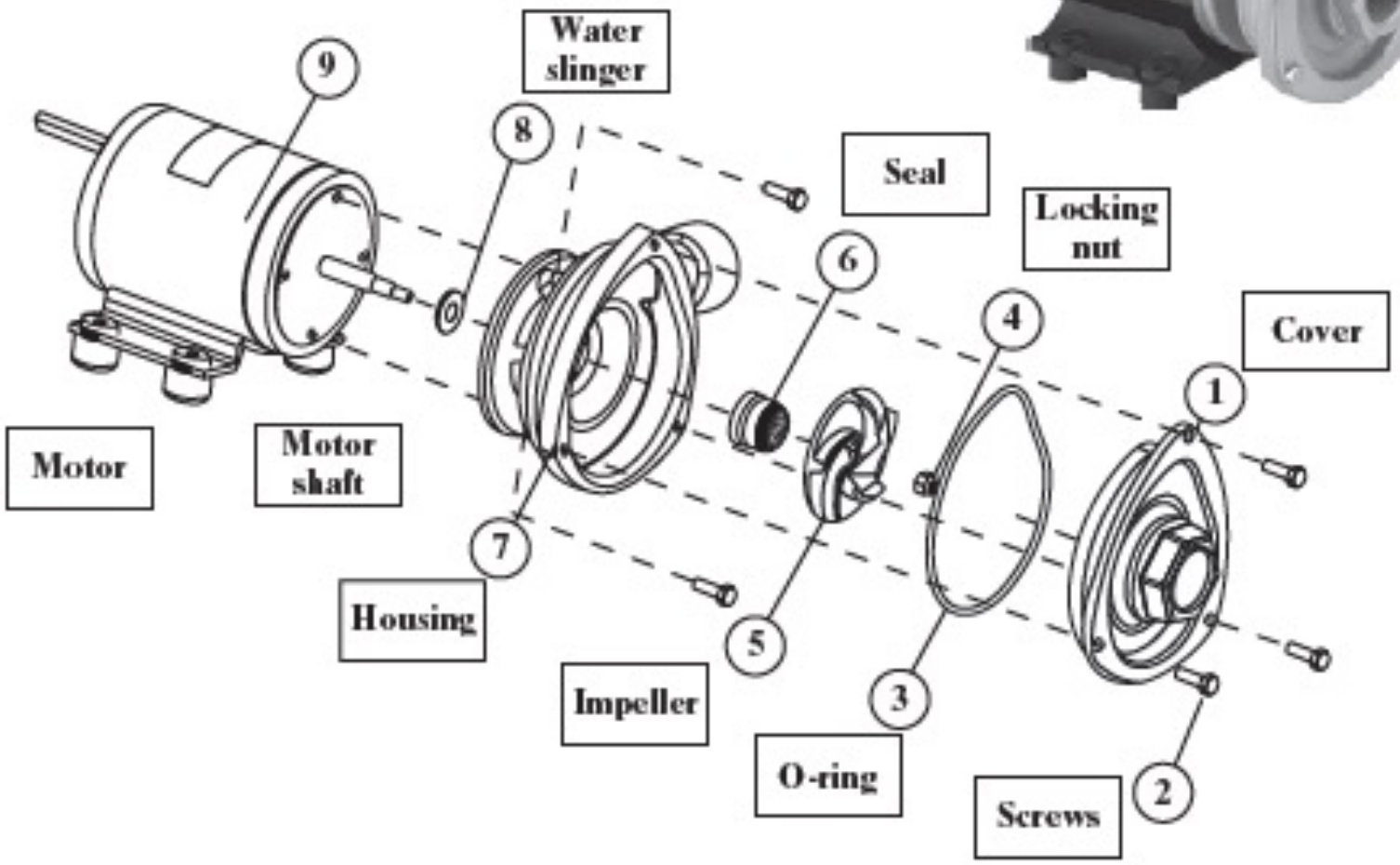


O sistema é decomposto em uma hierarquia de elementos cada vez menores.

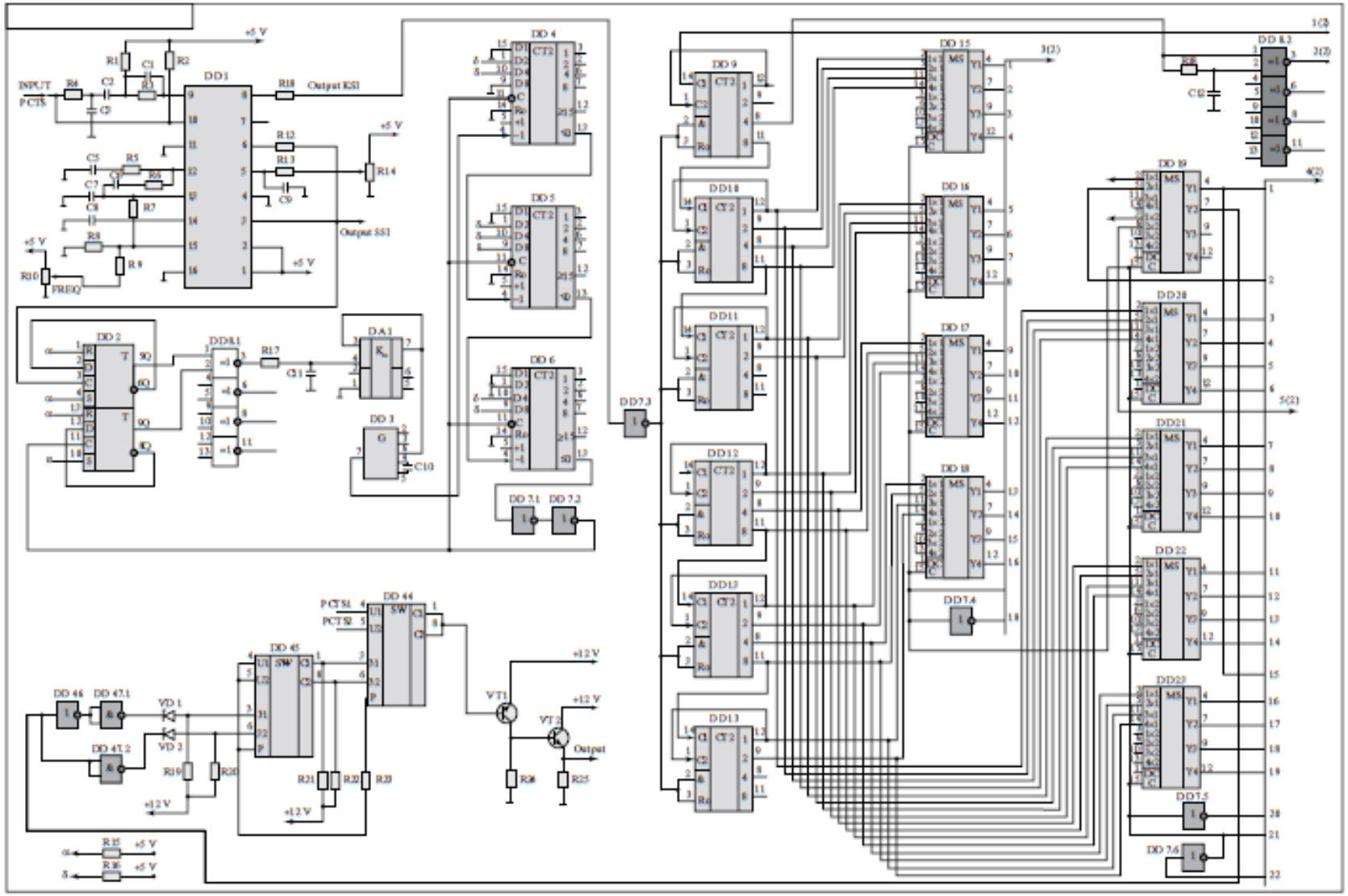
EXEMPLO DE FORMA: ILUSTRAÇÕES



EXEMPLO DE FORMA: BOMBA CENTRÍFUGA



EXEMPLO DE FORMA: COMPONENTES DO CIRCUITO



EXEMPLO DE FORMULÁRIO: CÓDIGO DE SOFTWARE

```
1   Procedure bubblesort (List array, number length_of_array)
2         for i=1 to length_of_array - 1;
3               for j=1 to length_of_array – I;
4                     if array [j] > array [j+1] then
5                           temporary = array [j+1]
6                           array[j+1] = array [j]
7                           array[j] = temporary
8                     end if
9               end of j loop
10        end of i loop
11  return array
12  End of procedure
```

ARKHITEKTON

http://ordinary-citizen.com/2018/05/07/roger-scruton-on-craft-and-beauty/

A **arquitetura de um sistema** é a **realização do conceito**, a **alocação da função física/informacional aos elementos da forma** e a **definição de relações entre os elementos com o contexto** circundante.

UM POUCO SOBRE FUNÇÕES

DEFINIÇÃO DE FUNÇÃO

- **Função** é uma atividade, operação ou transformação que causa ou contribui para o desempenho.
- Em sistemas, **funções são as ações para as quais um sistema existe**, o que, em última análise, leva à entrega de valor.
- A **função é executada pela forma**, que é o instrumento da função.
- A **função emerge da interação funcional** entre entidades. Função é um atributo produto/sistema.

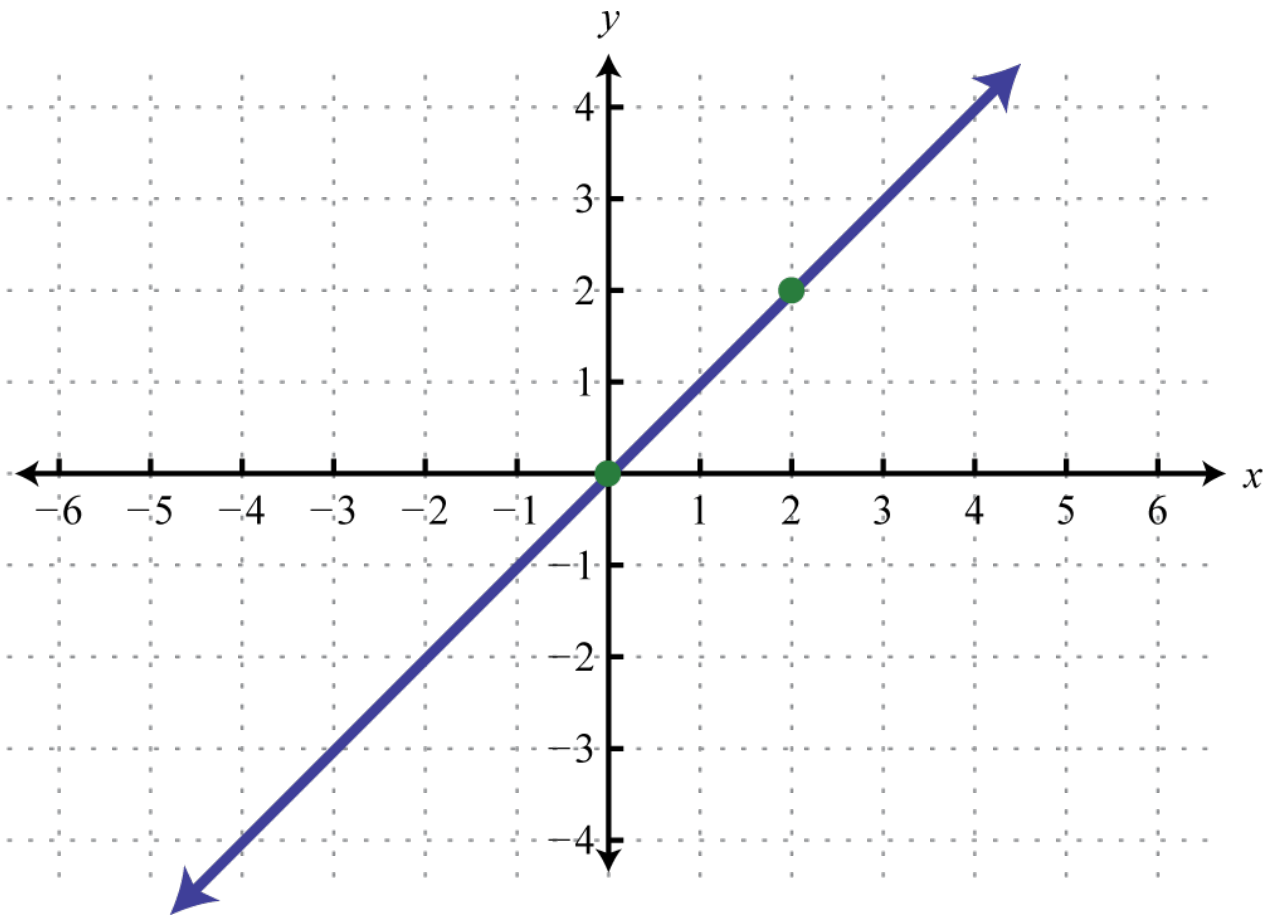
- Função é sobre atividade, em contraste com a forma, que é sobre a existência.

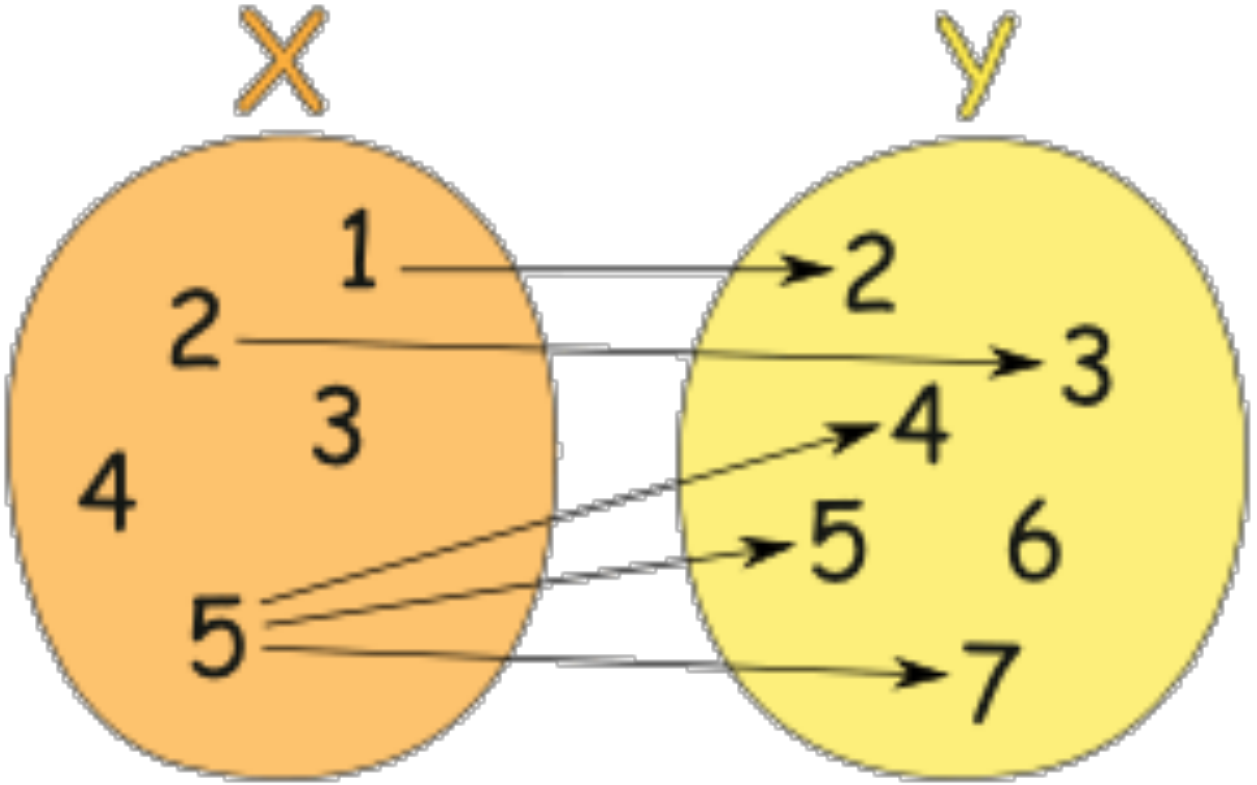
IMPORTÂNCIA DA FUNÇÃO

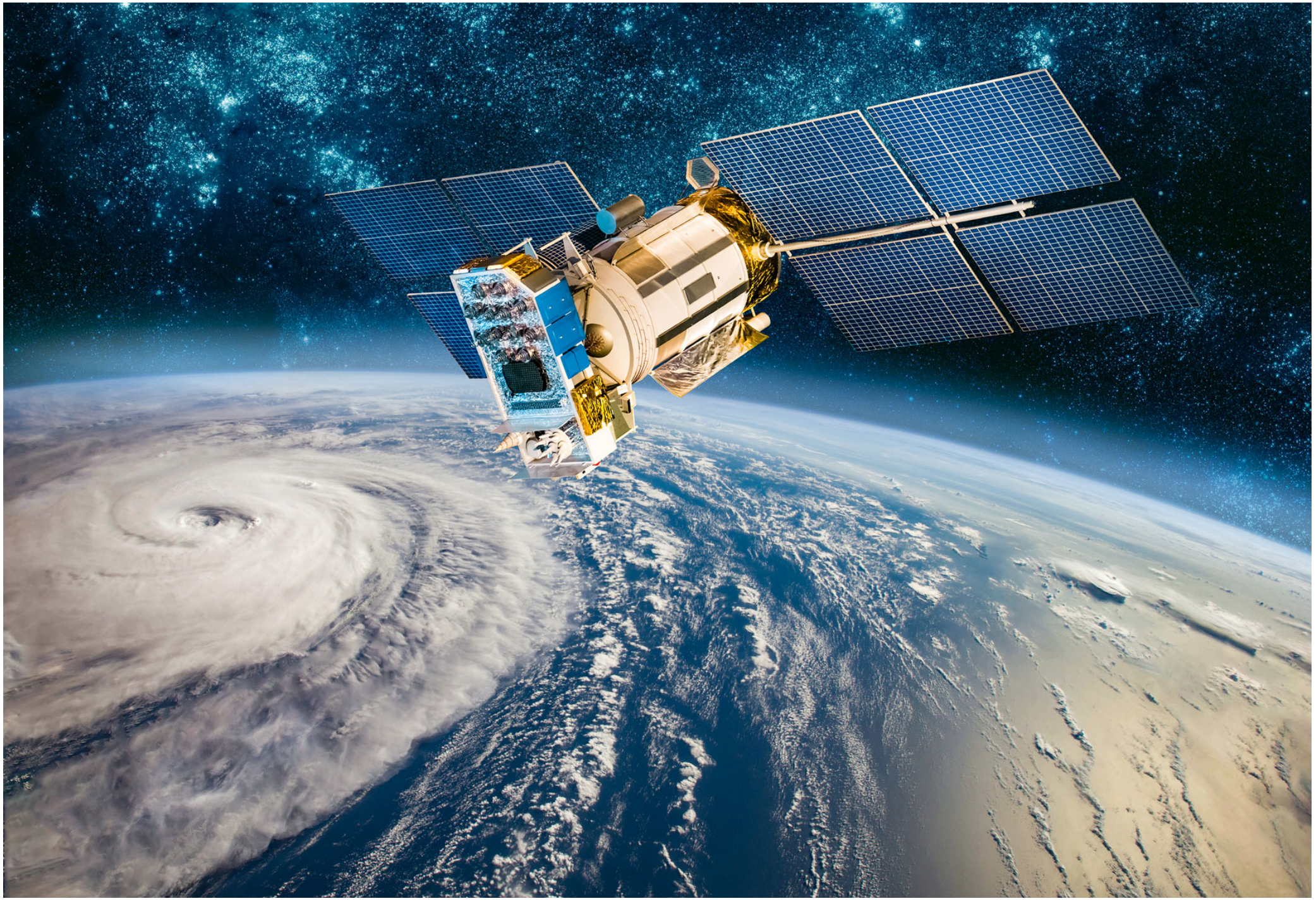
- Assim como a forma tem entidades de forma (objetos) e relações de forma entre as entidades (estrutura), a **função tem entidades de função e relações funcionais** entre as entidades (interações).
- A função que vemos na superfície de um sistema é resultado da emergencia que ocorre entre essas entidades de função.
- **Toda a magia dos sistemas e seu surgimento, e quase todo o desafio de projetá-los, é encontrada no domínio funcional.**

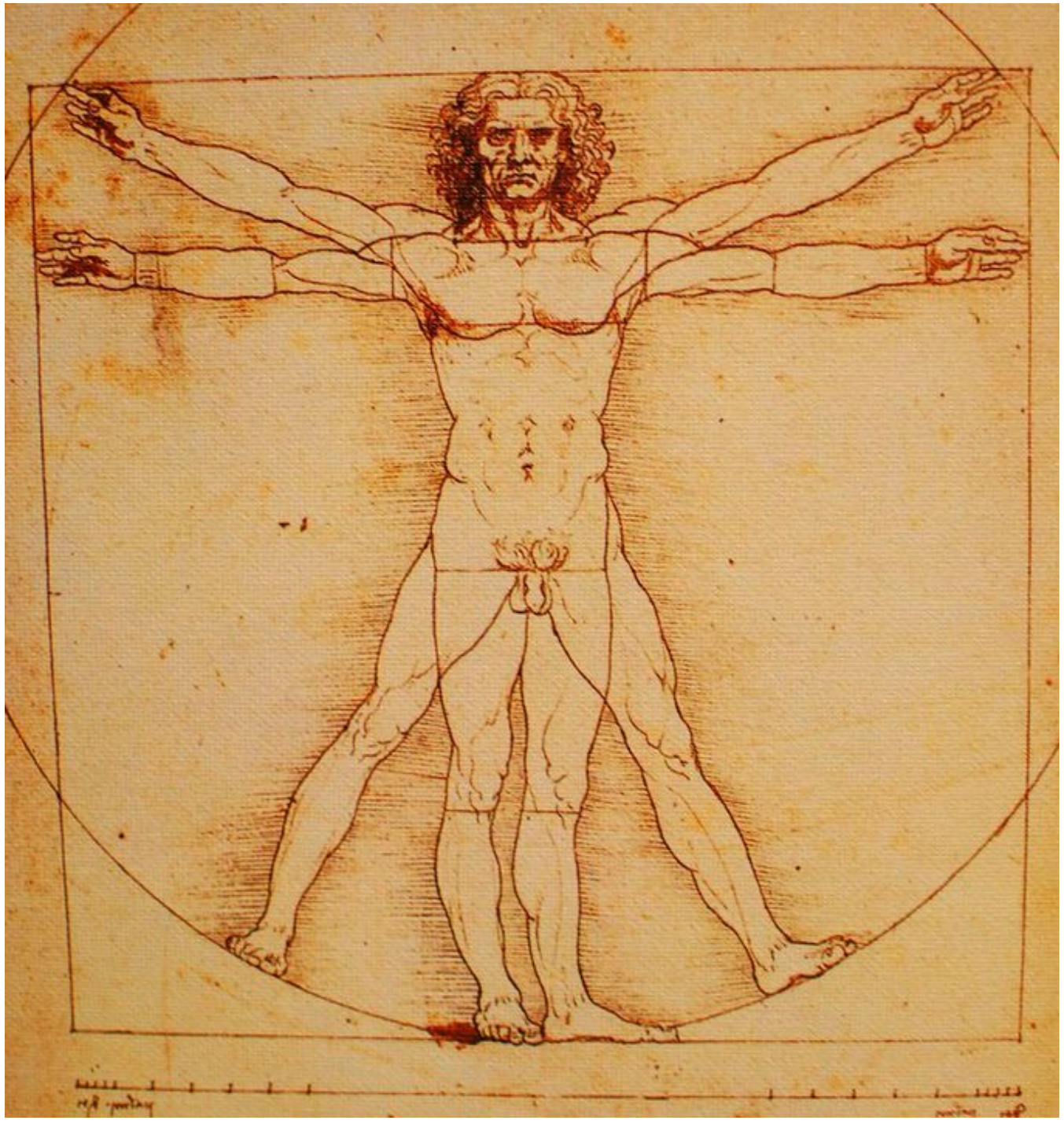
$$f(x) = x$$

x	$f(x)$
0	0
2	2




X
Y
1
2
3
4
5
2
3
4
5
6
7

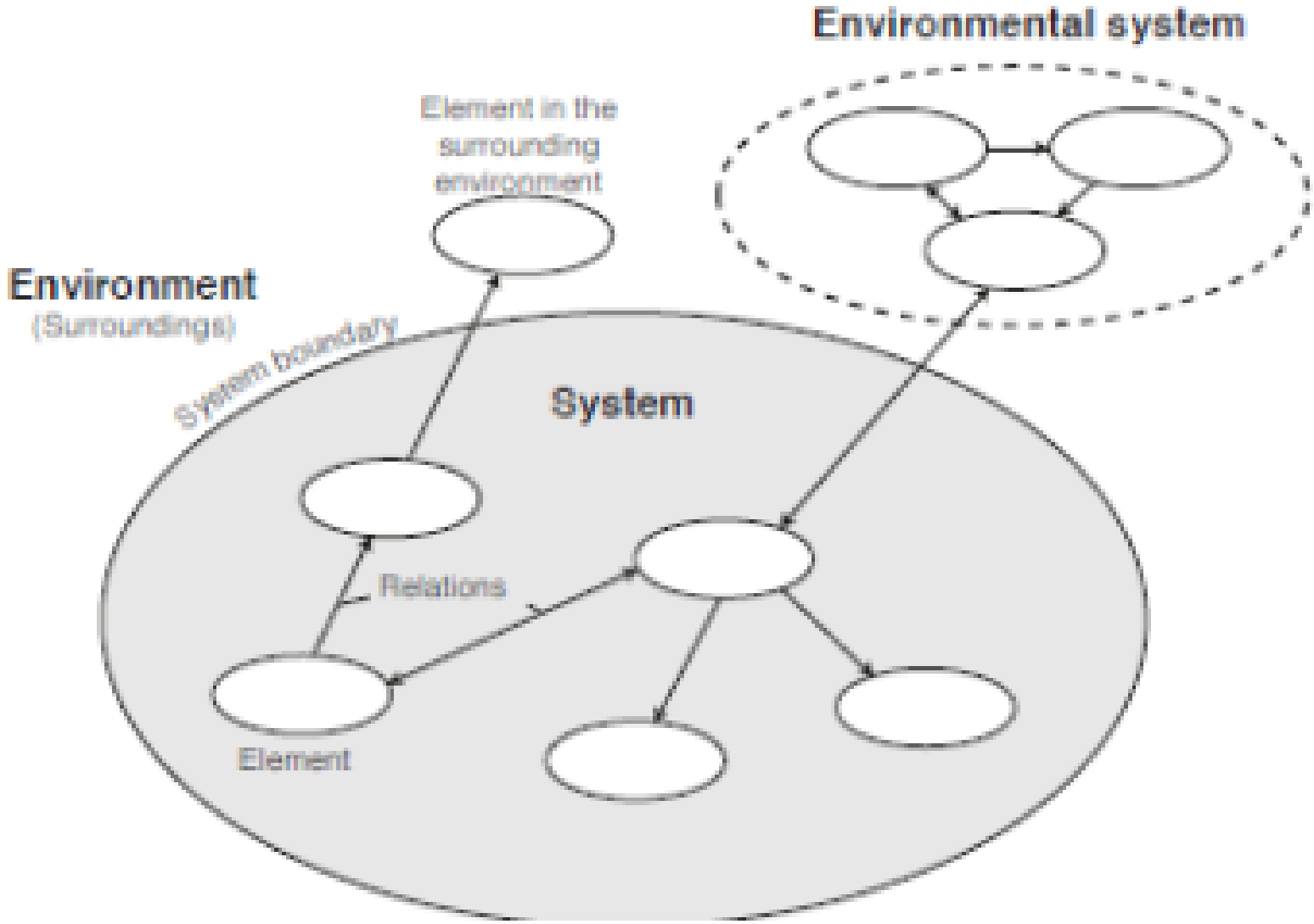





off the mark.com
by Mark Parisi
LET'S SHUT DOWN ALL HIS BODY FUNCTIONS THEN START THEM OVER AGAIN...
MARK PARISI
offthemark.com
ATLANTIC FEATURE SYND. ©MARK PARISI
WHY COMPUTER ENGINEERS SHOULD NOT BE SURGEONS

FRONTEIRA

- Entende-se por fronteira uma **borda entre o sistema e seus arredores** ou o ambiente em que está inserido.



FUNÇÕES INTERNAS

- Dentro de um sistema existem funções internas e relações entre essas funções internas.
- Juntos, eles definem a **arquitetura funcional** do sistema.
- A função, o desempenho e as "ilidades" emergem dessas funções e relacionamentos internos.

- A principal tarefa é identificar as entidades que alocam cada função interna.
-

IDENTIFICANDO A FUNÇÃO INTERNA

- Como identificamos as funções internas?
  - Existem várias abordagens, incluindo engenharia reversa da forma, projetos e metáforas.
- Nós raciocinamos, **"O que esses elementos fazem?"**
  - Muitas vezes, apenas observar as operações também é valioso.. Ao aplicar essas abordagens, devemos primeiro nos concentrar nos processos internos que levam à criação de valor.
  - Também devemos aplicar o conhecimento e a experiência do domínio.

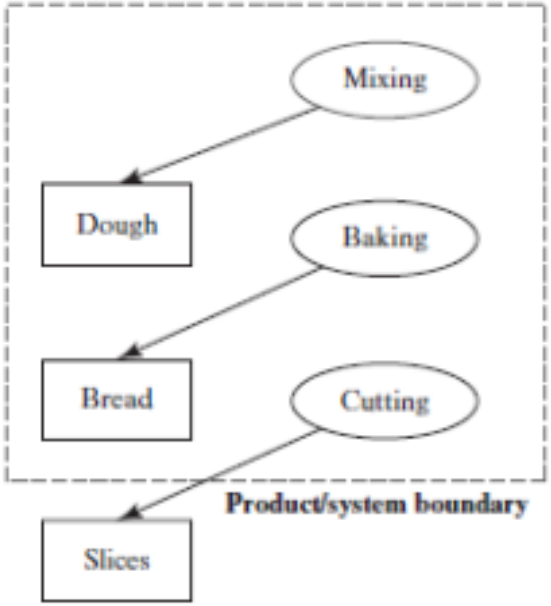


FIGURE 5.10 Internal functions of bread slice making.

FUNÇÕES EXTERNAS

- Entrega de valor
  - Primeiro, a função deve ser entregue externamente; deve cruzar a fronteira do sistema e influenciar algo no contexto.
  - **A função agrega valor quando atua externamente ao sistema**. É importante que a função seja entregue externamente, que não seja apenas interna ao sistema..
- Restrição de ambiente
  - **Elementos externos também podem restringir/conduzir o sistema**.
  - Essas funções são (geralmente) incontroláveis e são fonte de ESTADOS/MODOS para o sistema.

MODELAGEM FUNCIONAL

MODELAGEM FUNCIONAL

- A modelagem funcional em Engenharia de Sistemas é uma **representação estruturada de funções** (ou seja, atividades, ações, processos, operações) dentro do sistema modelado.
- O objetivo do modelo funcional é (i) descrever as funções e processos, (ii) auxiliar na descoberta das necessidades de informação, (iii) ajudar a identificar oportunidades e (iv) estabelecer uma base para determinar os custos de produtos e serviços.

MF: PARTICIONAMENTO FUNCIONAL

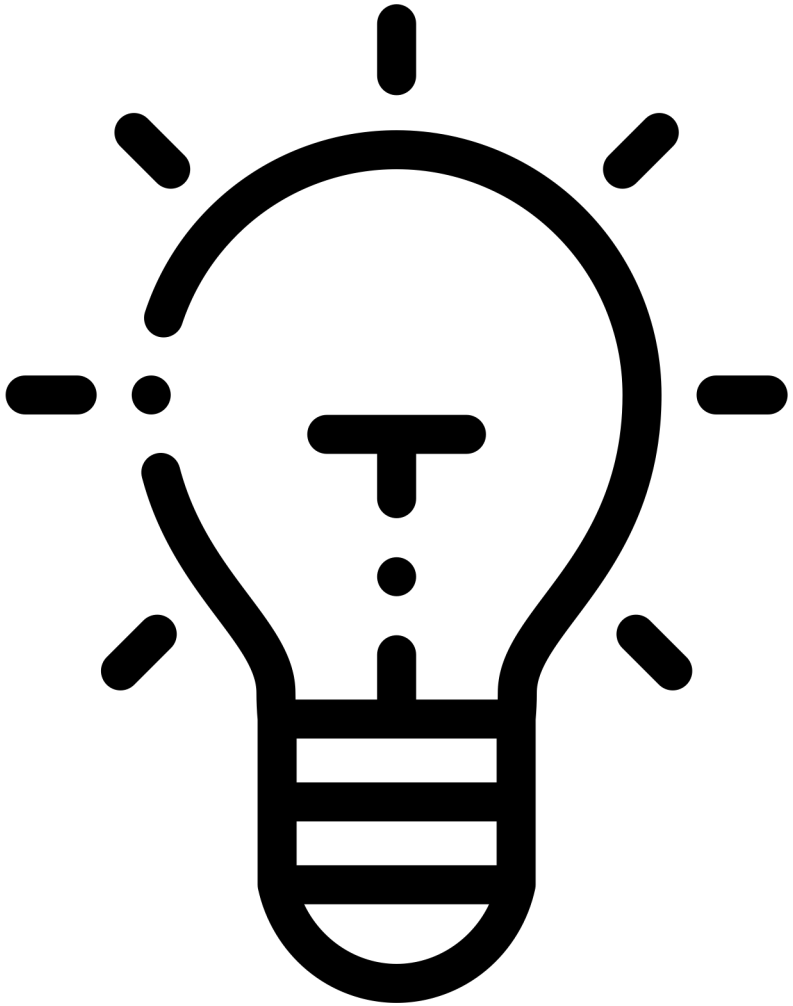
- O particionamento funcional é o processo de agrupamento de funções que se ajustam logicamente aos componentes que provavelmente serão usados e para minimizar as interfaces funcionais.
- **O particionamento é realizado como parte da decomposição funcional. Ele identifica agrupamentos lógicos de funções que facilitam o uso de componentes modulares e projetos de sistema aberto**.
- O particionamento funcional também é útil para entender como os equipamentos ou componentes existentes (incluindo comerciais) funcionarão com ou dentro do sistema..

MF: "RECONSTRUTIBILIDADE"

- A decomposição funcional refere-se ao processo de resolução de uma relação funcional em suas partes constituintes de tal forma que a **função original possa ser reconstruída a partir dessas partes.**

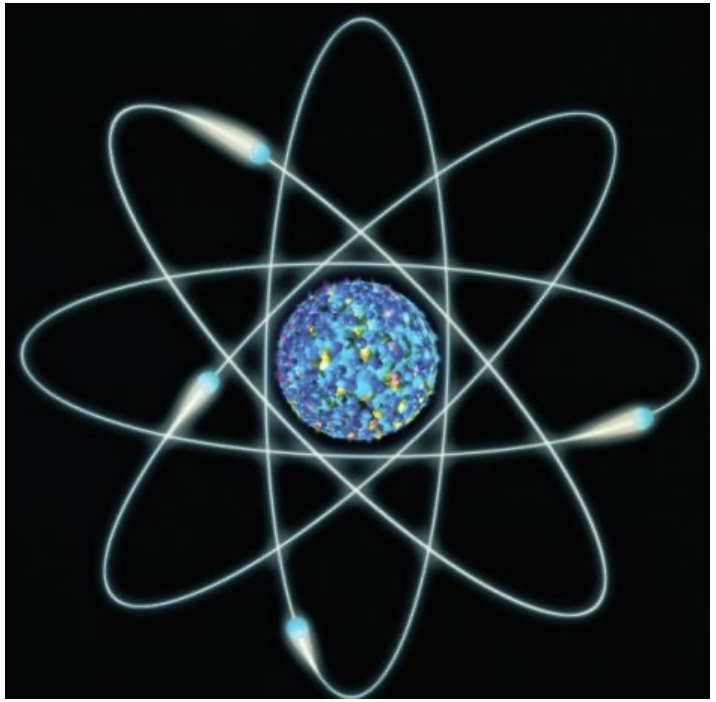
MF: ENTENDIMENTO

- Em geral, esse processo de decomposição é realizado com o propósito de obter **informações sobre a identidade dos componentes constituintes ou com o propósito de obter uma representação comprimida da função global** – uma tarefa que só é viável quando os processos constituintes possuem um certo nível de modularidade.
-



MF: FUNÇÃO ATÔMICA

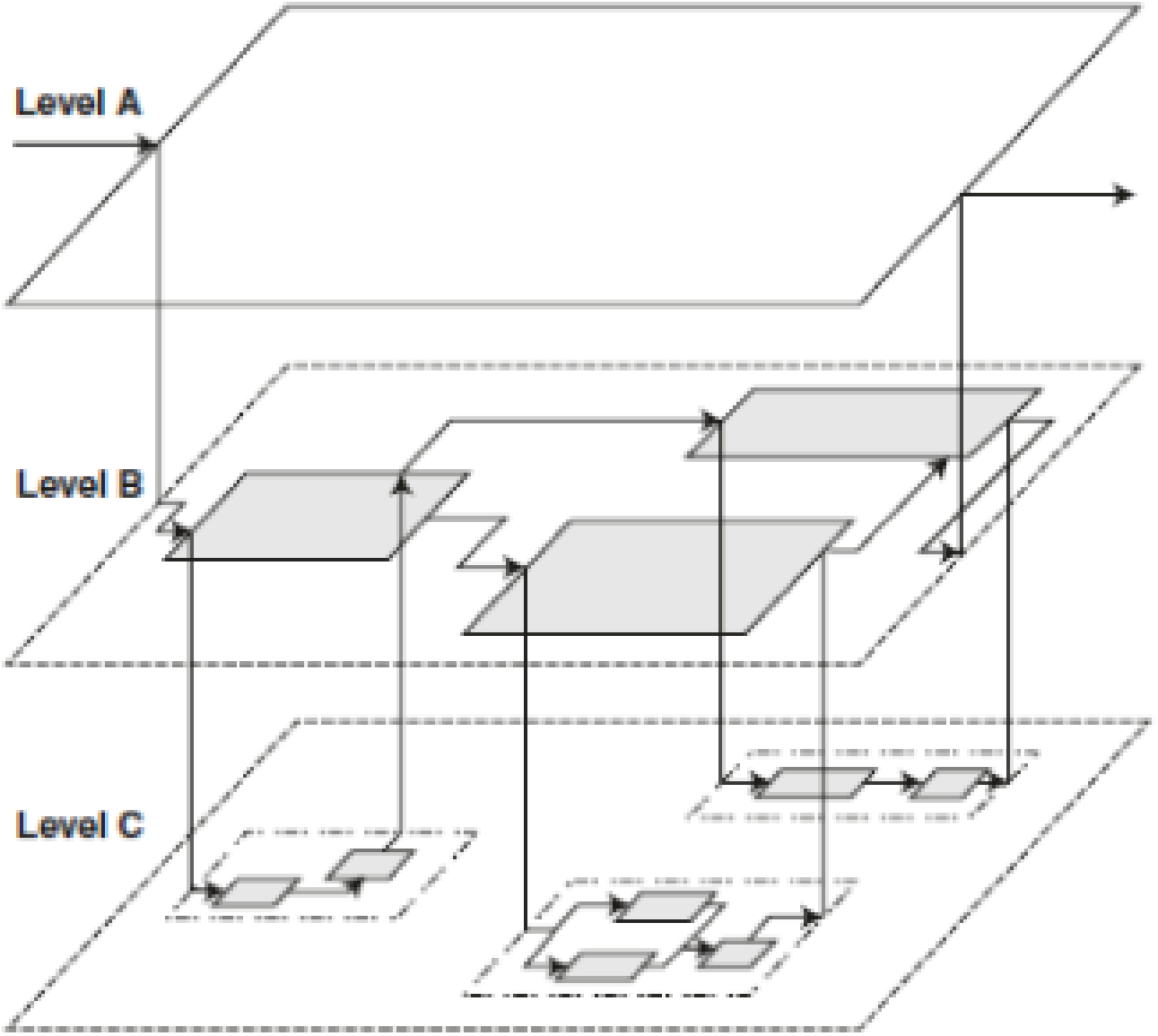
- Na decomposição funcional de sistemas, que é um método para analisar sistemas, a ideia básica é tentar **dividir um sistema de tal forma que cada bloco do diagrama de blocos possa ser descrito sem usar as palavras 'e' ou 'ou'.**
- Este exercício força cada parte do sistema a receber uma **função pura**.



MF: SIMPLICIDADE

- Quando um sistema é composto de funções puras, elas podem ser reutilizadas ou substituídas.
- Um efeito colateral usual de tal composição é que as interfaces entre blocos se tornam simples e genéricas. Como as interfaces geralmente se tornam simples, é mais fácil substituir uma função pura por uma função relacionada e semelhante.

MF: DECOMPOSIÇÃO FUNCIONAL

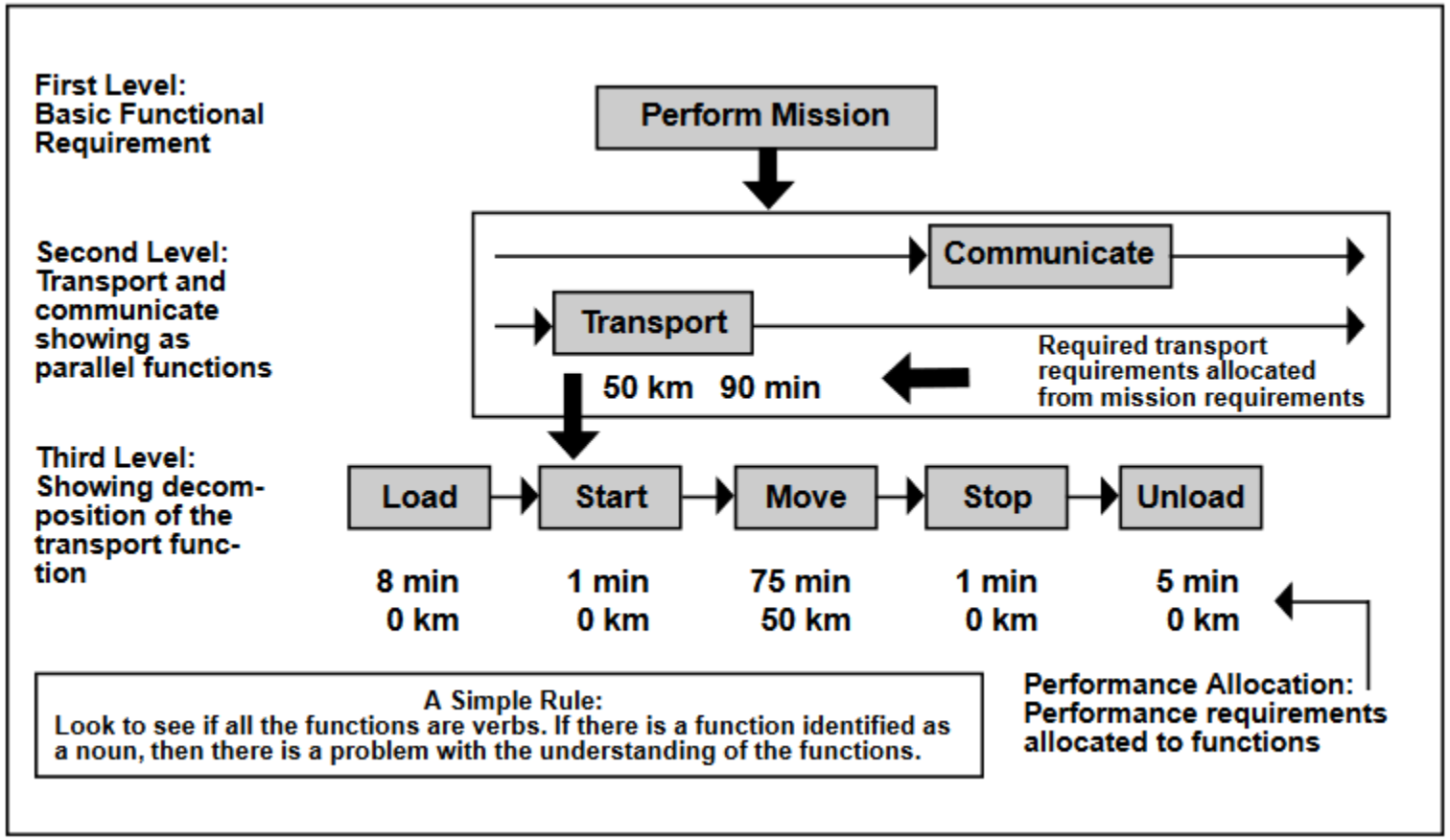


ARQUITETURA FUNCIONAL

ARQUITETURA FUNCIONAL

- A arquitetura funcional é (geralmente) uma **decomposição top-down** dos requisitos funcionais e de desempenho do sistema.
- A arquitetura mostra não apenas as funções que devem ser executadas, mas também **o sequenciamento lógico das funções** e os requisitos de desempenho associados às funções.

EXEMPLO DE ARQUITETURA FUNCIONAL

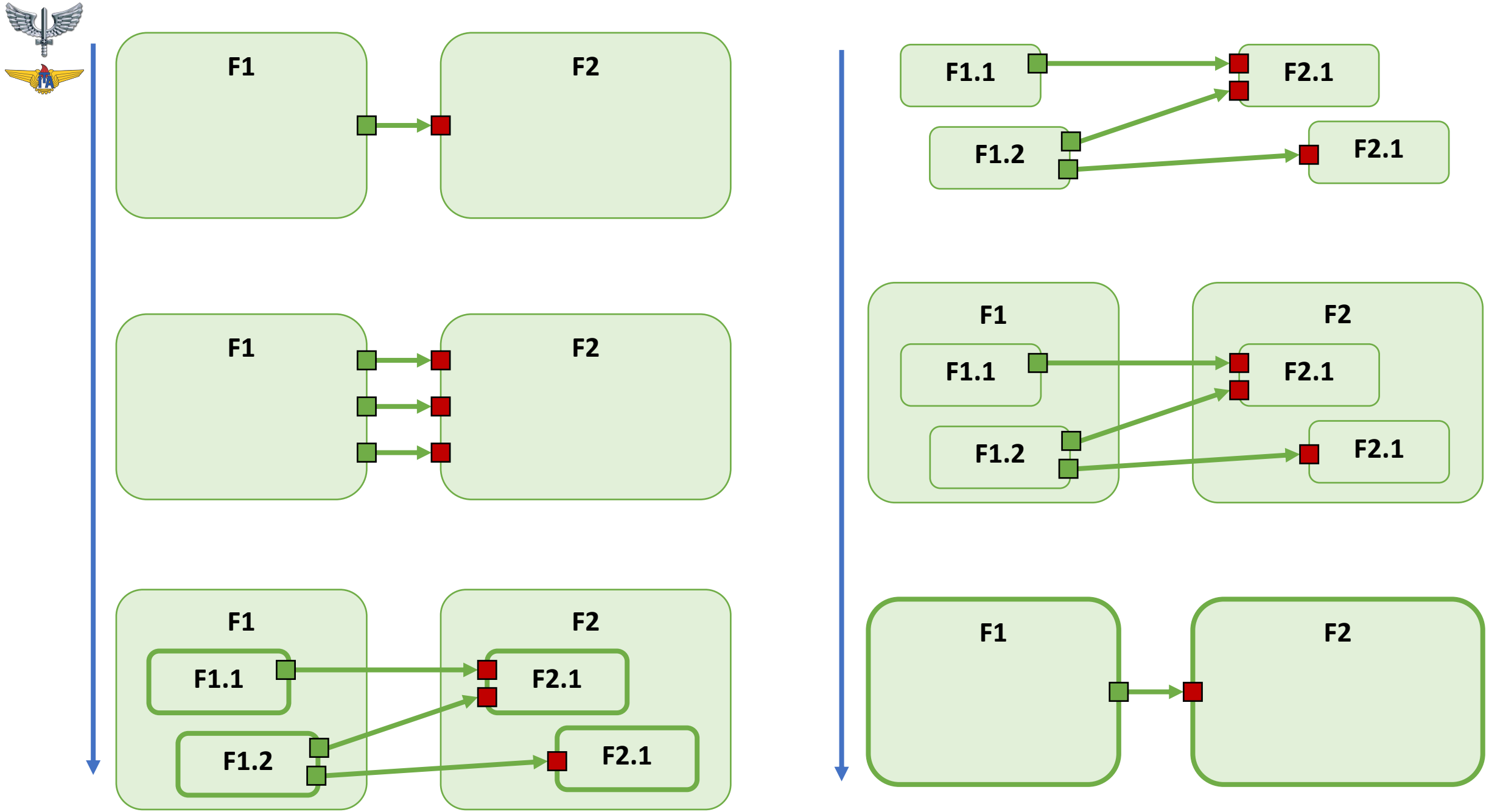


O Corpo de Fuzileiros Navais tem a necessidade de transportar tropas em unidades de nível de esquadrão a uma distância de 50 quilômetros. As tropas devem ser transportadas no prazo de 90 minutos a contar da data de chegada do sistema de transporte. A comunicação constante é necessária durante o transporte de tropas.

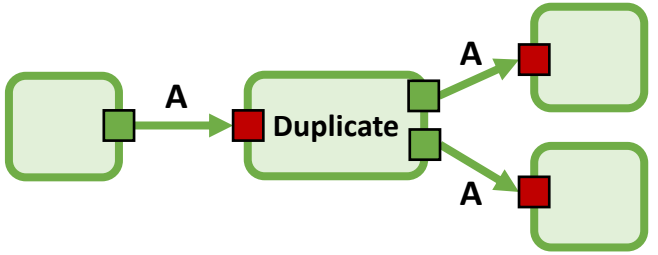
MAIS PARTICULARMENTE

FUNÇÕES

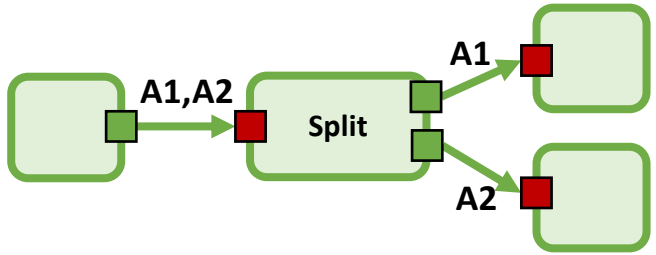
- Uma função é uma ação, uma operação ou um serviço, executada pelo sistema ou um de seus componentes, ou também por um ator interagindo com o sistema.
  - Executar uma função geralmente produz **itens de troca (exchanges)** esperados por outras funções e, para fazer isso, requer outros itens fornecidos por outras funções.
  - Várias funções podem ser agrupadas em uma função mãe (elas são então chamadas de subfunções, ou funções filhas, desta função). Simetricamente, uma função pode ser refinada em várias funções.
  - Este agrupamento não é uma forte relação de decomposição estrutural; o agrupamento de funções forma apenas uma representação sintética destes, essencialmente para fins documentais.
- Geralmente, em um modelo finalizado, apenas as funções folha (sem subfunções) referem-se e carregam a descrição funcional esperada.
- **Por convenção, uma função é nomeada com um verbo.**

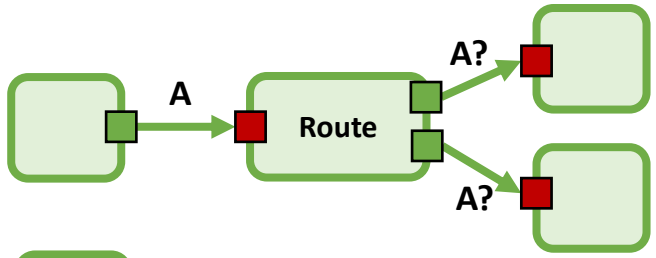

F1
F2
F1
F2
F1
F2
F1.1
F1.2
F2.1
F2.1
F1
F2
F1.1
F1.2
F2.1
F2.1
F1
F2
F1.1
F1.2
F2.1
F2.1
F1
F2

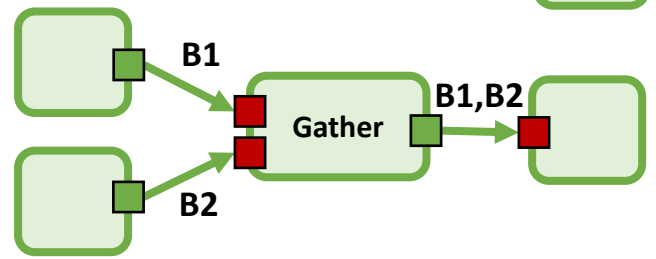
CONTROLES DE FLUXO

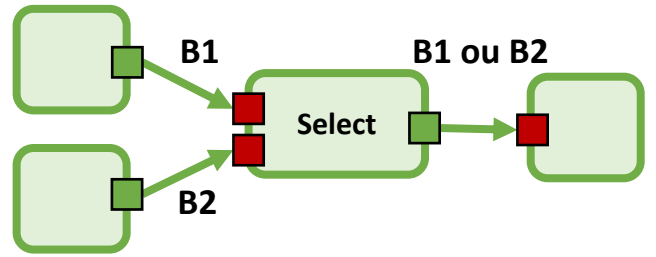


- Funções de controle de fluxo são intermediários entre a(s) fonte(s) e o(s) destinatário(s), responsáveis por controlar as condições de interação:
  - para especificar uma **difusão simultânea** de uma troca de origem para vários destinatários, definimos uma função Duplicate que **transmite os mesmos itens de troca para todos os destinatários**;
  - para especificar a **difusão simultânea** de alguns dos itens de troca para cada destinatário seletivamente, uma função **Split** que **roteia cada parte para um destinatário separado**;
  - para especificar a seleção de um entre vários destinatários potenciais, uma função **Route** que **transmite (na maioria das vezes sujeita a condições) para cada destino apenas alguns dos itens de troca recebidos**;
  - para especificar a combinação de itens de várias trocas de diferentes fontes, uma função **Gather** pode ser um **único item de troca combinando aqueles recebidos de diferentes fontes**;
  - para especificar a seleção de uma fonte entre muitas, uma função **Select** que direciona apenas os elementos provenientes da fonte selecionada (na maioria das vezes sujeita a condições)





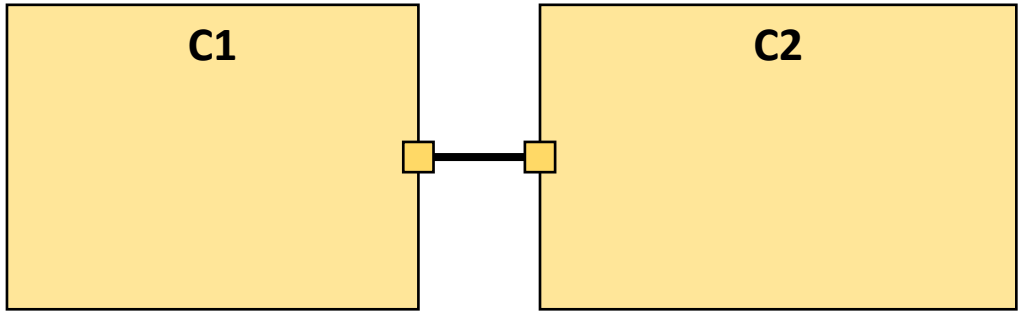


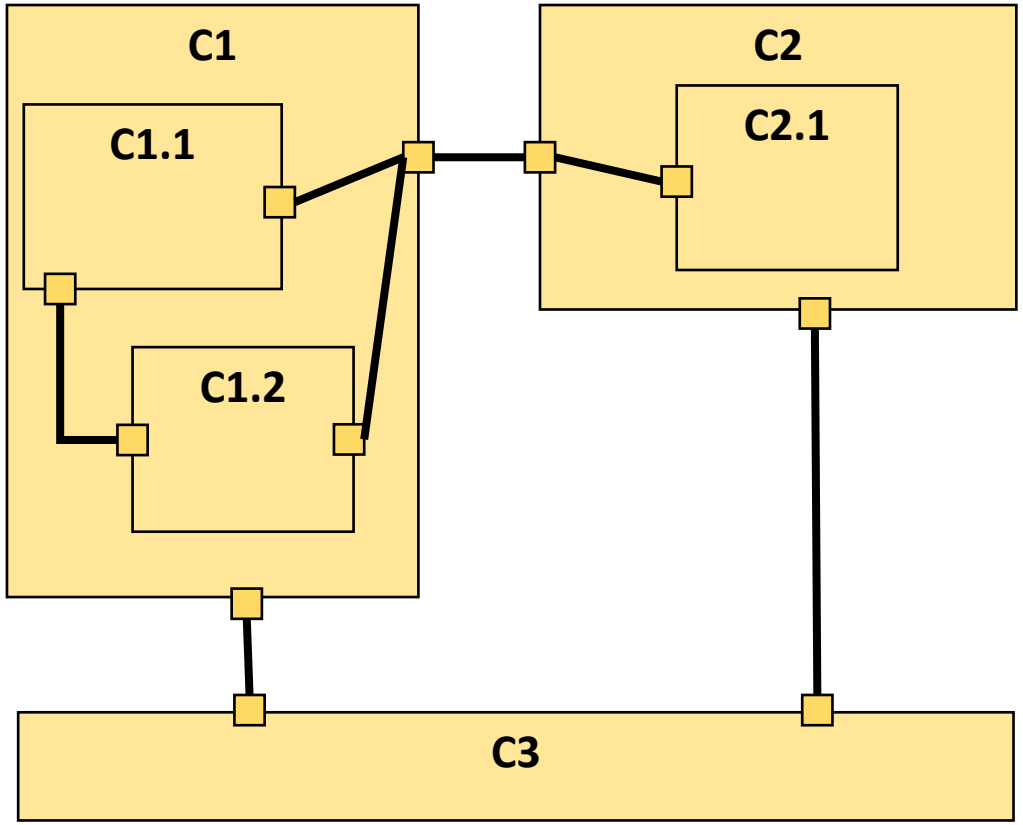


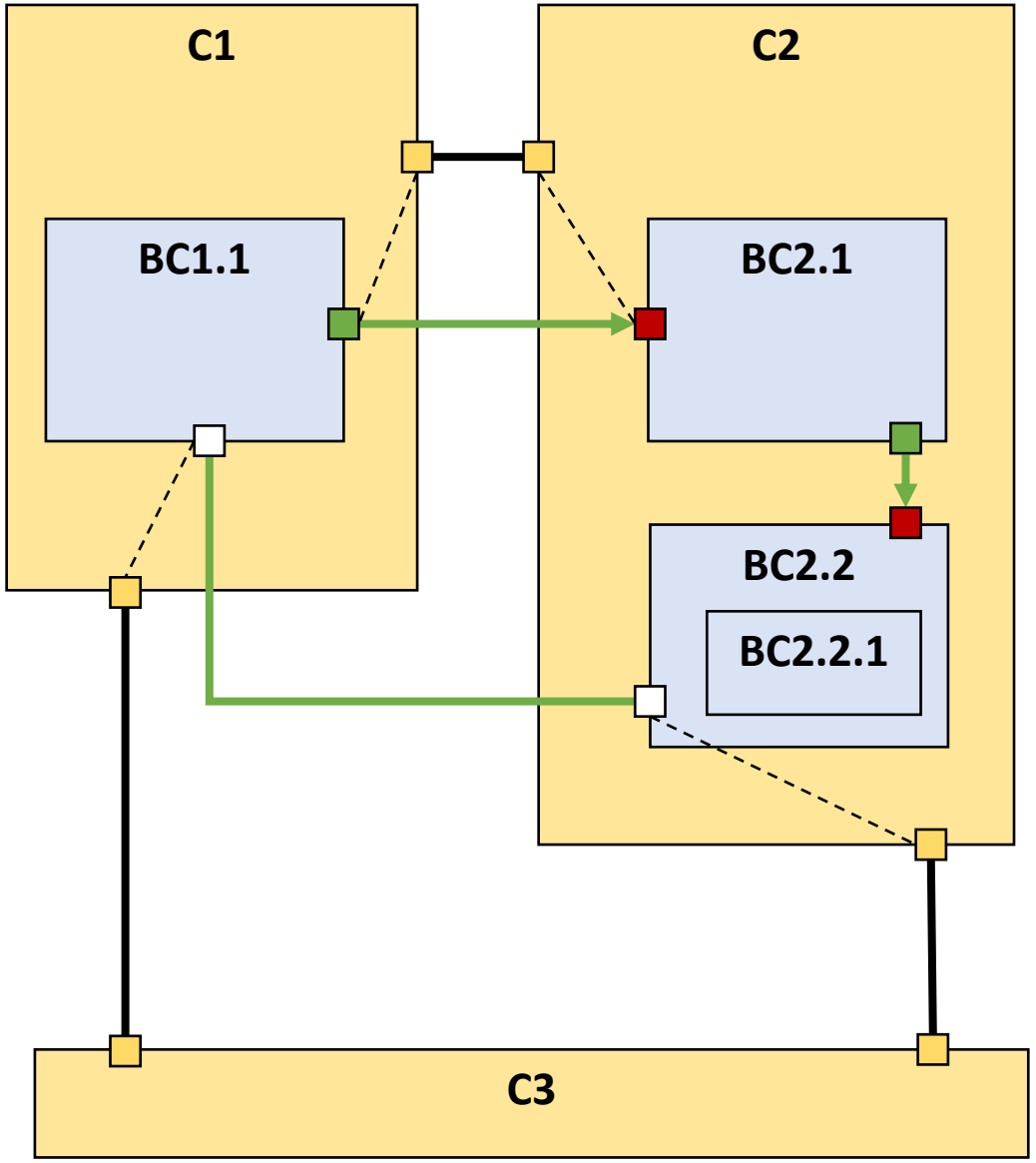
FORMA

ORGANIZAÇÃO DAS FORMAS

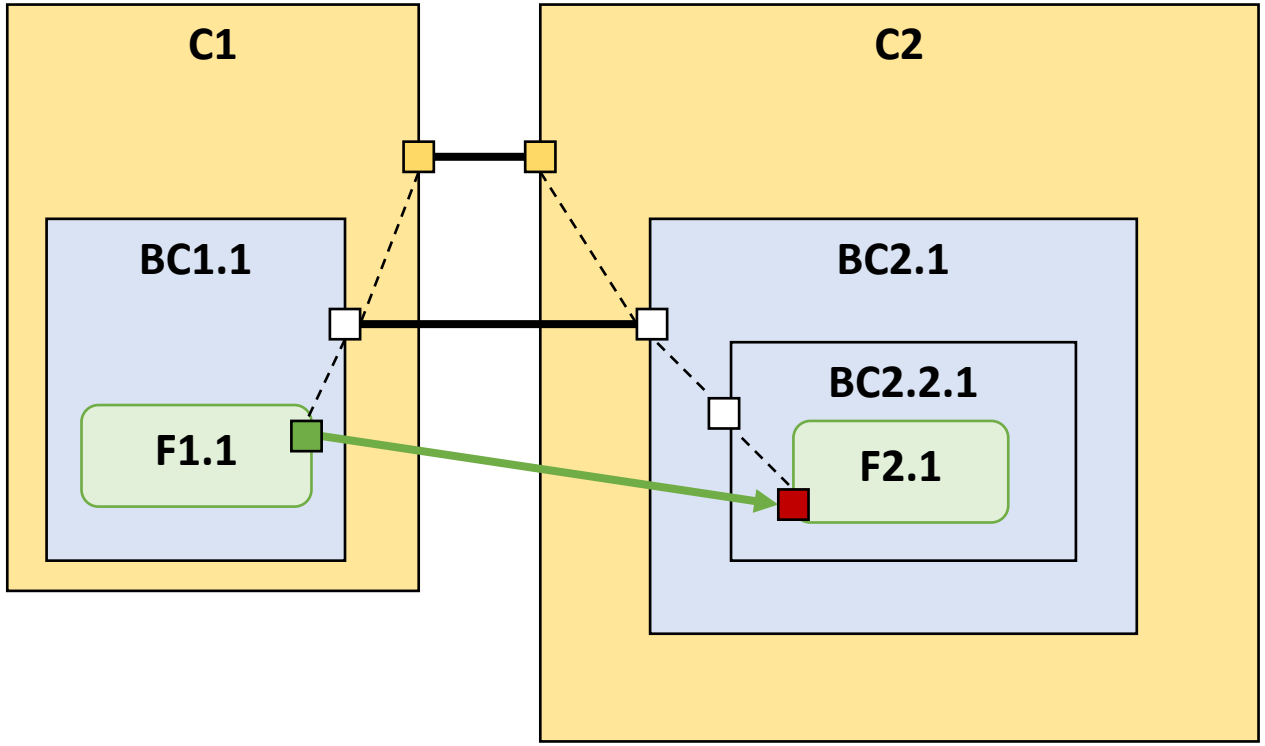
- O **sistema** é um conjunto ordenado de elementos que funcionam como um todo, respondendo à demanda e necessidades do cliente e do usuário, e sujeitos à engenharia apoiada pela Arcadia.
- Um **ator** é uma entidade que é externa ao sistema (humano ou não), interagindo com ele, especialmente através de suas interfaces.
- Um **componente** é uma parte constituinte do sistema, contribuindo para o seu comportamento e/ou propriedades, juntamente com outros componentes e atores externos ao sistema.
  - Um componente **pode ser dividido em subcomponentes**.
  - Para generalizar, um componente também pode ser alocado a um ator, para definir suas interações e conexões com o sistema ou outros atores.
- Um **componente comportamental** é um componente do sistema, responsável por realizar algumas das funções devolvidas ao sistema, interagindo com seus outros componentes comportamentais e atores externos.
- Um **componente físico** hospeda um componente comportamental, fornecendo-lhe os recursos necessários para funcionar e interagir com seu ambiente..
  - Um **componente comportamental é hospedado por um e apenas um componente físico**.


C1
C2


C1
C2
C1.1
C2.1
C1.2
C3


C1
C2
BC1.1
BC2.1
BC2.2
BC2.2.1
C3

INSTANCIANDO FUNÇÕES


C1
C2
BC1.1
BC2.1
BC2.2.1
F1.1
F2.1

UM TOSTÃO SOBRE AS TROCAS (EXCHANGES)

ITENS DE TROCAS (EXCHANGE ITENS)

- Um item de troca é um **conjunto ordenado de referências a elementos roteados** juntos, durante uma interação ou troca entre funções, componentes e atores.
- Os itens são roteados simultaneamente, nas mesmas condições, com as mesmas propriedades não funcionais. Esses itens são chamados de dados e são caracterizados pela **classe** à qual pertencem.
- **Um item de troca é definido por:**
  - **um nome;**
  - **A lista de elementos do item de troca; cada elemento é definido no item de troca por um nome, e a classe à qual ele pertence, e se a troca é bidirecional, a direção de transmissão (por convenção, "in" na direção da troca por padrão, "out" na direção oposta, ou "in/out");**
  - **a descrição das condições de comunicação, se necessário, por exemplo, serviço, mensagem, evento, fluxo de dados, dados compartilhados, fluxo de material, quantidade física, etc.**

TROCAS FUNCIONAIS

- **Pelo menos um item de troca** deve ser alocado para cada porta funcional em uma função para caracterizar o conteúdo que a função pode produzir ou que ela precisa.
  - Este item de troca **pode ser compartilhado por várias portas**.

- Se uma porta transporta **vários itens de troca, então precisamos especificar, em cada uma das trocas funcionais conectadas a ela, o(s) item(ns) realmente roteado(s),** que deve ser coerente com os das portas conectadas pela troca. Além disso, por conveniência, é possível começar alocando um item para uma troca, antes de propagá-lo para as portas conectadas a ele.

- **Recomenda-se definir apenas um único item em cada troca funcional**.

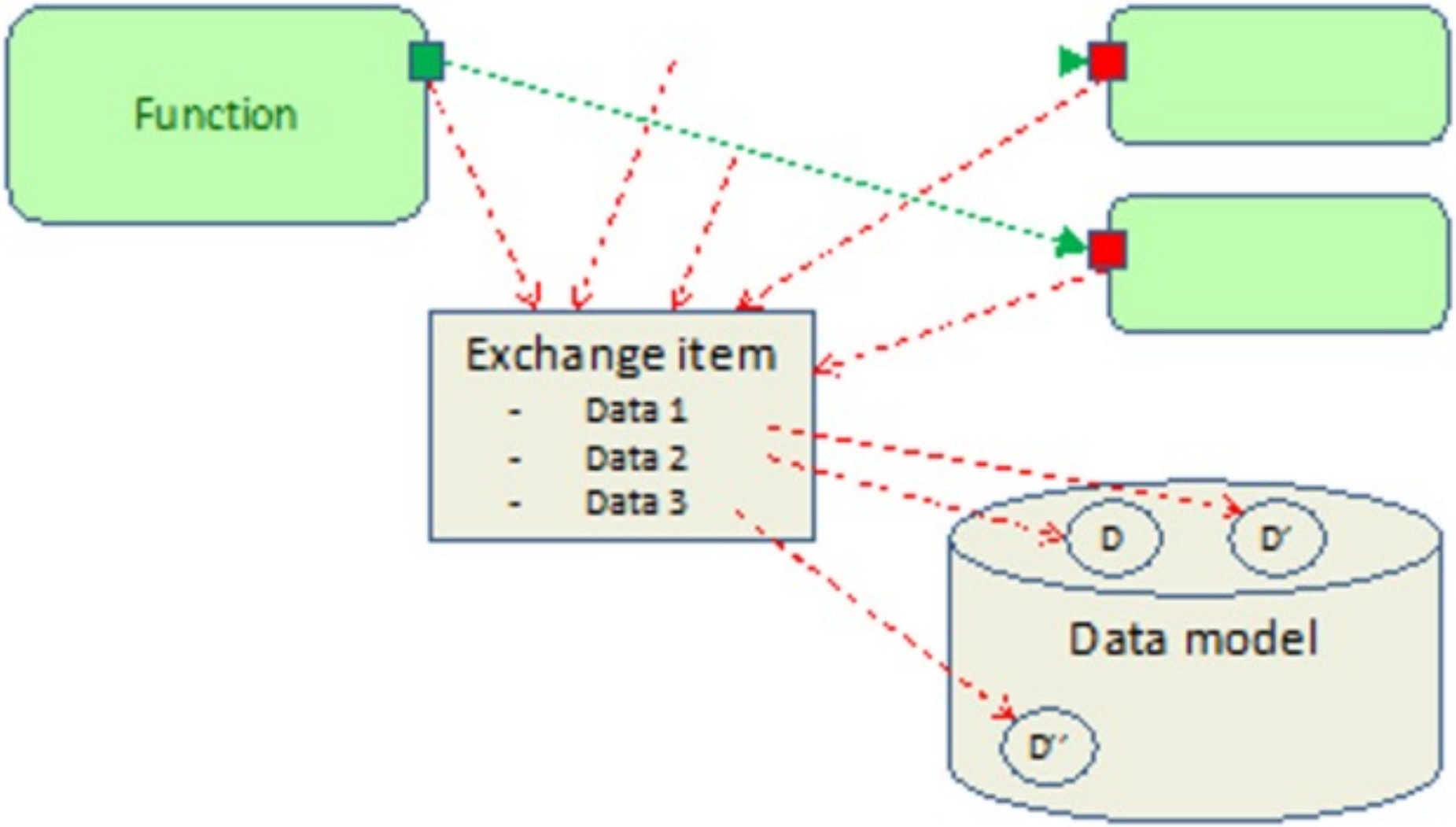
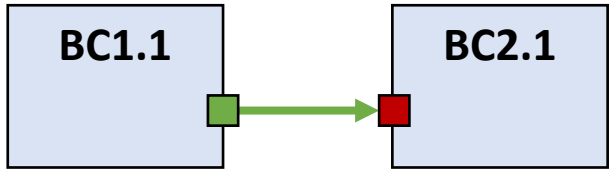


Figure 21.2. *Allocation of exchange items to functional ports and exchanges*

TROCAS ENTRE COMPONENTES COMPORTAMENTAIS (CONCEITUAIS)



- O conteúdo de uma troca entre componentes comportamentais já é definido pelos itens de troca transportados pelas trocas funcionais que ela implementa.

- **Uma interface** é um conjunto de itens de troca que permite que dois componentes (e o sistema e os atores), se comuniquem, de acordo com um "contrato" de comunicação compartilhado entre eles..

- **Várias interfaces podem ser agrupadas em uma única interface.**

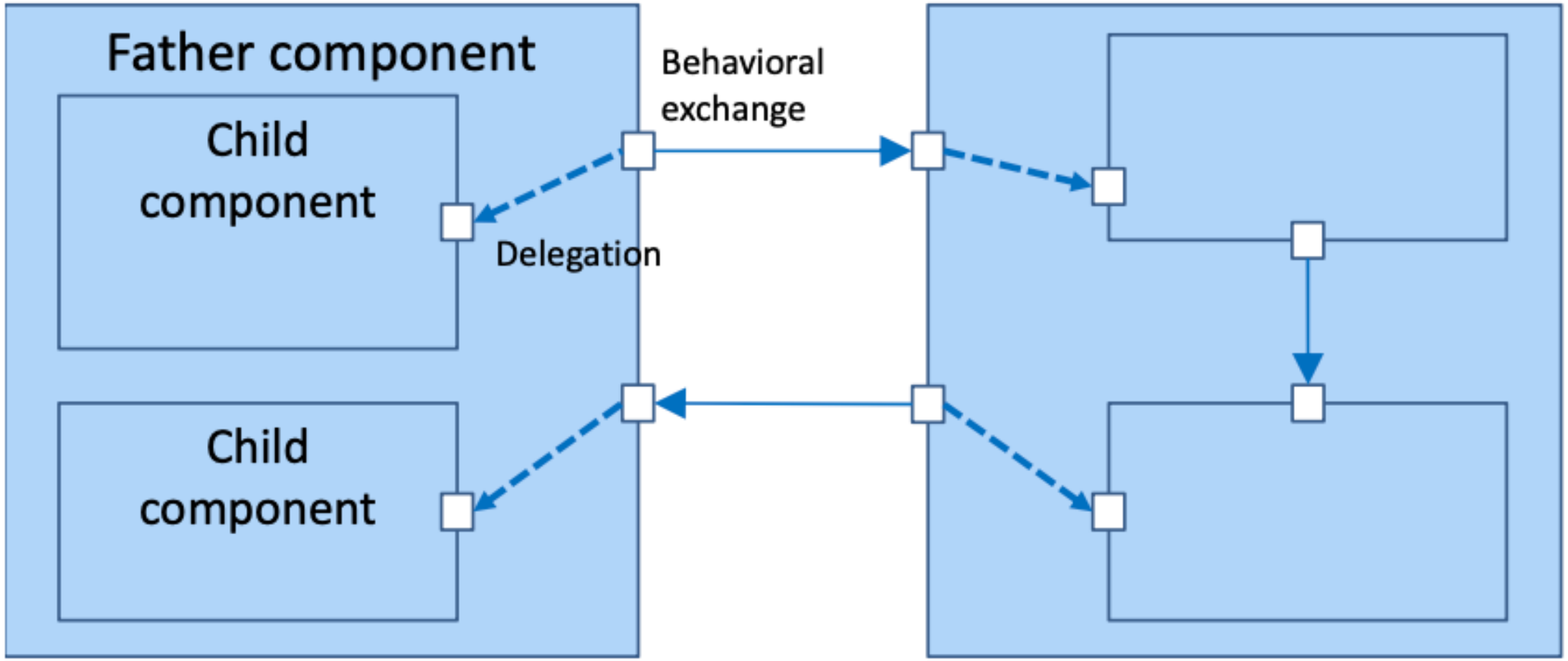


Figure 19.2. *Behavioral components, ports, exchanges and delegation*

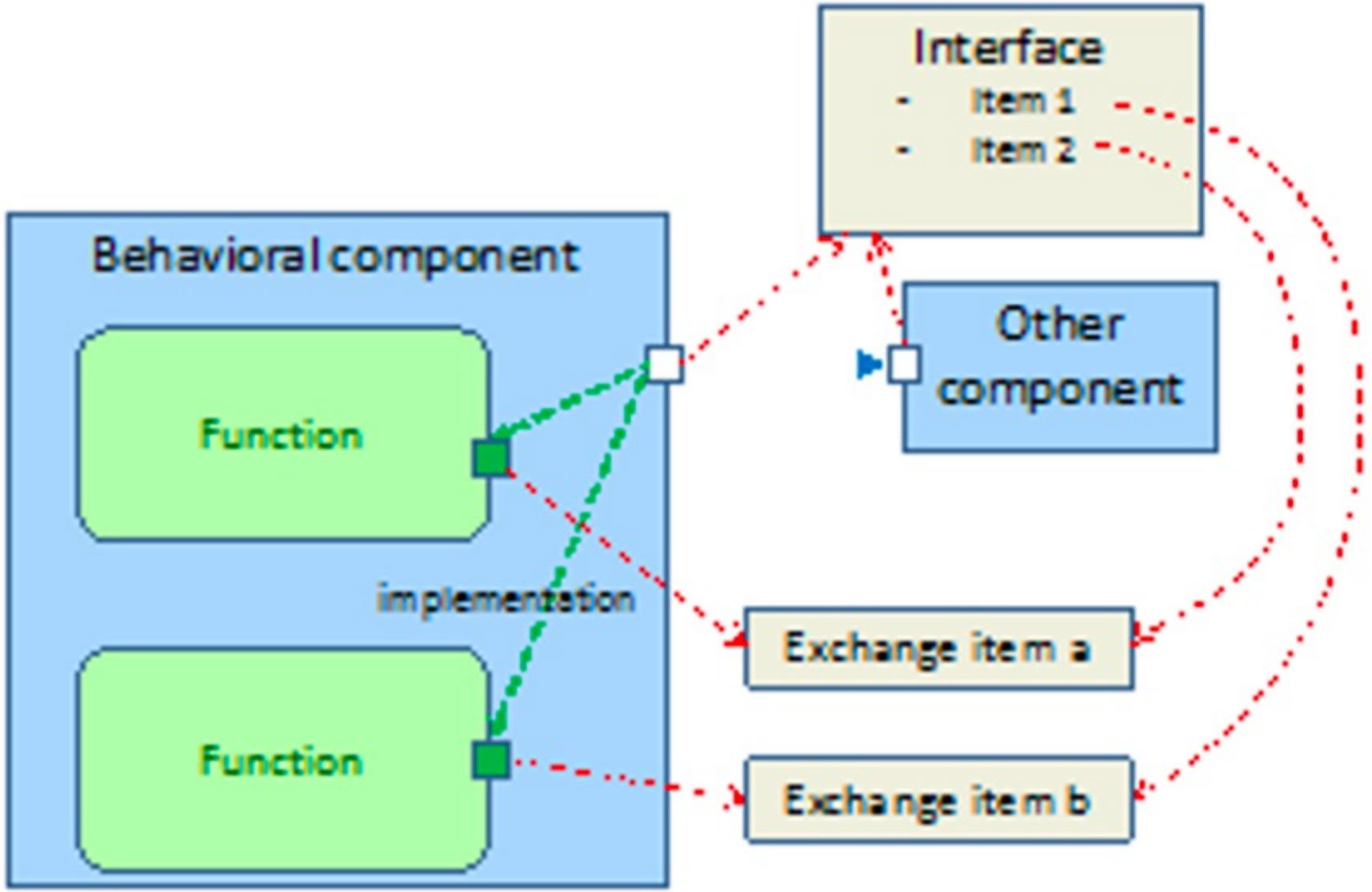
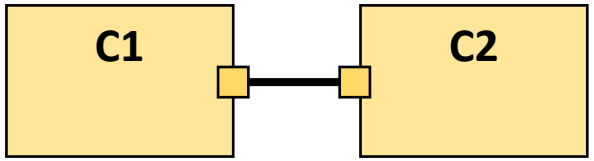


Figure 21.3. *Links between exchange elements involved in the functional and structural description*

TROCAS ENTRE COMPONENTES FÍSICOS



- Um **link físico** é um meio de comunicação, transporte ou roteamento entre dois componentes físicos de hospedagem, usado como **suporte para trocas comportamentais**.

- Um link físico (interface) vincula duas portas físicas de dois componentes. Ele pode vincular diretamente alguns subcomponentes uns aos outros (através de suas portas).

- **Um link físico faz referência às trocas comportamentais que transporta**.

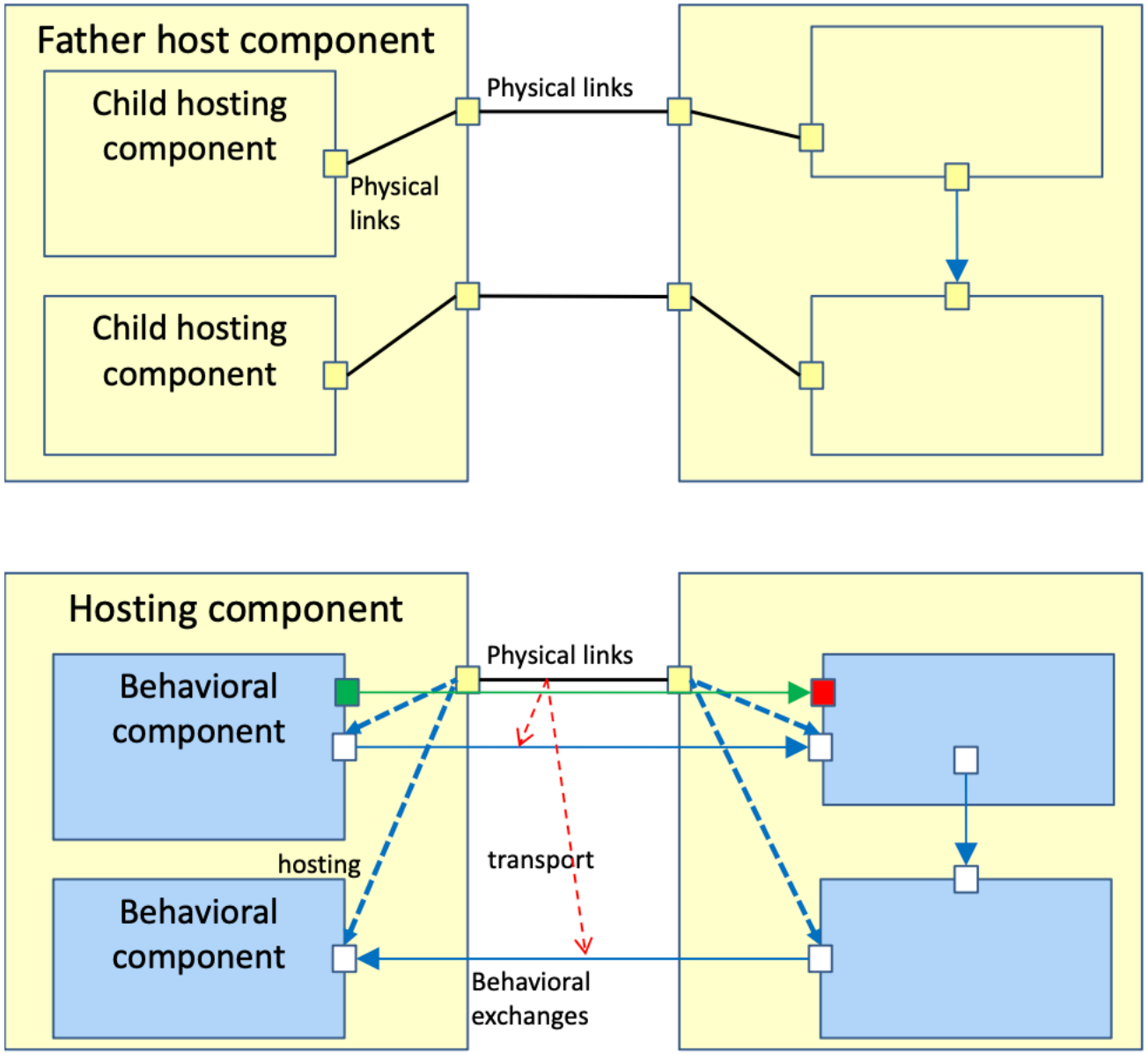


Figure 19.3. *Breakdown of a hosting component and hosting of behavioral components*

FUNÇÕES: COESÃO E ACOPLAMENTO

Um dos critérios mais importantes para julgar um design:

Coupling (acoplamento) e **cohesion (coesão)**

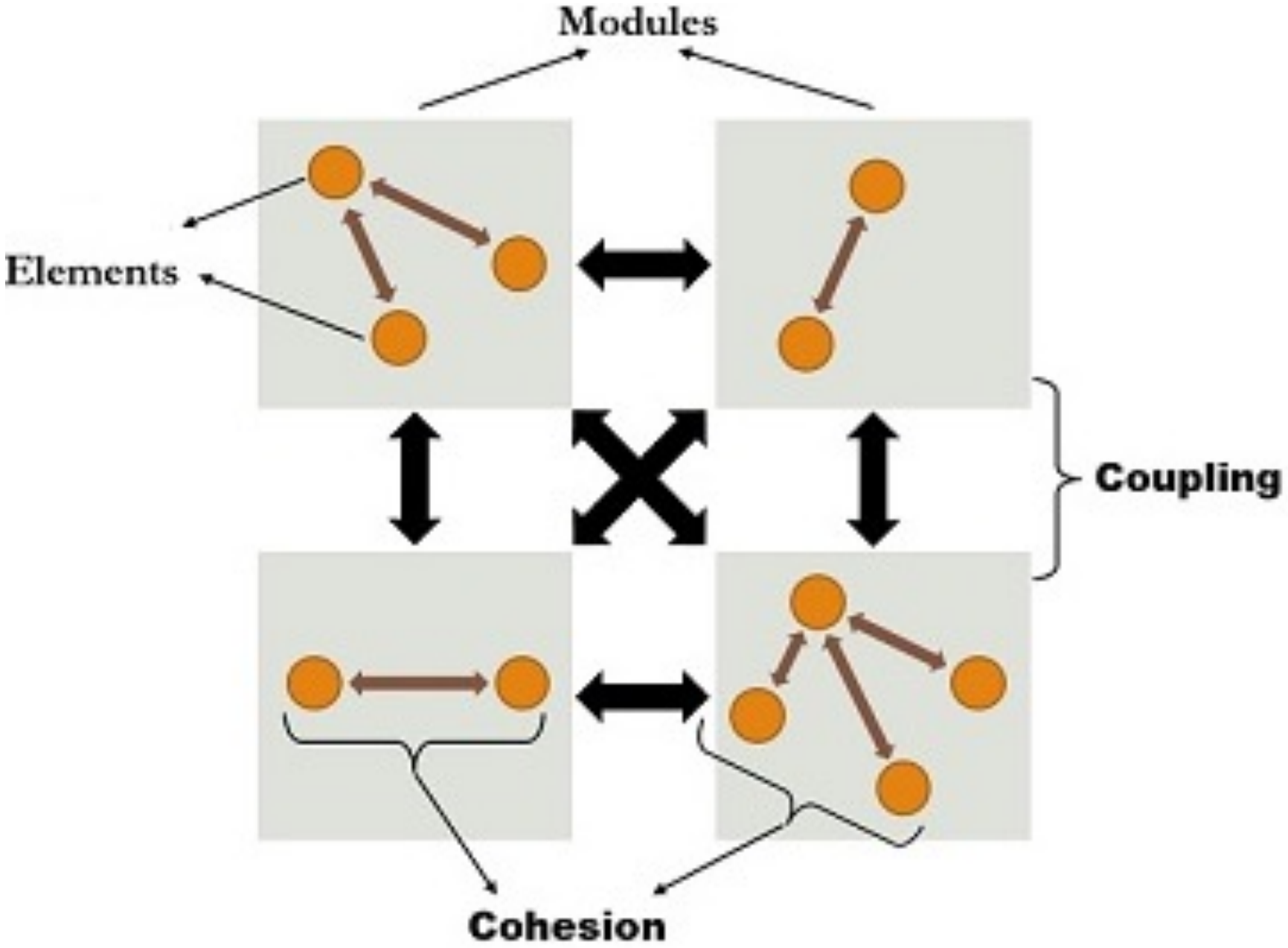
juntos, esses dois conceitos formam a teoria central do design.

Coupling and cohesion

Coupling and cohesion are outcomes of structured design, and are largely credited to the work of Larry Constantine, a distinguished researcher at MIT and IBM.

Constantine formulated the metrics of coupling and cohesion in the late 1960s, presenting his work in 1968 to the National Symposium on Modular Programming.[12]

Laplante, P. A. and Ovaska, S. J. Real-Time Systems Design and Analysis. 4th Ed. 2012.
John C. Goodpasture. Project Management the Agile Way: Making it Work in the Enterprise. 2010.


Modules
Elements
Coupling
Cohesion

DEFINIÇÃO DE COESÃO

- **Coesão é a avaliação da força funcional relacionada de um módulo**. Em palavras simples, é a medida da quantidade de interação **dentro do módulo**.

- **É basicamente a interação das funções dentro de um módulo.**

- Coesão é o aprimoramento no conceito de ocultação/encapsulamento de informações, que descreve que os módulos devem ser especificados e projetados de tal forma que os outros módulos não possam acessar as informações contidas nos módulos.

DEFINIÇÃO DE ACOPLAMENTO

- **Acoplamento é o termo usado para representar a interdependência entre os módulos**.

- Se uma **enorme quantidade de dados** é trocada entre os módulos, então eles são considerados interdependentes.

- O acoplamento entre os módulos depende da **complexidade** da interface.

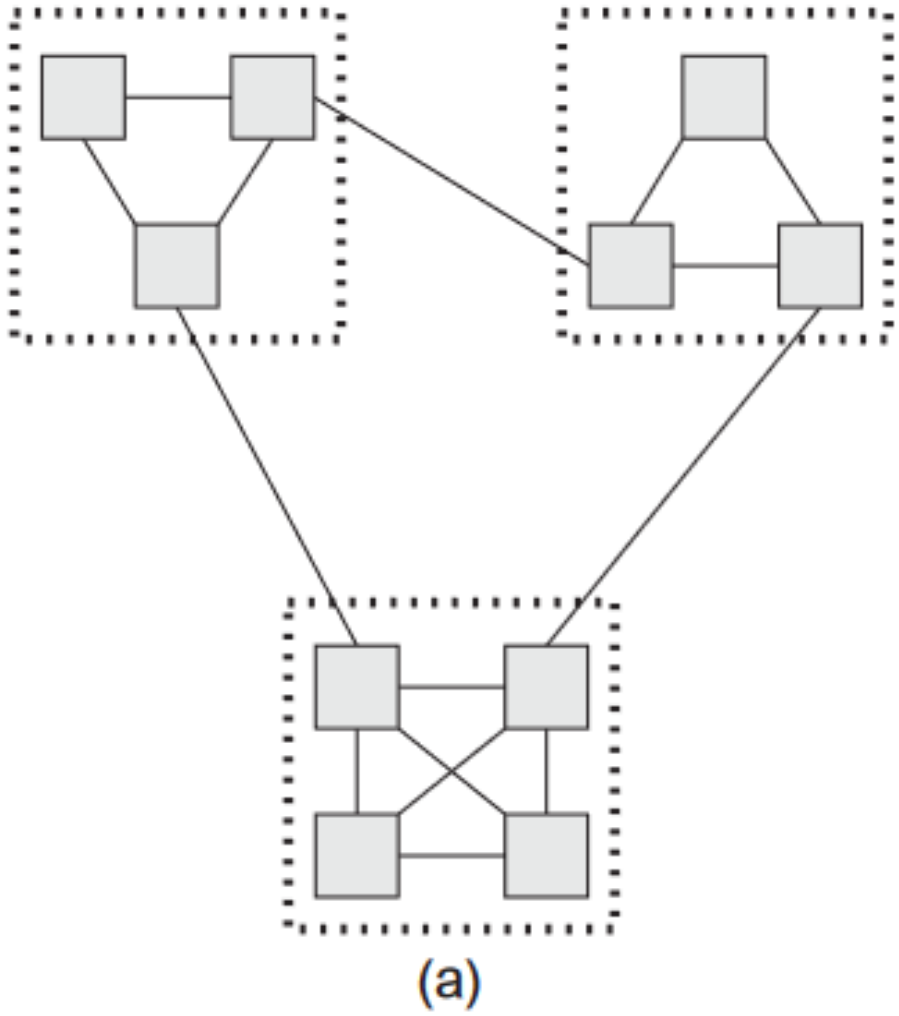
DIFERENÇA ENTRE COESÃO E ACOPLAMENTO

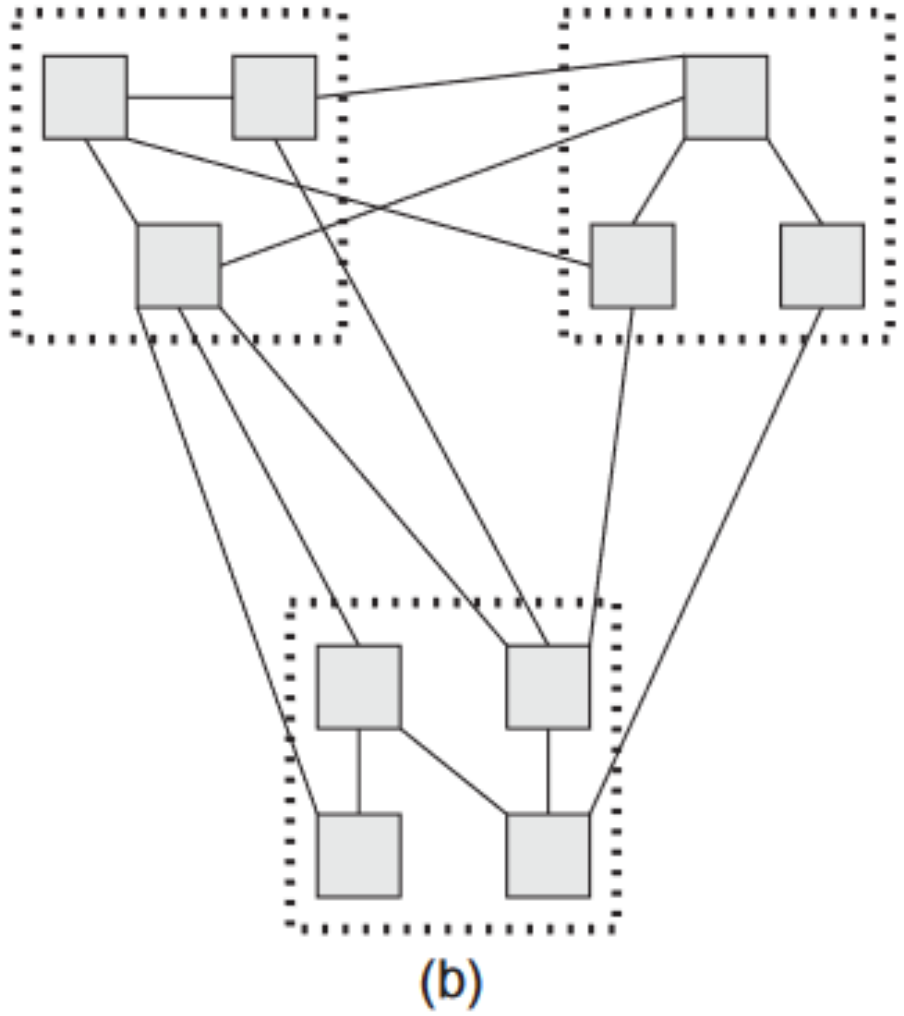
- **Coesão e acoplamento são usados como o método de categorização para especificar as interações dentro e entre os elementos do sistema.**

- Por exemplo, na engenharia de software, sabemos que o software pode conter milhares de linhas de código ou até mais do que isso, o que poderia resultar em um aumento imprevisto na complexidade do código. Para estabilizar a complexidade do código, o conceito de modularização foi introduzido.

- **A coesão é usada para descrever a interação dentro dos módulos, enquanto o acoplamento especifica as interações entre os módulos.**

Estruturas de sistemas com (a) alta coesão e baixo acoplamento, e (b) baixa coesão e alto acoplamento.





Laplante, P. A. and Ovaska, S. J. Real-Time Systems Design and Analysis. 4th Ed. 2012.

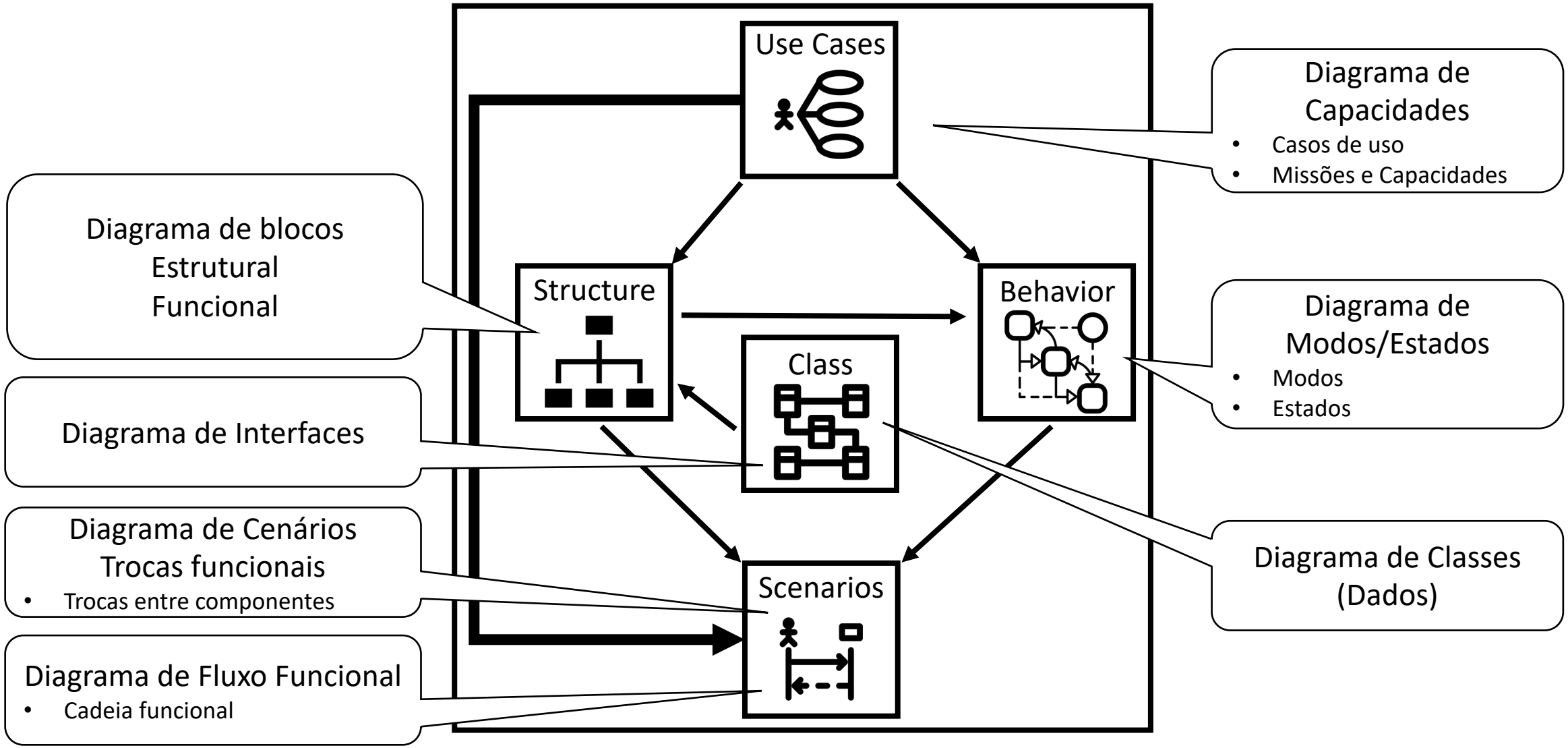
- Quando um grande sistema é decomposto em entidades menores, **é inevitável que essas entidades interajam umas com as outras**.

1. Se os **limites (boundaries) dessas entidades tiverem sido mal identificados**, as entidades **dependerão fortemente** e frequentemente interagirão umas com as outras.

2. E em um design ruim, também pode acontecer que propriedades e funções dentro de uma entidade **executem diversas tarefas** e, portanto, **não pareçam pertencer juntas**.

LINGUAGEM “ATUAL”

ROTEIRO DOS DIAGRAMAS



BLOCK DIAGRAMS

MODELING STRUCTURE WITH BLOCKS – CHAPTER 7

BLOCK DIAGRAM INTRODUCTION

- **The block is the modular unit of structure in SysML** that is used to define a type of system, component, component interconnection, or item that flows through the system, as well as external entities, conceptual entities, or other logical abstractions. A block describes a set of **instances** that share the block's definition. **A block is defined by its features, which may be subdivided into structural features and behavioral features.**
- The **block definition diagram (bdd)** is used to define blocks in terms of their **features and their structural relationships** with other blocks. The complete header for a block definition diagram is as follows:

  bdd [model element kind] model element name [diagram name]

- The **internal block diagram or ibd** resembles a traditional system block diagram and shows the connections between parts of a block. The internal block diagram header is depicted as follows:

  ibd [block] block name [diagram name]

EXAMPLE BLOCK DEFINITION DIAGRAM

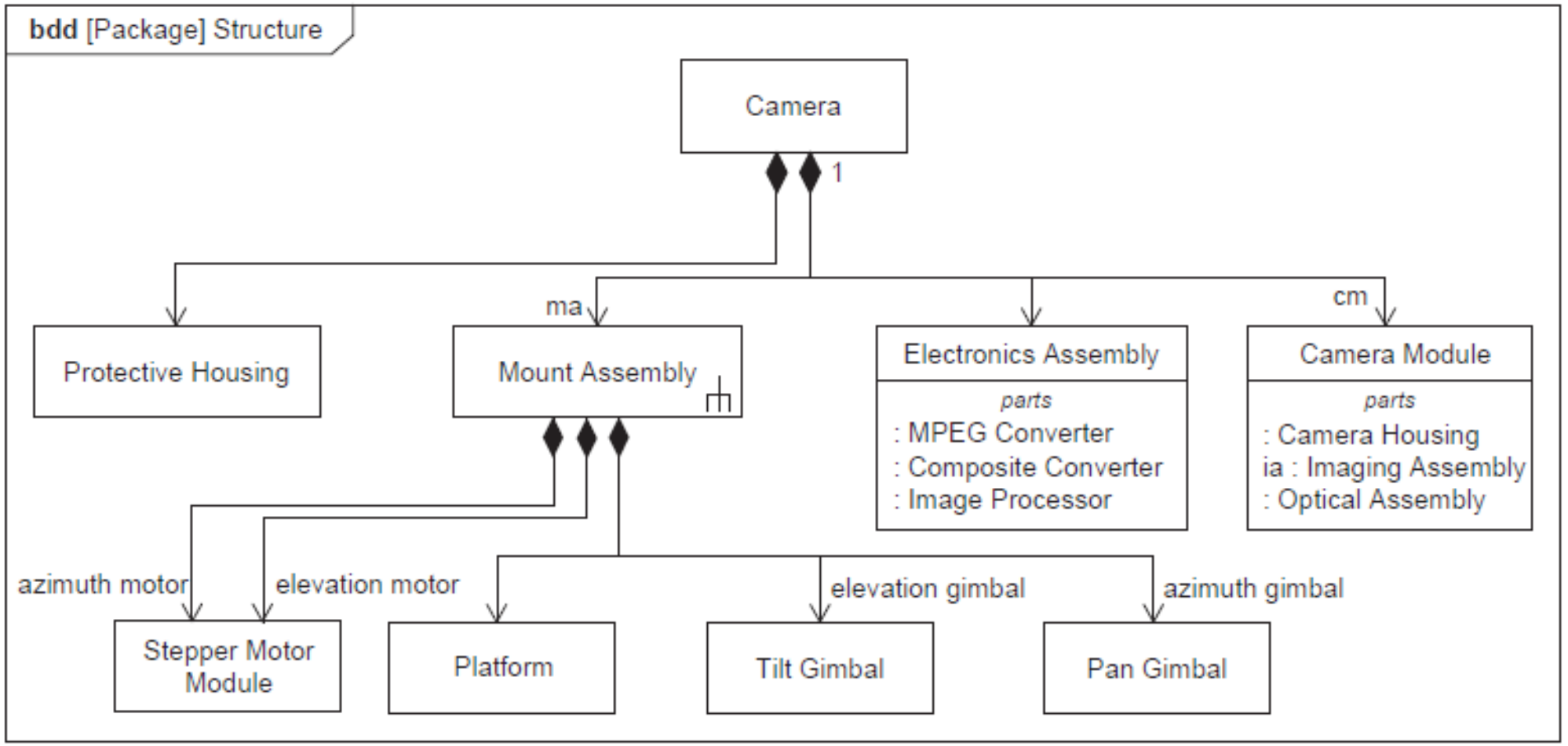


FIGURE 7.1

Example block definition diagram.

EXAMPLE INTERNAL BLOCK DIAGRAM

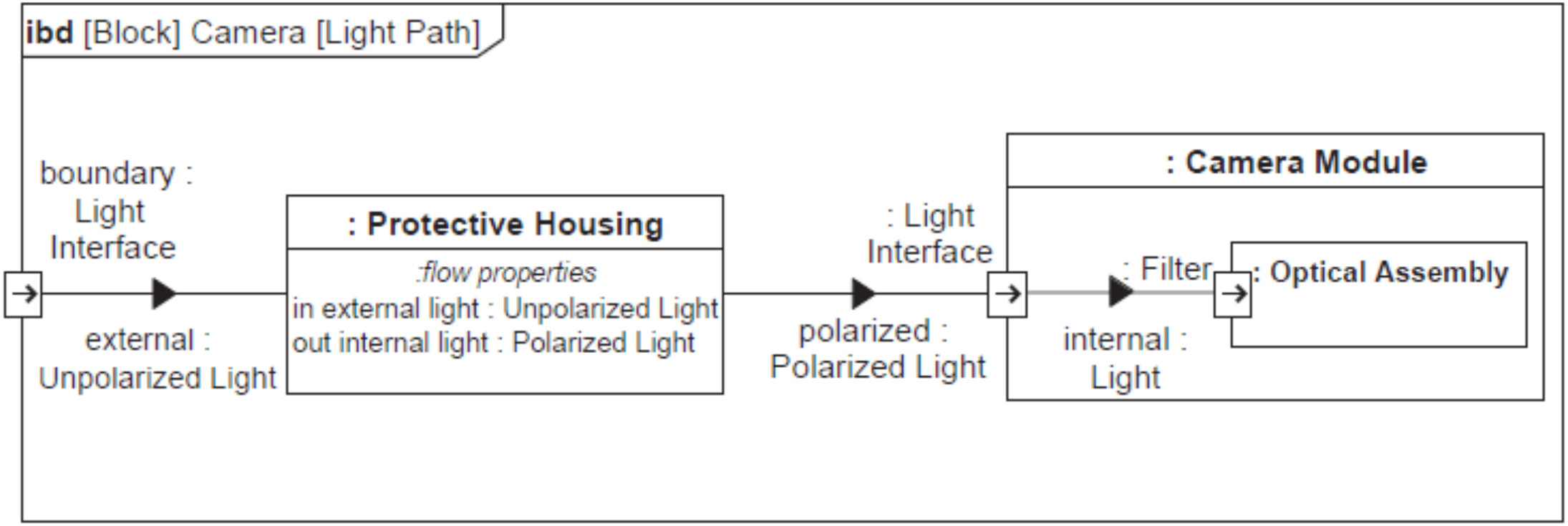
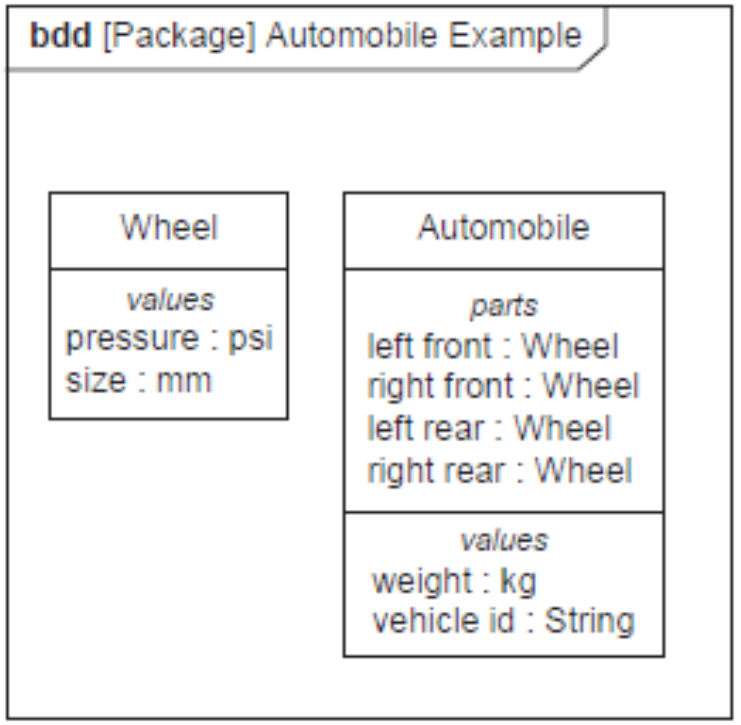


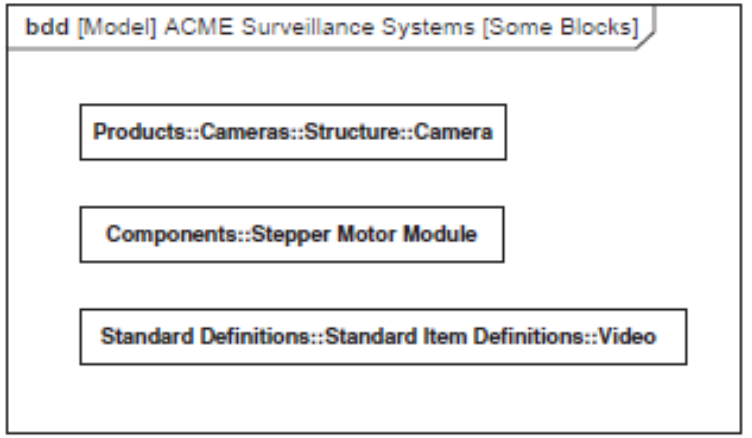
FIGURE 7.2

Example internal block diagram.

BLOCKS

- The block is the **fundamental modular unit** for describing system structure in SysML.
- **The block symbol is a rectangle that is segmented into a series of compartments**. The name compartment appears at the top of the symbol and is the only mandatory compartment.
- Other kinds of block features—such as parts, operations, value properties, and ports—can be presented in other **compartments** of the block symbol.
  - All compartments, apart from the name compartment, have labels that indicate the kind of feature they contain. *The labels are depicted in lower case italics, are plural, and include spaces between words.*

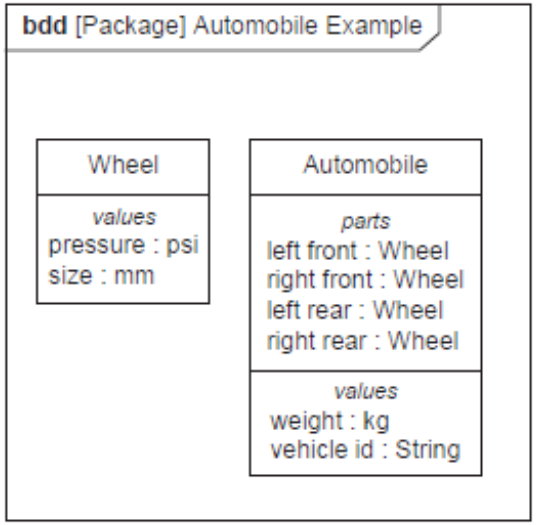




PROPERTIES

- Properties are structural features of a block. A property has a type that defines its characteristics, which may be another block, or some more basic type such as an integer.
  - **Part** properties (parts for short) describe the decomposition of a block into its constituent elements.
  - **Reference** properties whose values refer to parts of other blocks.
  - **Value** properties describe the quantifiable characteristics of a block, such as its weight or velocity.
- The properties compartment of a block can display its properties of any kind.

PARTS



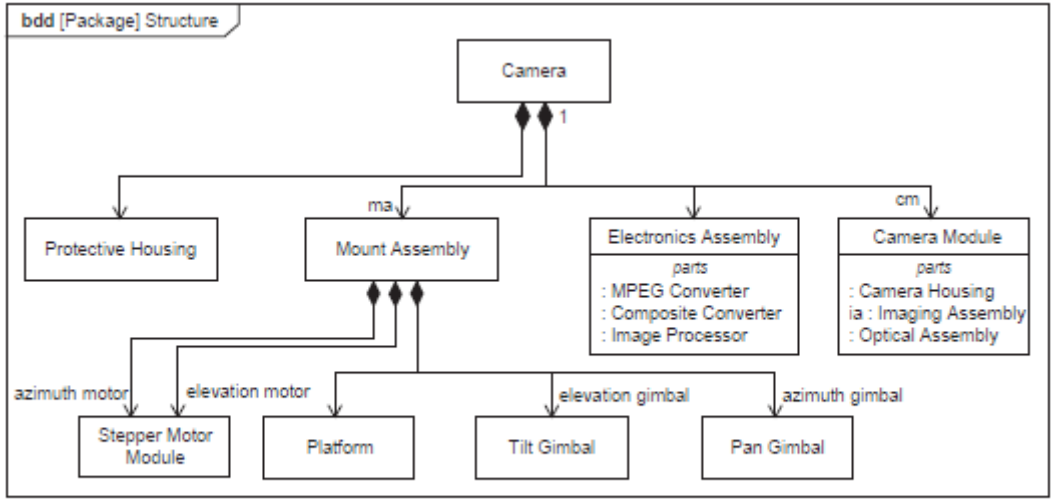


FIGURE 7.5

Showing a block composition hierarchy on a block definition diagram.

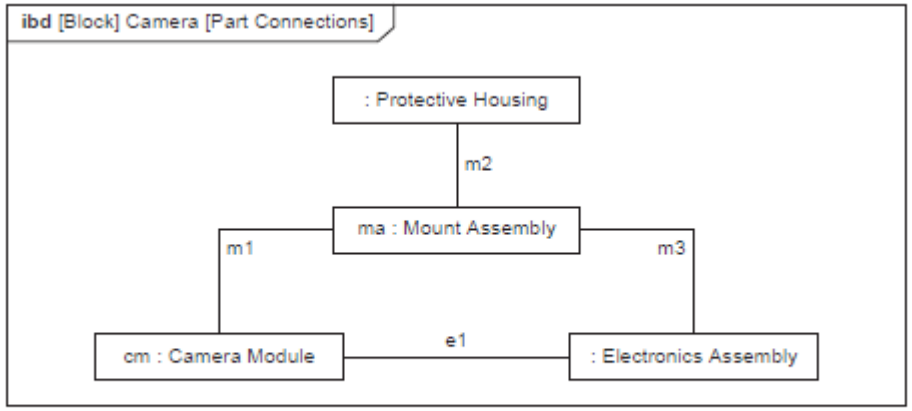


FIGURE 7.7

Connecting parts on an internal block diagram.

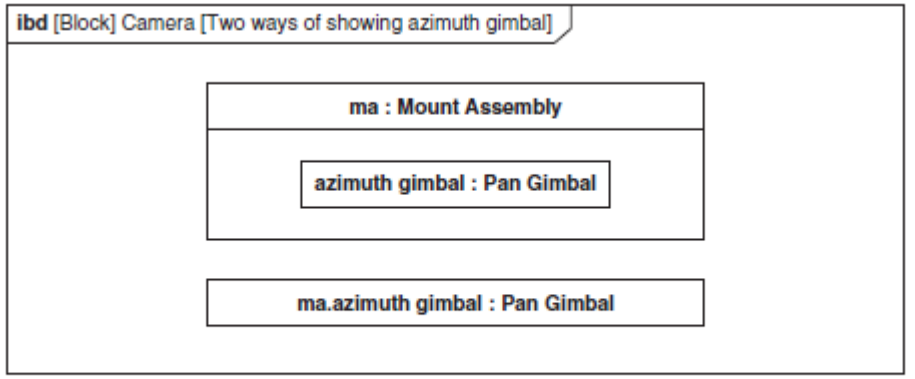


FIGURE 7.8

Showing deep-nested parts on an internal block diagram.

REFERENCES

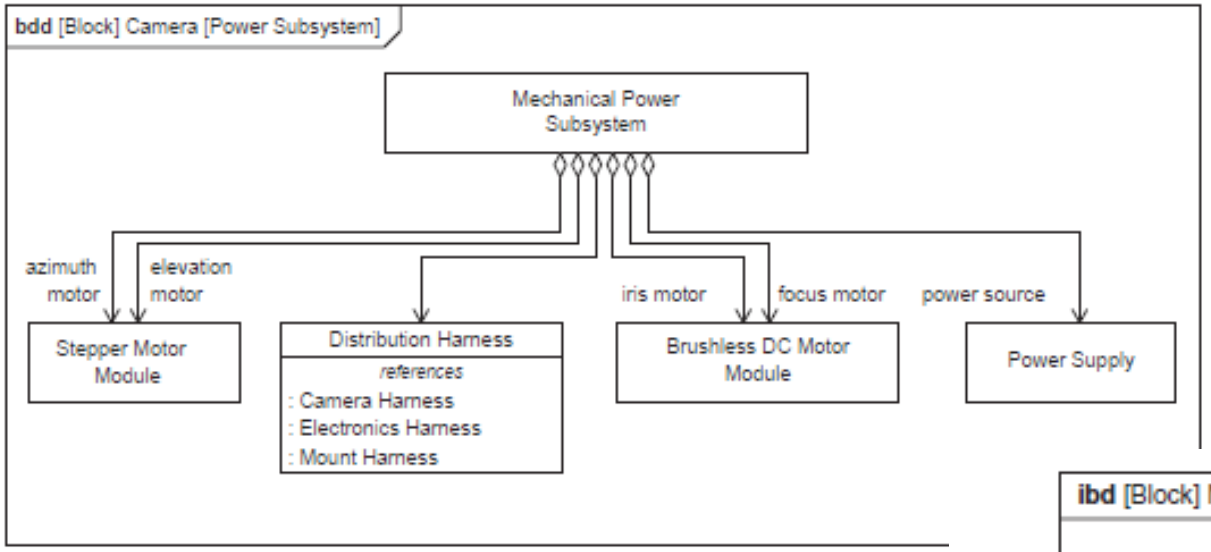


FIGURE 7.10

Reference associations on a block definition diagram.

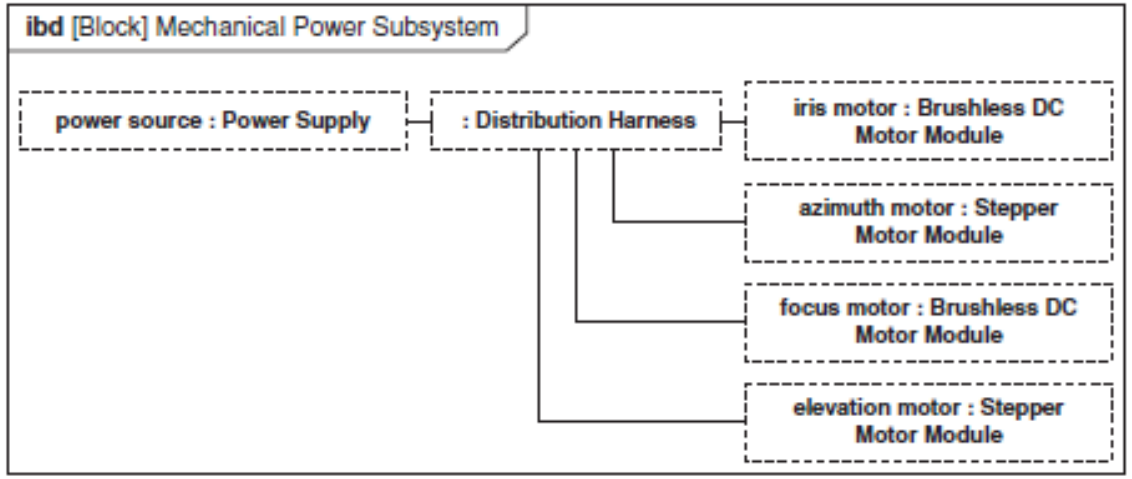


FIGURE 7.11

Reference properties and their interconnections on an internal block diagram.

VALUES

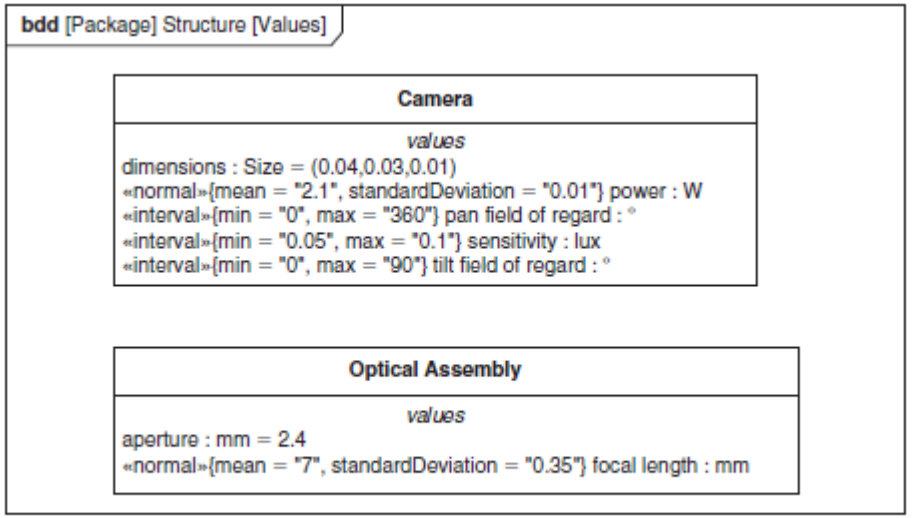


FIGURE 7.22

Examples of property values and distributions.

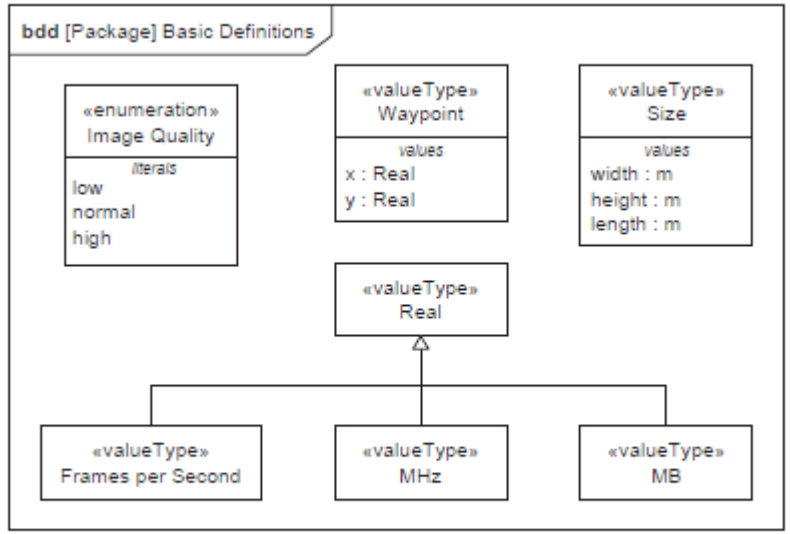


FIGURE 7.17

Definition of basic value types in a block definition diagram.

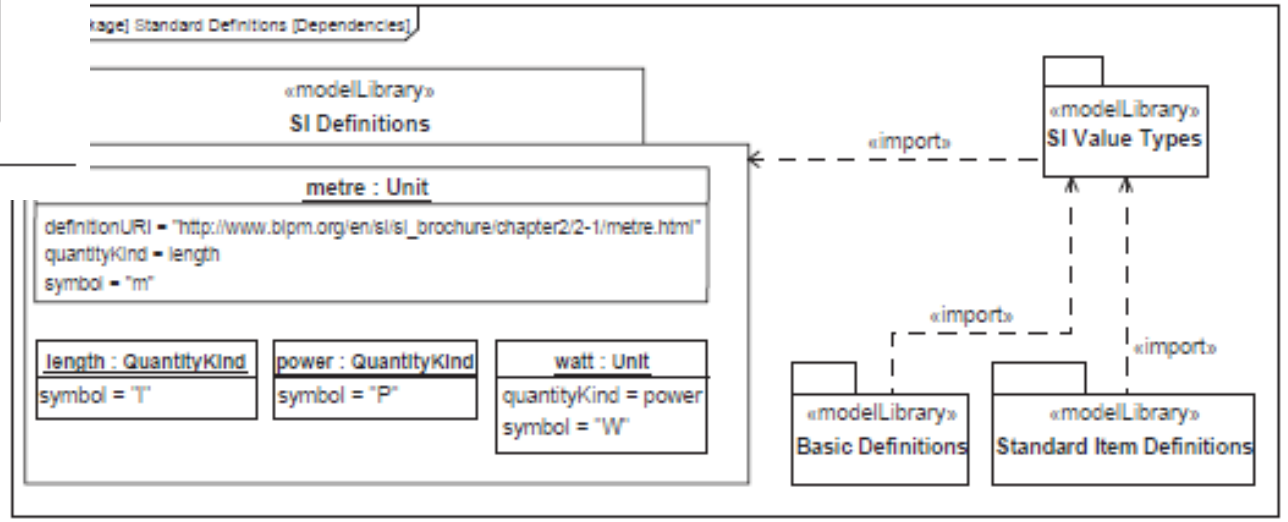


FIGURE 7.18

Importing the SI definitions defined by SysML.

RELATIONSHIPS BETWEEN BLOCKS

- COMPOSITE ASSOCIATIONS - describe whole–part relationships.
- REFERENCE ASSOCIATIONS - describe a logical hierarchy that references blocks that are part of other composition hierarchies.
- HERITAGE - describe the hierarchical specialization of elements
  - Polymorfism – different objects, with the same root, that can response to the same request

COMPOSITION EXAMPLE

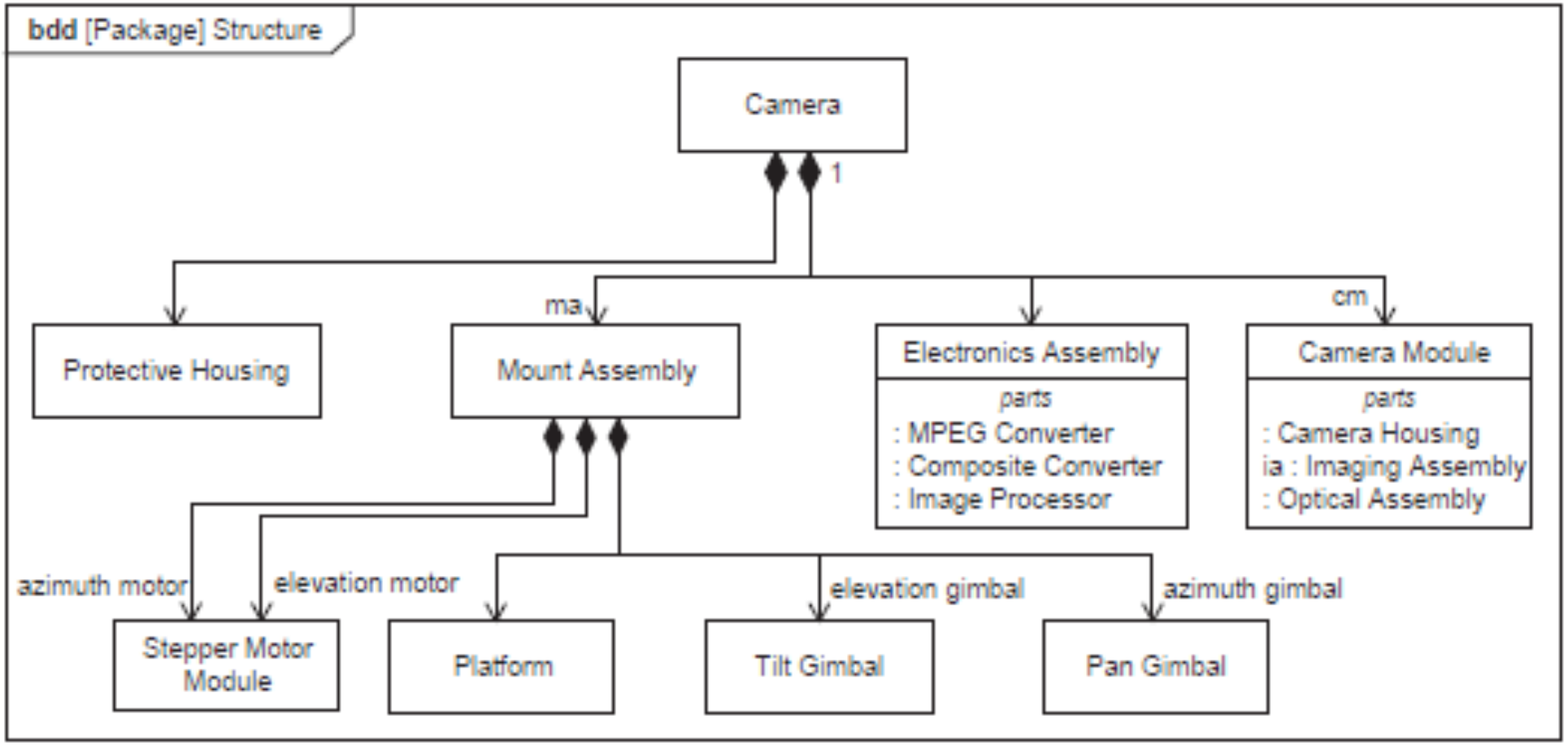


FIGURE 7.5

Showing a block composition hierarchy on a block definition diagram.

REFERENCE EXAMPLE

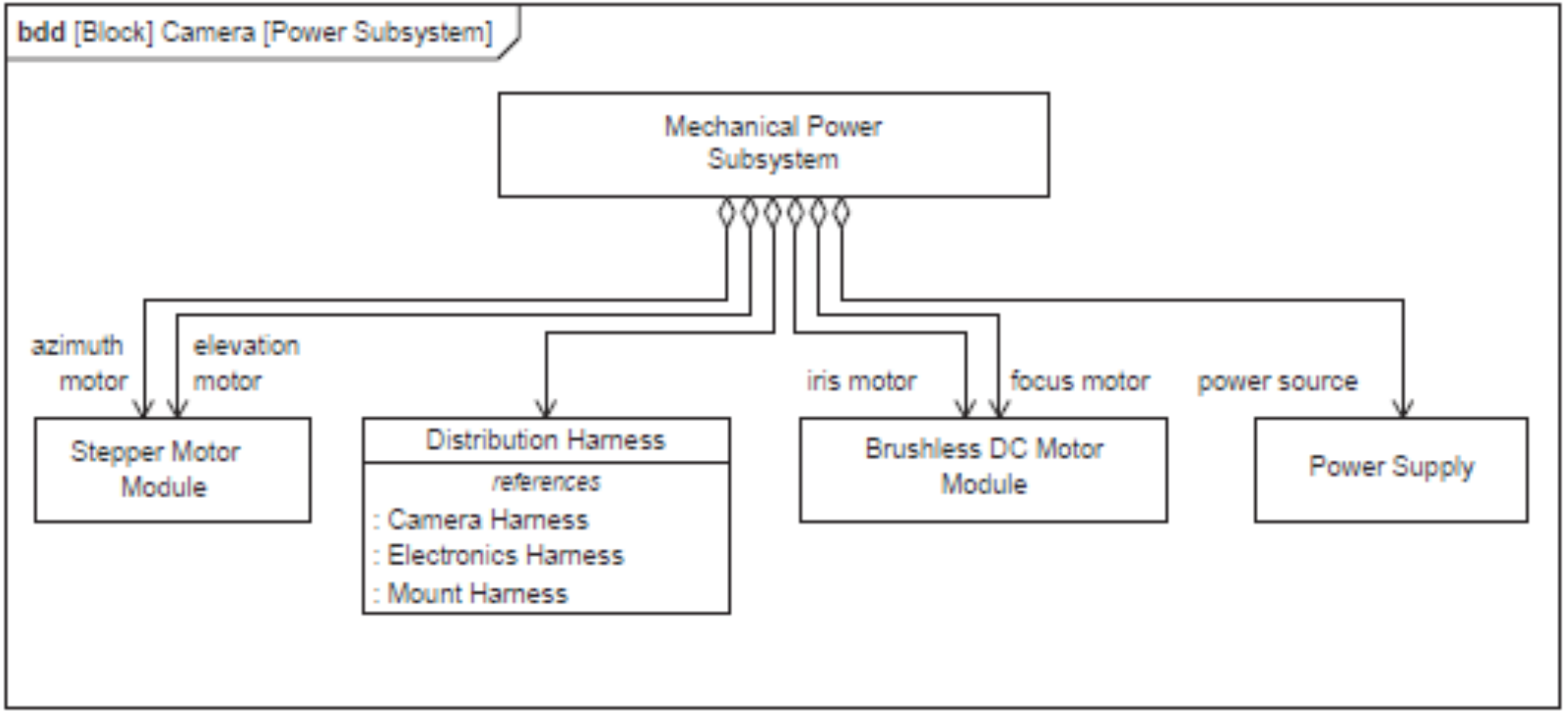


FIGURE 7.10

Reference associations on a block definition diagram.

HERITAGE EXAMPLE

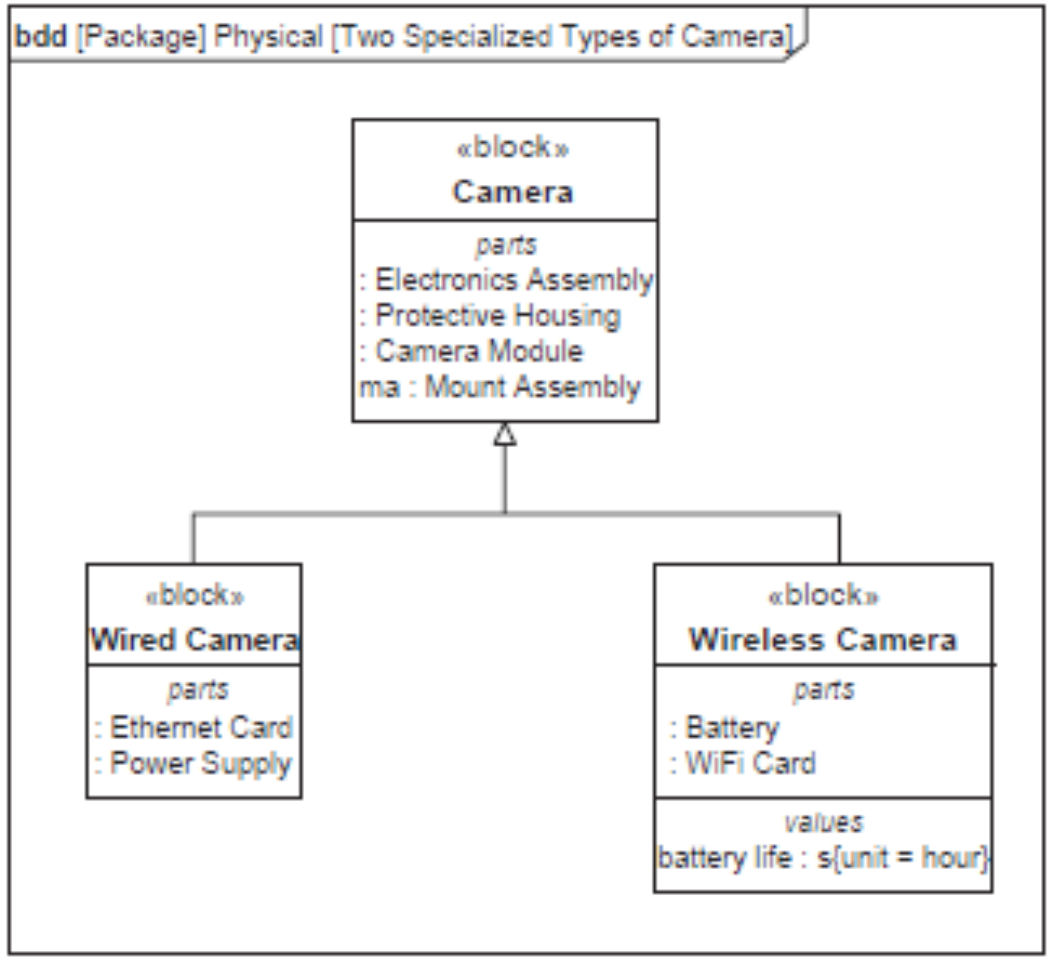


FIGURE 7.48

Example of block specialization.

POLYMORPHISM EXAMPLE

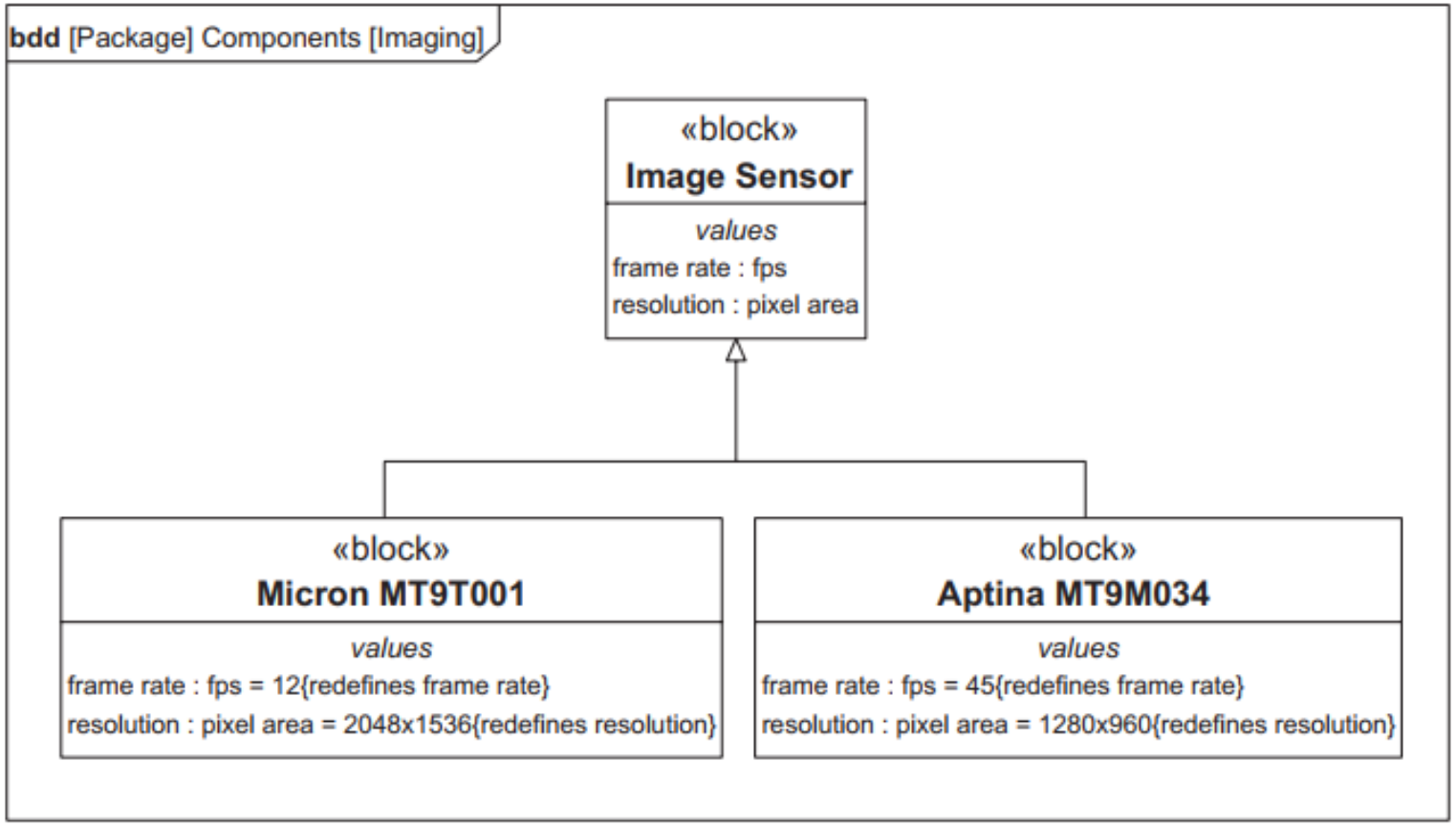


FIGURE 7.52

Two kinds of Imaging Assembly.

FLOWS

- Defining the flows between different parts of a system can provide an abstract view of their interactions.
- Each flow property has a name, type, multiplicity, and direction.
- An item flow specifies the type of item flowing and the direction of the flow.

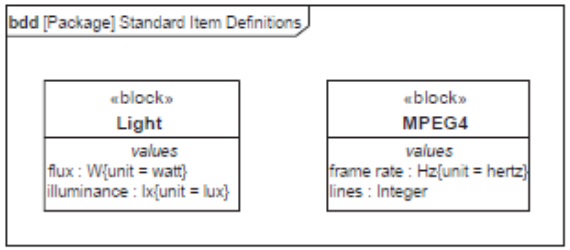


FIGURE 7.23

Items that flow in the *Camera* system.

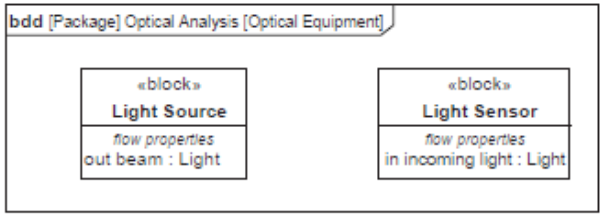


FIGURE 7.24

Flow properties on blocks.

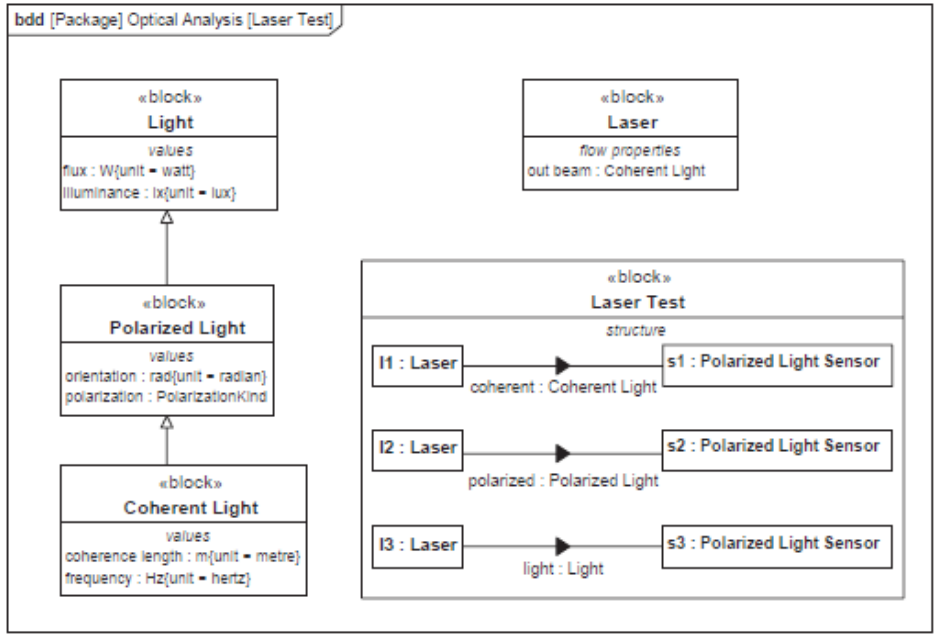


FIGURE 7.27

Item flows between parts.

BLOCKS CAN MODEL BEHAVIOR (FUNCTIONS)

- A block **may designate one behavior as its main or classifier behavior**, which starts executing when the block is instantiated. Other behaviors may be designated as methods, which provide the detail of how service requests are handled.
- **Behaviors have parameters that are used to pass items into or out of the behavior** before, after, and sometimes during execution.
- An **operation** is a behavioral feature that is typically triggered by a synchronous request (i.e., when the requester waits for a response). Each operation defines a set of parameters that describes the arguments passed in with the request, or passed back out once a request has been handled, or both. A **reception** is associated with a signal that defines a message with a set of attributes that represent the content of the message; the parameters of the reception must be the same as the attributes of the associated signal.
- **The main behavior (also called classifier behavior)** of a block starts executing at the beginning of the block's lifetime and generally terminates at the end of its lifetime, although it may terminate before then.
- SysML supports the notion of **polymorphism**, which means that many different blocks may respond to the same stimulus, but each may do so in a specific way, by invoking a specific method.

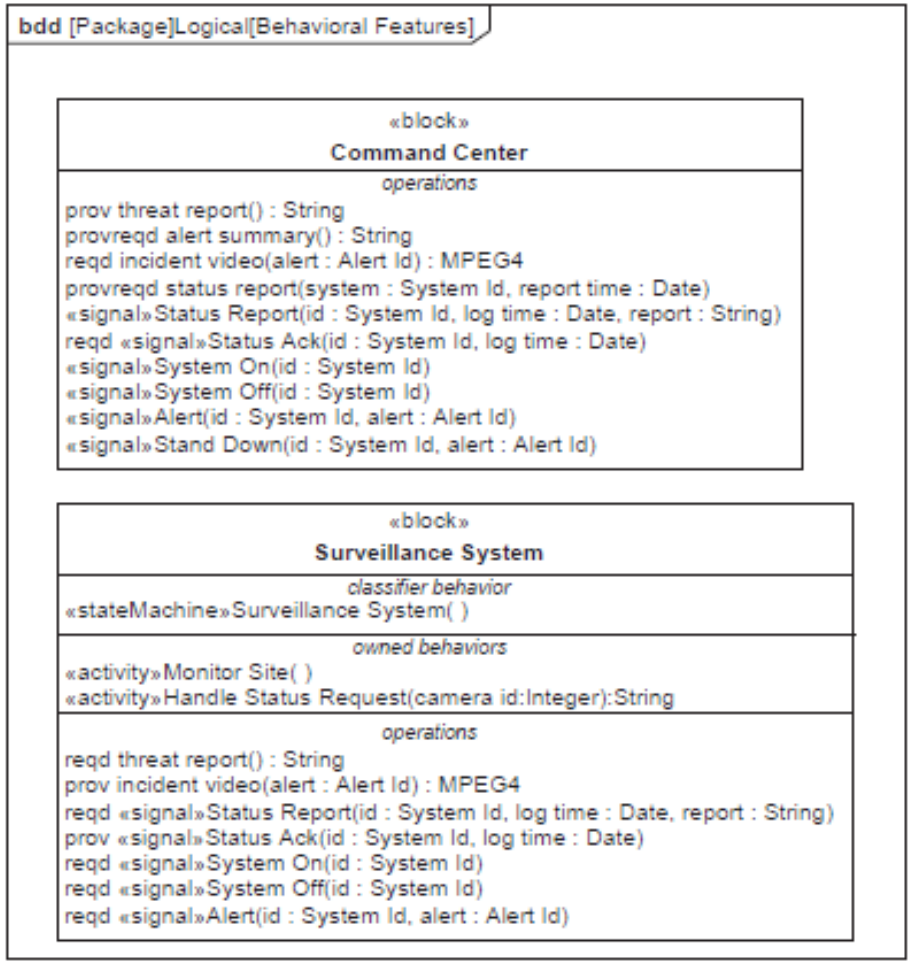


FIGURE 7.30

Blocks with behavioral features.

PORTS

- A port represents an access point on the boundary of a block and on the boundary of any part or reference typed by that block. A block may have many ports that specify different access points. Ports can be connected to one another by connectors on an internal block diagram to support the interaction between parts.
  - A full port is equivalent to a part on the boundary of the parent block that is made available as an access point to and from the block.
  - A proxy port does not constitute a part of its parent block, but instead provides external access to and from the features of its parent block or the block's parts without modifying its inputs or outputs.

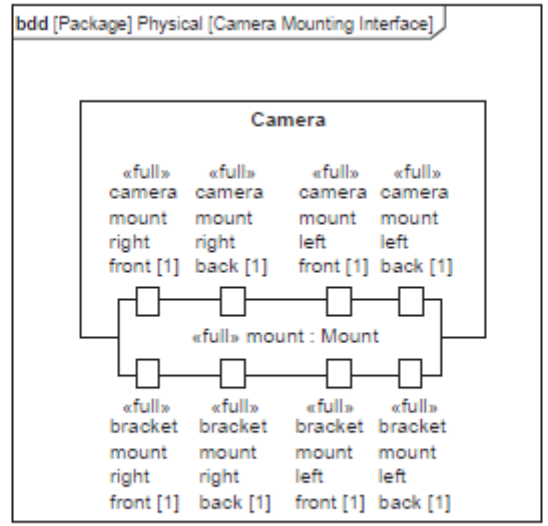


FIGURE 7.32

A full port with nested ports.

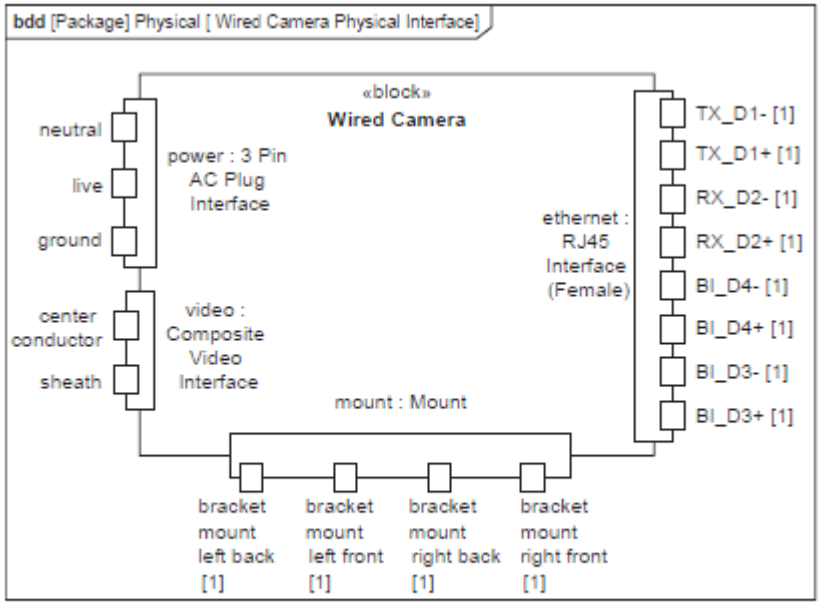


FIGURE 7.34

A block with nested ports.

CONNECTING PORTS

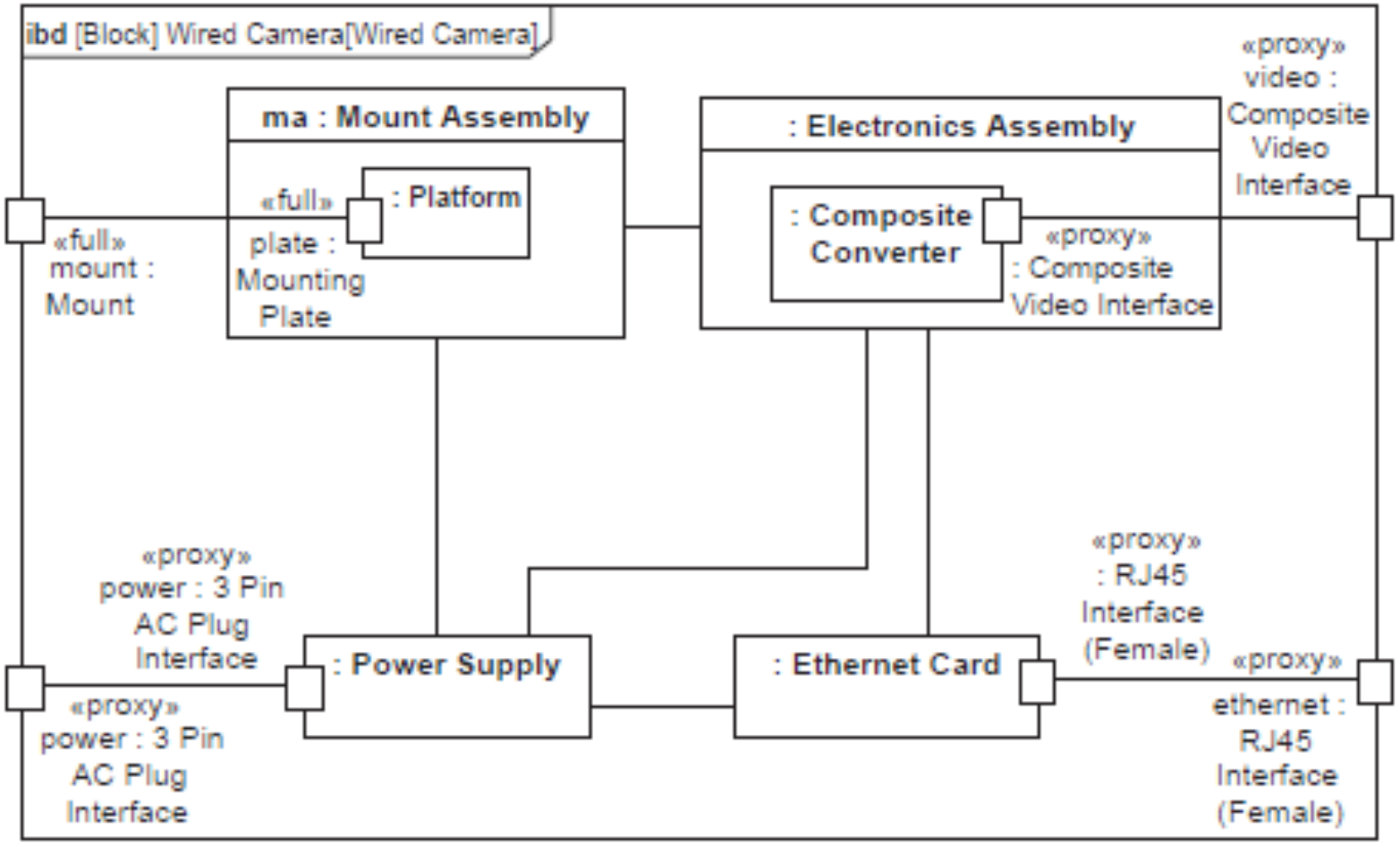


FIGURE 7.35

Connecting ports internally to a block.

ITEM FLOWS BETWEEN PORTS.

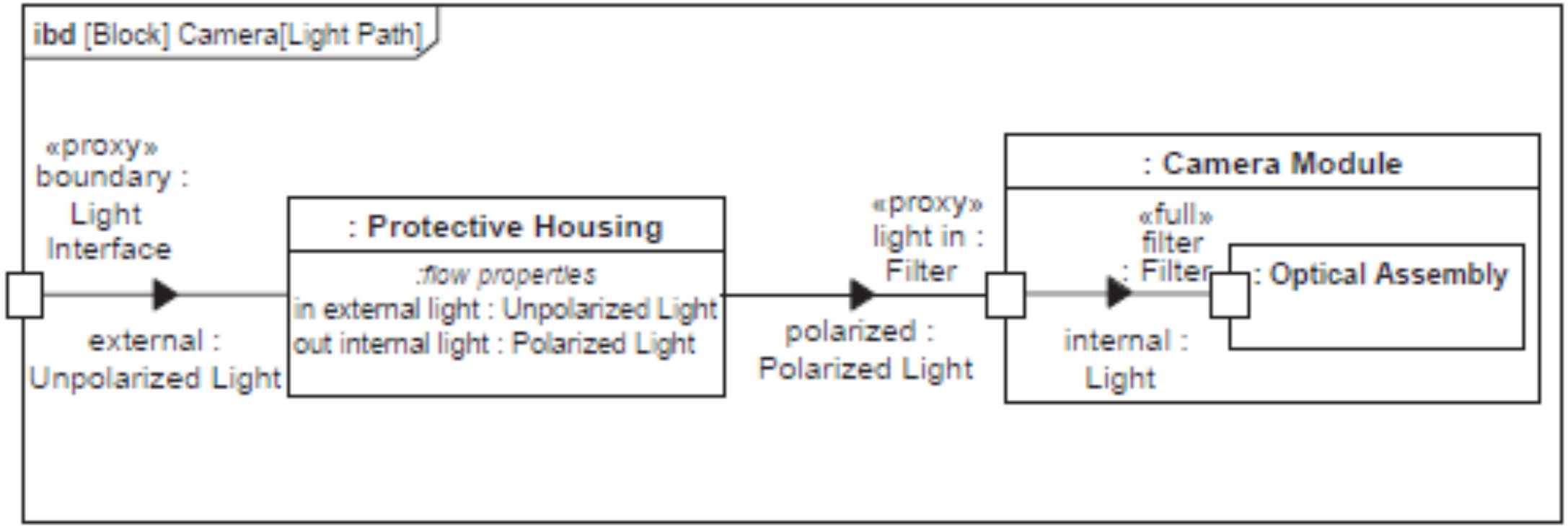


FIGURE 7.44

Item flows between ports.

INTERFACES

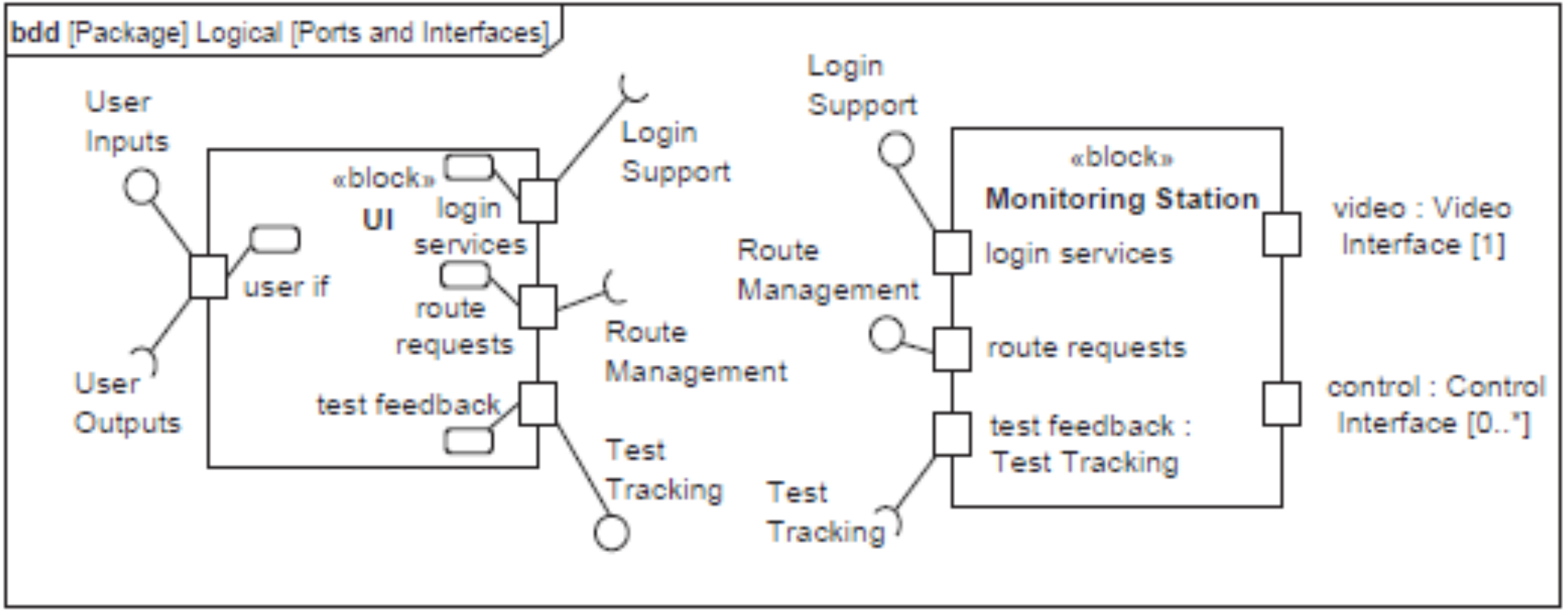


FIGURE 7.46

Defining a service-based interface using proxy ports.

CONSIDERAÇÕES FINAIS

SO...

- Temos que repor a aula de semana que vem, como vamos fazer?
  - Sugestão: quarta a noite remoto sincrono.
- AI-03 - Exercícios sobre arquitetura funcional e diagrama de blocos.
  - Escolha 15 exercício do capitulo do livro.
- AG-03 - Resumo sobre arquitetura funcional.
  - Pesquisem sobre como se faz arquitetura. G1: TOGAF. G2: DoDAF. G3: UAF. G4: NAF
  - Apresentação de 5 min na próxima aula presencial.