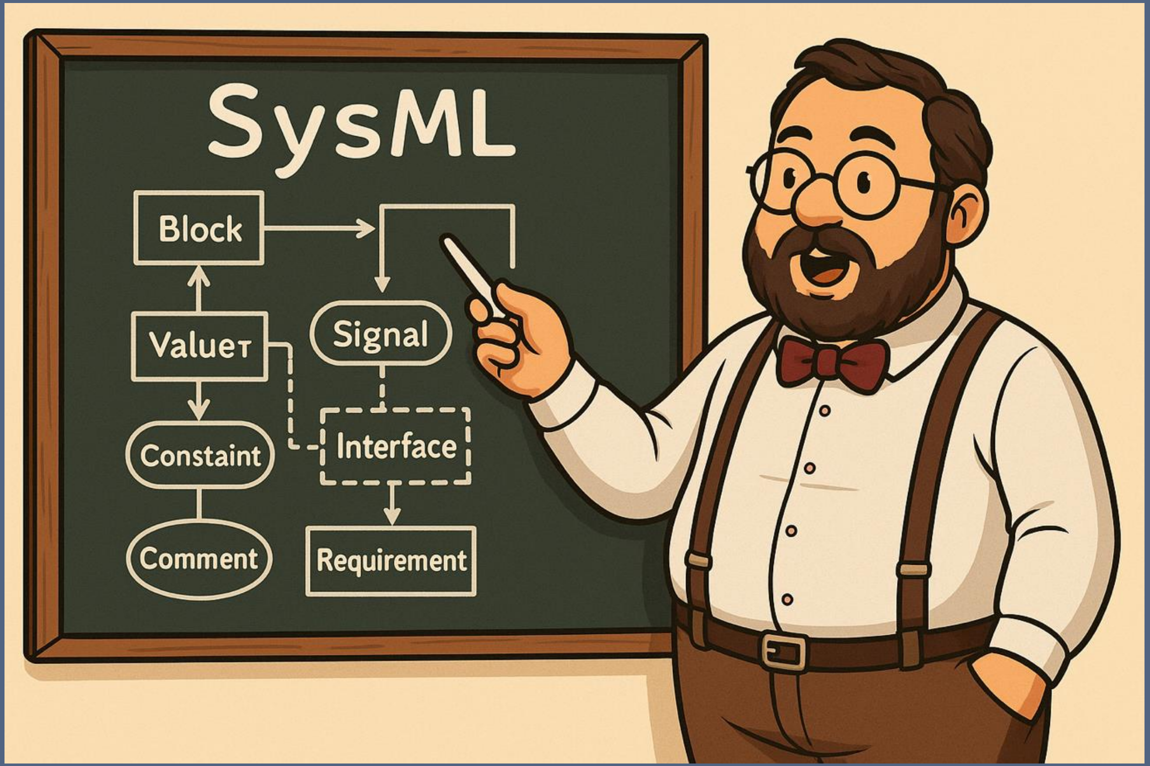
Graduate Program in Science and Space Technologies (PG-CTE)

SPACE SYSTEMS, TESTING AND LAUNCHING (CTE-E)



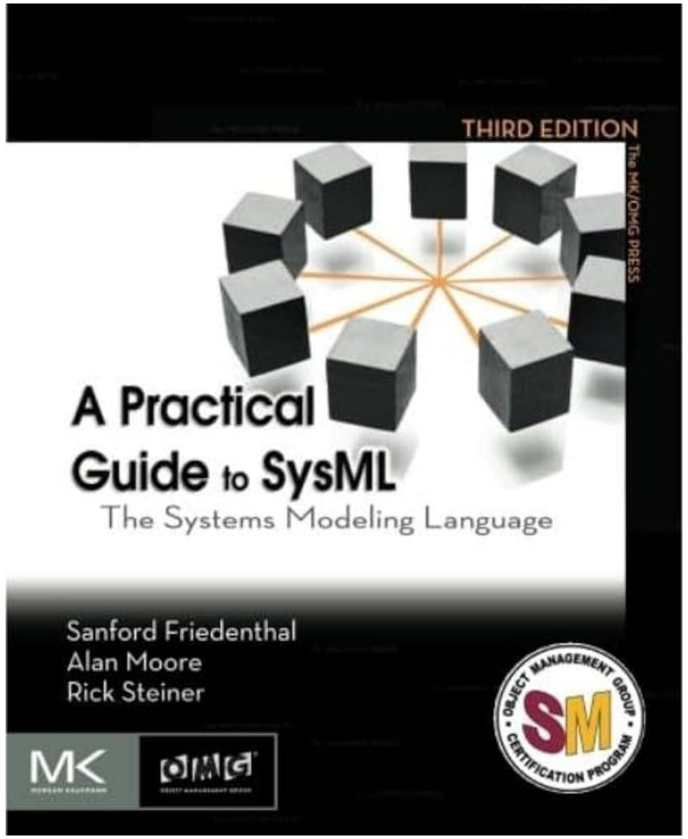
SYSML INTRODUCTION

Prepared by Prof. Dr. Christopher Shneider Cerqueira

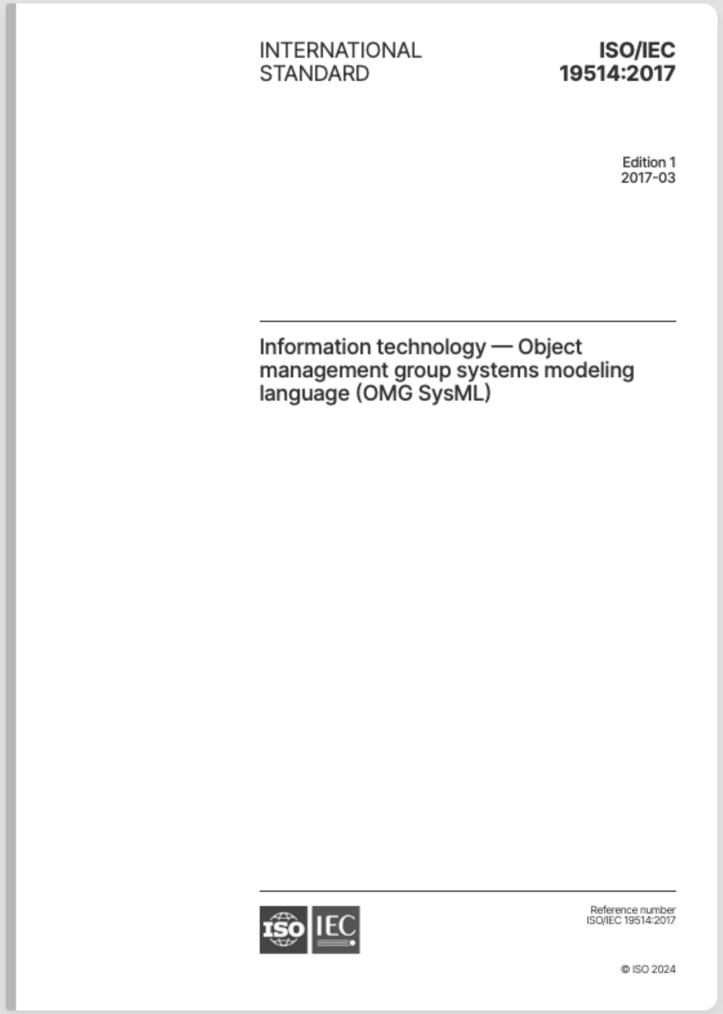
WEEK	CLASS ACTIVITY	REF	INDIVIDUAL	W	GROUP	W
~~**1** 28/Jul~~	~~Course Structure and Initial Definitions Systems Engineering Review~~	~~[1] [2] [3] [4]~~	~~IA-01 - Reading and Conceptual Questions (10)~~	~~10%~~		~~0%~~
~~**2** 04/Aug~~	~~Classical Systems Engineering Diagrams (IDEF-0/N2/eFFBD/DFD)~~	~~[4] *papers~~	~~IA-02~~	~~0%~~	~~GA-02 - Prepare a representation of your system using classical Diagrams~~	~~50%~~
~~**3** 11/Aug~~	~~Transition from Legacy to MBSE MBSE Methodologies MBSE Languages~~	~~[5] [7] *papers~~	~~IA-03 - Reading and Conceptual Questions (10)~~	~~10%~~		~~0%~~
~~**4** 18/Aug~~	~~OPM - Basic~~	~~[6]~~	~~IA-04 - Exercises~~	~~10%~~		~~0%~~
~~**5** 25/Aug~~	~~OPM - Extended~~	~~[6]~~	~~IA-05 - Exercises~~	~~10%~~		~~0%~~
6 01/Sep	OPM - Group Presentation		IA-06	0%	G6 - Prepare a presentation of your system using OPM	50%
7 08/Sep	SysML Introduction (bdd/ibd)	[7]	IA-07 - Exercises	10%		0%
8 15/Sep	P1 - Conceptual Questions and Case	[1] [2] [3] [4] [6]	IA-08 - Questions and a mini-case	50%	GA-08 -	
				100%		**100%**

Resources for this lecture

- Para baixar o demo enquanto não temos o "oficial"
  - ✓ 1- UML 2 diagramming, OO software modeling, Source code engineering Tool MagicDraw UML from No Magic
  - ✓ https://www.magicdraw.com/main.php?ts=download&cmd_show_mirrors=8731&menu=download_cameo_systems_modeler&c=a62cfa6271bd46bf5452e1480138f711&pr=8013&NMSESSID=9f3d0fe455b0b1dd17b3d81f94244dcc&product_version=2021x&product_edition=Cameo+Systems+Modeler%7CEnterprise&group=39
  - ✓ 2- Fazer cadastro
  - ✓ 3 - Selecionar Cameo Systems Modeler no menu à esquerda
  - ✓ 4 - versão 2021x LTR / Edição Enterprise
  - ✓ 5 - Baixar a versão (No Install)



Essa é uma versão trial para ser usada somente nas nossas aulas!!!


INTERNATIONAL STANDARD

ISO/IEC 19514:2017

Edition 1
2017-03

Information technology — Object management group systems modeling language (OMG SysML)

Reference number
ISO/IEC 19514:2017

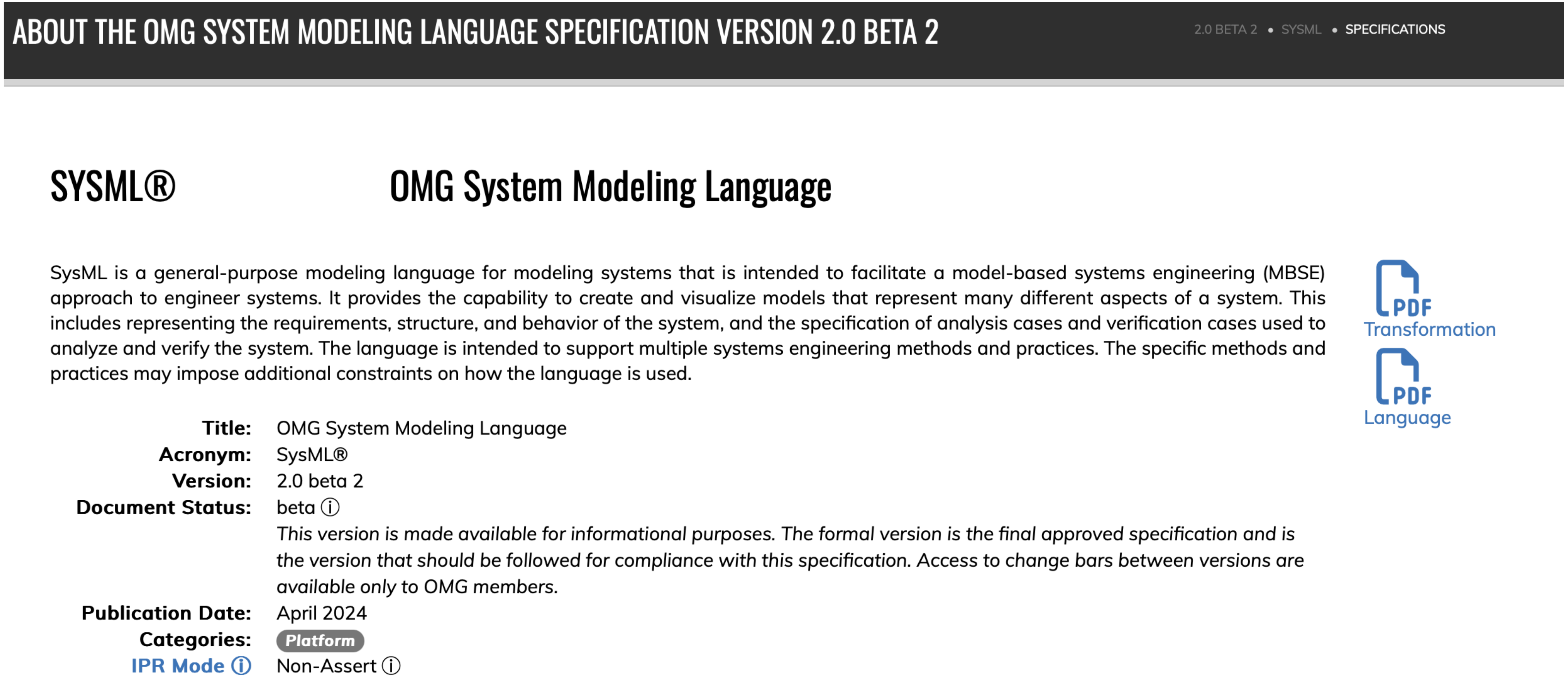
© ISO 2024

Read sample

ISO/IEC 19514:2017

Information technology — Object management group systems modeling language (OMG SysML)

Published (Edition 1, 2017)



ABOUT THE OMG SYSTEM MODELING LANGUAGE SPECIFICATION VERSION 2.0 BETA 2

2.0 BETA 2 • SYSML • SPECIFICATIONS

SYSML® OMG System Modeling Language

SysML is a general-purpose modeling language for modeling systems that is intended to facilitate a model-based systems engineering (MBSE) approach to engineer systems. It provides the capability to create and visualize models that represent many different aspects of a system. This includes representing the requirements, structure, and behavior of the system, and the specification of analysis cases and verification cases used to analyze and verify the system. The language is intended to support multiple systems engineering methods and practices. The specific methods and practices may impose additional constraints on how the language is used.

PDF Transformation

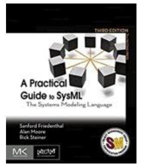
PDF Language

Title:	OMG System Modeling Language
Acronym:	SysML®
Version:	2.0 beta 2
Document Status:	beta ⓘ *This version is made available for informational purposes. The formal version is the final approved specification and is the version that should be followed for compliance with this specification. Access to change bars between versions are available only to OMG members.*
Publication Date:	April 2024
Categories:	**Platform**
IPR Mode ⓘ	Non-Assert ⓘ

https://www.omg.org/spec/SysML

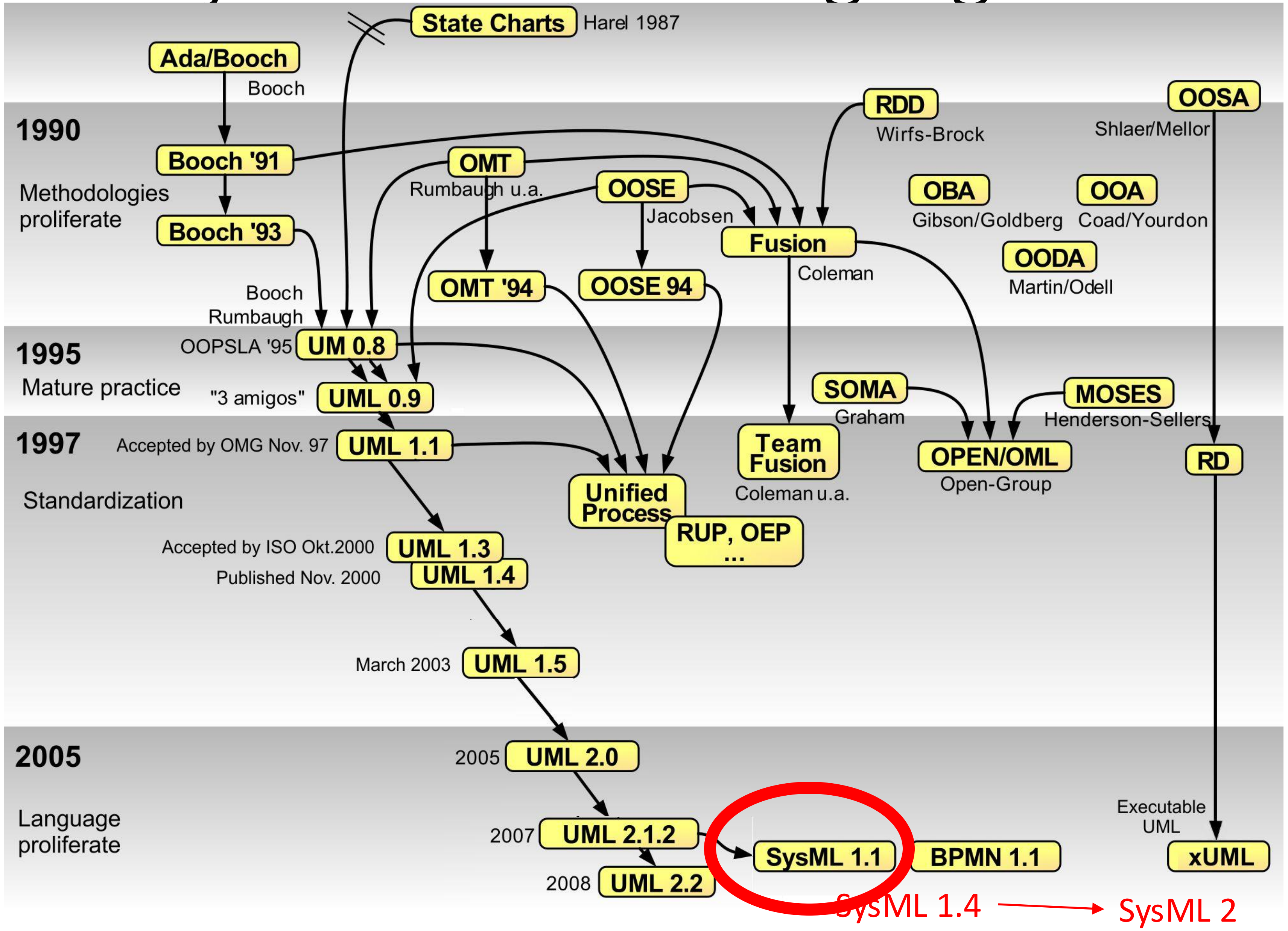
SysML History / Timeline

SYSML

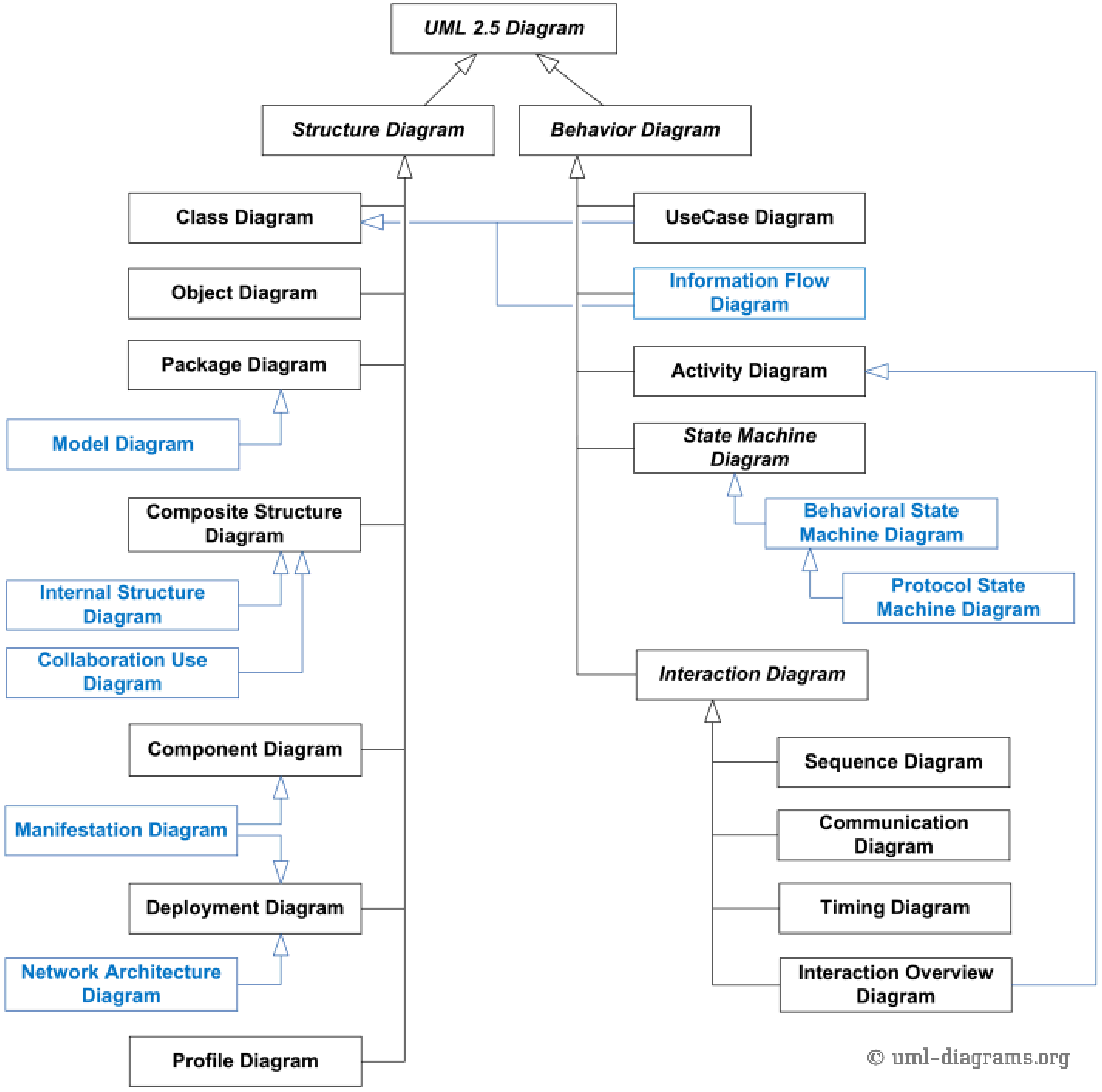


- SysML is a **general-purpose graphical modeling language** that supports the **analysis, specification, design, verification, and validation** of complex systems. These systems can include **hardware and equipment, software, data, personnel, procedures, facilities, and other elements of human** and natural systems.
- SysML can represent the following aspects of systems, components, and other entities:
  - *Structural composition, interconnection and classification;*
  - *Flow-based, message-based, and stateful behavior;*
  - *Constraints on physical and performance properties;*
  - *Allocations between behavior, structure, and constraints; and*
  - *Requirements and their relationship to other requirements, design elements, and test cases.*

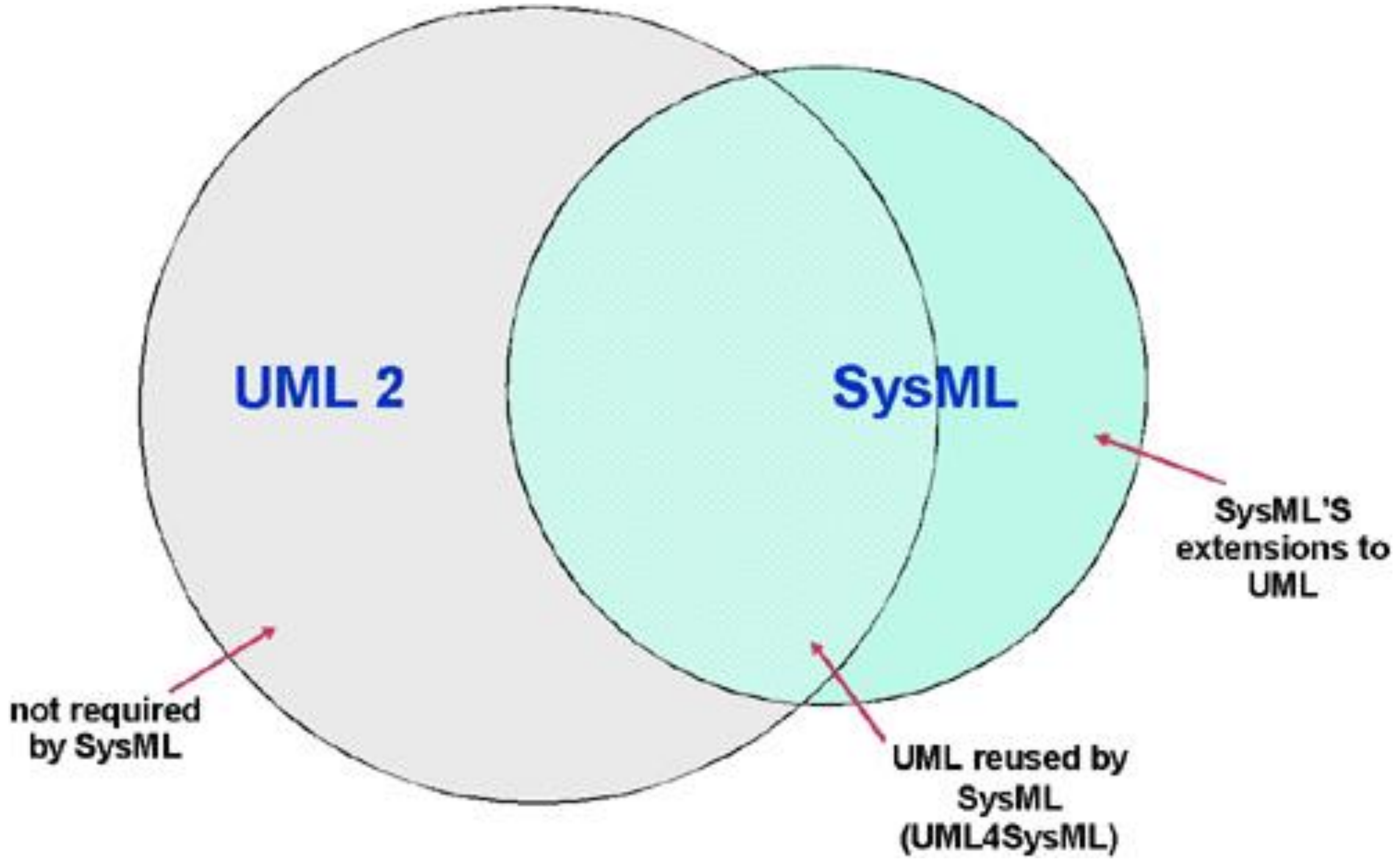
History of Object-Oriented Languages



UML – Unified Modelling Language



Adaptation of the UML to the systemic domain



SysML taxonomy.

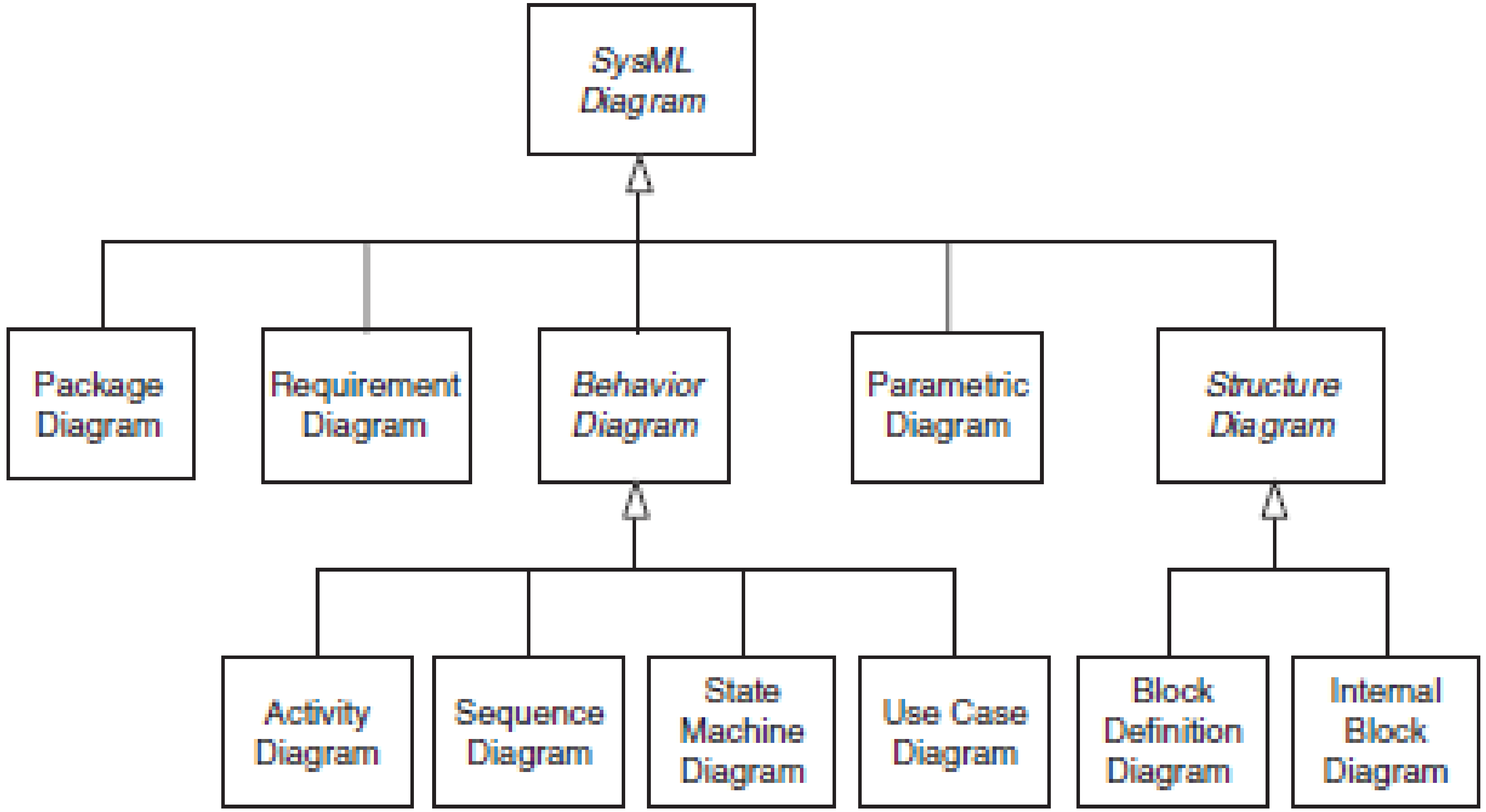
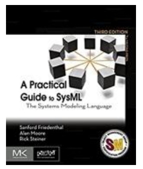
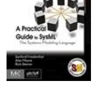


FIGURE 3.1

SysML diagram taxonomy.



- Each type of diagram is summarized here, along with its relationship to UML diagrams:
  - **Package diagram** – presents the organization of a model in terms of packages that contain model elements (same as UML package diagram).
  - **Requirement diagram** – presents text-based requirements and their relationships to other requirements, design elements, and test cases to support requirements traceability (not in UML).
  - **Activity diagram** – presents a flow-based behavior indicating the order in which actions are performed based on the availability of their inputs, outputs, and control, and how actions transform inputs into outputs (UML activity diagram modification).
  - **Sequence diagram** – presents the behavior in terms of a sequence of messages exchanged between systems or parts of systems (same as UML sequence diagram).
  - **State machine diagram** – presents the behavior of an entity in terms of its transitions between event-triggered states (same as the UML state machine diagram).
  - **Use case diagram** – presents functionality in terms of how a system is used by external entities (i.e., actors) to achieve a set of goals (same as the UML use case diagram).
  - **Block definition diagram** – presents structural elements, called blocks, and their composition and classification (modification of the UML class diagram).
  - **Internal block diagram** – presents interconnection and interfaces between the parts of a block (modification of the UML composite structure diagram).
  - **Parametric diagram** – presents constraints on property values, such as F = m * a, used to support engineering analysis (not in UML).



SysML structural diagrams

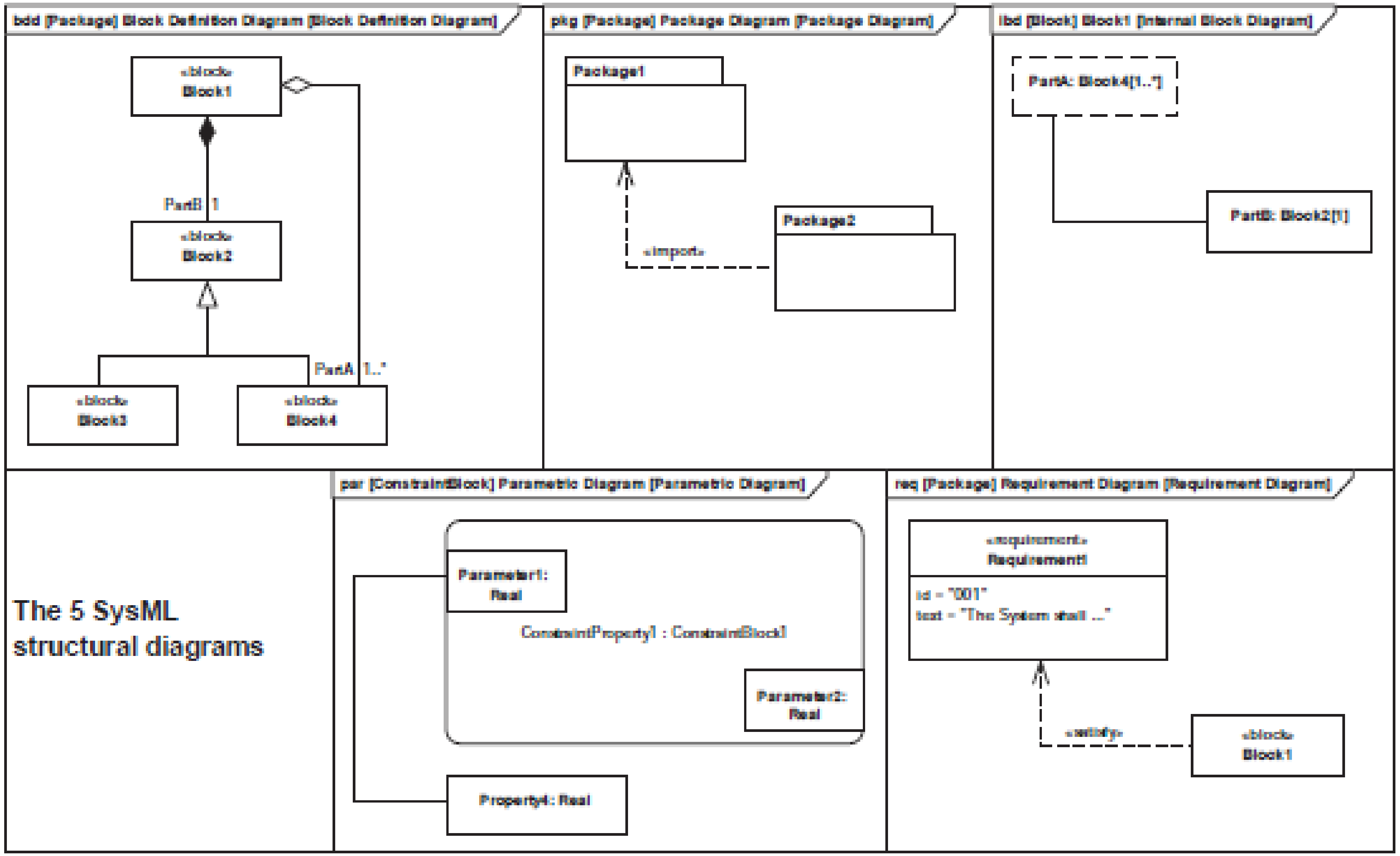

bdd [Package] Block Definition Diagram [Block Definition Diagram]
«block» Block1
PartB 1
«block» Block2
PartA 1..*
«block» Block3
«block» Block4
pkg [Package] Package Diagram [Package Diagram]
Package1
Package2
«import»
ibd [Block] Block1 [Internal Block Diagram]
PartA: Block4[1..*]
PartB: Block2[1]
The 5 SysML structural diagrams
par [ConstraintBlock] Parametric Diagram [Parametric Diagram]
Parameter1: Real
ConstraintProperty1 : ConstraintBlock1
Parameter2: Real
Property4: Real
req [Package] Requirement Diagram [Requirement Diagram]
«requirement» Requirement1
id = "001"
text = "The System shall ..."
«satisfy»
«block» Block1

Figure 4.7 SysML structural diagrams

SysML Behavioral Diagrams

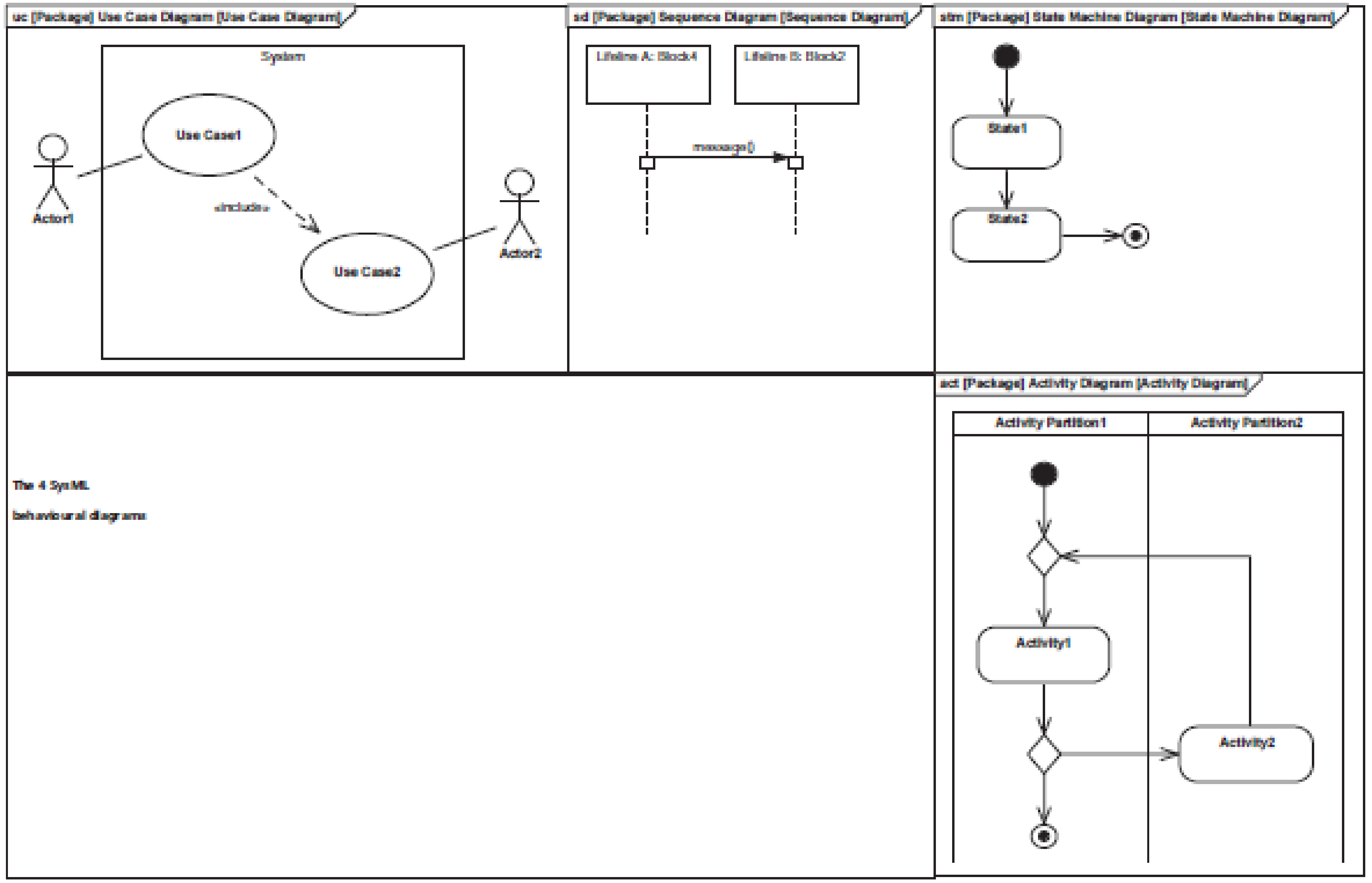

uc [Package] Use Case Diagram [Use Case Diagram]
System
Use Case1
Use Case2
«include»
Actor1
Actor2
sd [Package] Sequence Diagram [Sequence Diagram]
Lifeline A: Block4
Lifeline B: Block2
message()
stm [Package] State Machine Diagram [State Machine Diagram]
State1
State2
act [Package] Activity Diagram [Activity Diagram]
Activity Partition1
Activity Partition2
Activity1
Activity2
The 4 SysML
behavioural diagrams

Figure 4.8 SysML behavioural diagrams

SysML

- It is
  - a **visual modelling language** that provides
    - Semantics (meaning)
    - Notation (Representation of meaning)
- It is not
  - a methodology or a tool

Diagram Basic Parts

Frame

- Each **SysML diagram** should have a **diagram frame** that includes the contents of the diagram.
- The frame corresponds to a model element that provides the context for the content of the diagram.

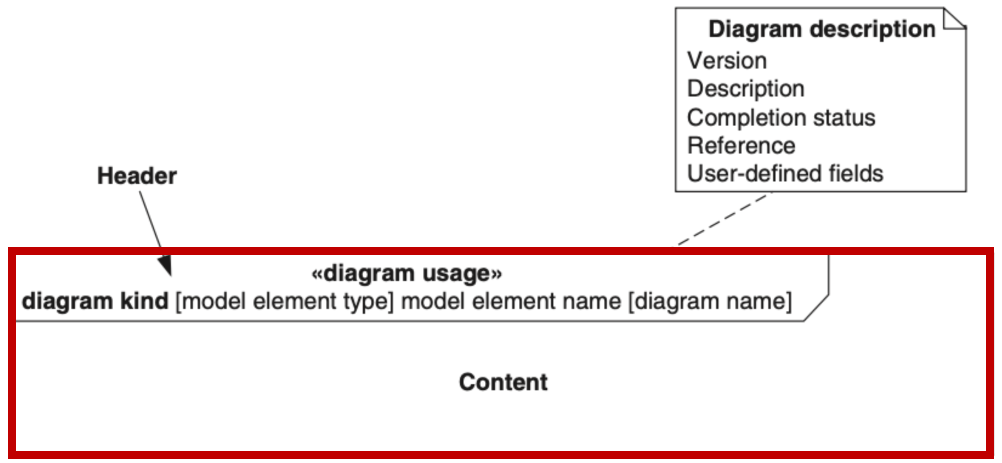


FIGURE 5.5

A diagram frame.

Header

- The **header diagram** is a rectangle with its bottom right corner cut off. Includes the following information:
  - **Diagram kind** - An abbreviation indicating the type of diagram.
  - **Model element kind** - The type of model element that the diagram frame corresponds to.
  - **Model element name** - The name of the model element to which the diagram frame corresponds.
  - **Diagram name** - the name of the diagram, which is usually used to indicate the purpose of the diagram.
  - **Diagram usage** - a keyword that indicates a specialized use of a diagram.

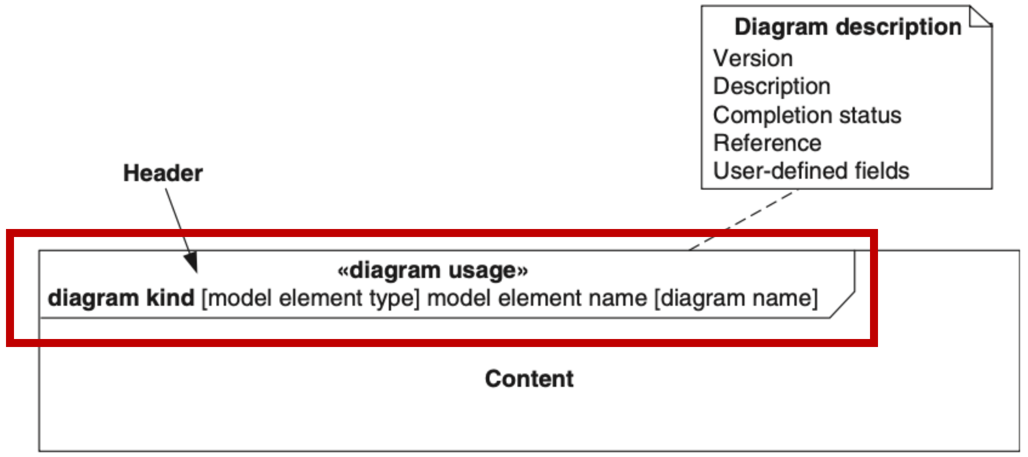


FIGURE 5.5

A diagram frame.

Header – Diagram Kind

- *Activity diagram - act*
- *Block definition diagram - bdd*
- *Internal block diagram - ibd*
- *Package diagram - pkg*
- *Parametric diagram - par*
- *Requirement diagram - req*
- *Sequence diagram - sd*
- *State machine diagram - stm*
- *Use case diagram - uc*

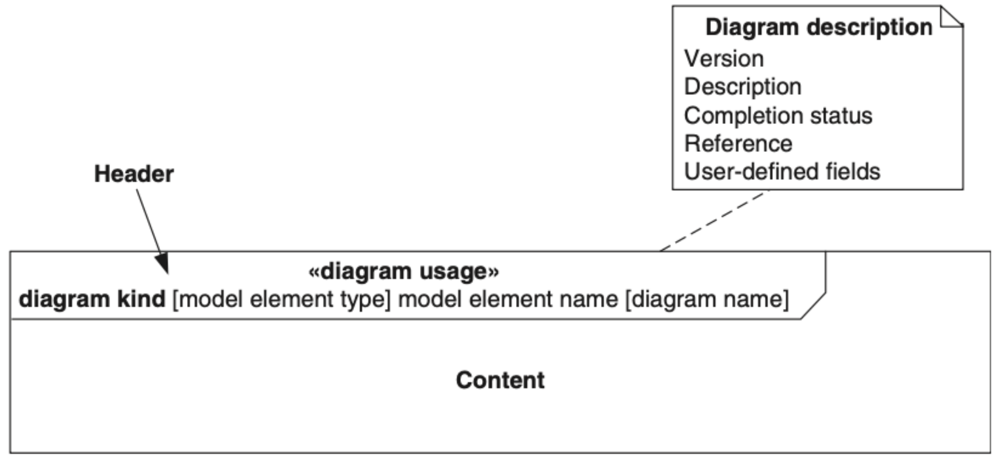


FIGURE 5.5

A diagram frame.

Header – Element Kind

- Different diagram types have diagram frames that correspond to different types of model elements.
  - *Activity diagram—activity*
  - *Block definition diagram—block, constraint block, package, model, model library*
  - *Internal block diagram—block*
  - *Package diagram—package, model, model library, profile, view*
  - *Parametric diagram—activity, block, constraint block*
  - *Requirement diagram—package, model, model library, requirement*
  - *Sequence diagram—interaction*
  - *State machine diagram—state machine*
  - *Use case diagram—package, model, model library*

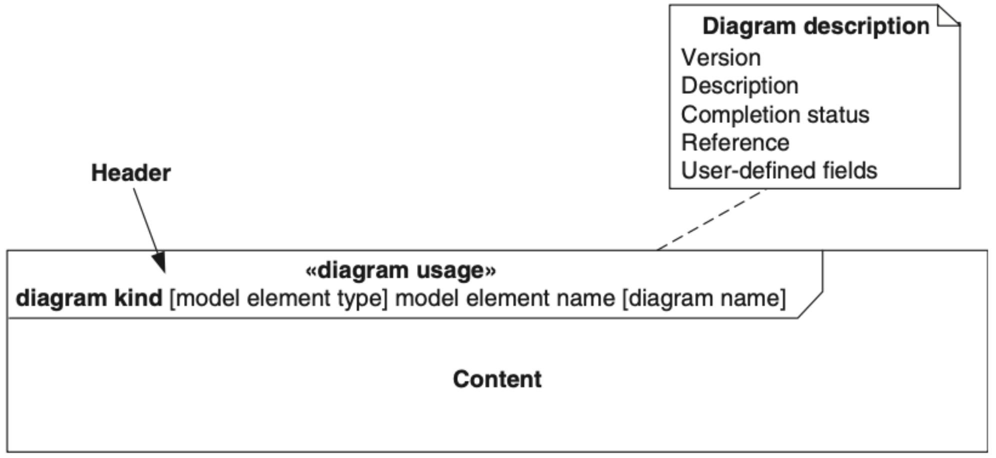


FIGURE 5.5

A diagram frame.

The template element kind should be shown in the header to avoid ambiguity if the diagram can represent more than one type of template element allowed.

Header – Diagram Name

- Because a model can contain considerable amounts of information, the modeler can choose to **include only selected model elements in a specific diagram for a given purpose**, while hiding other model elements that might detract from that purpose.
- The **diagram name** is user-defined and is intended to provide a concise description of the purpose of the diagram.

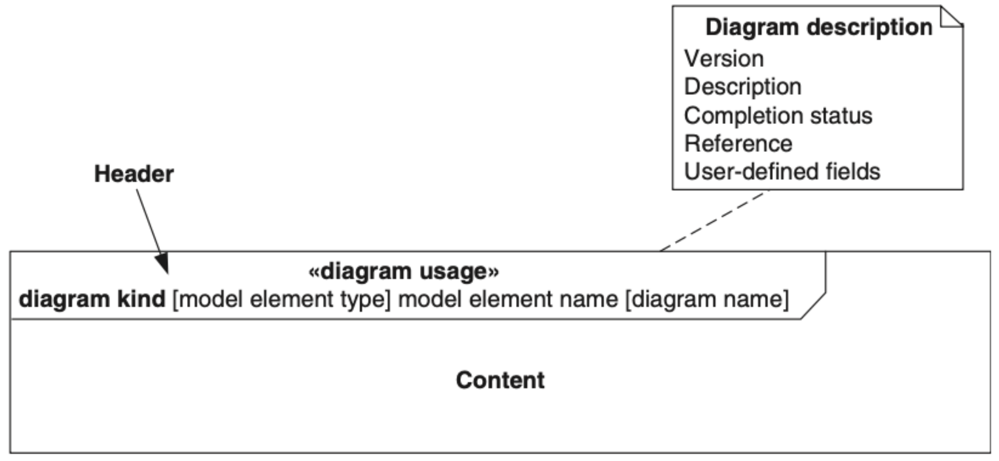


FIGURE 5.5
A diagram frame.

Header – Diagram usage

- The **diagram usage** indicates that a diagram is intended to support a specific usage.
- The usage name of the diagram is included in the header in angle brackets called **guillemets**.

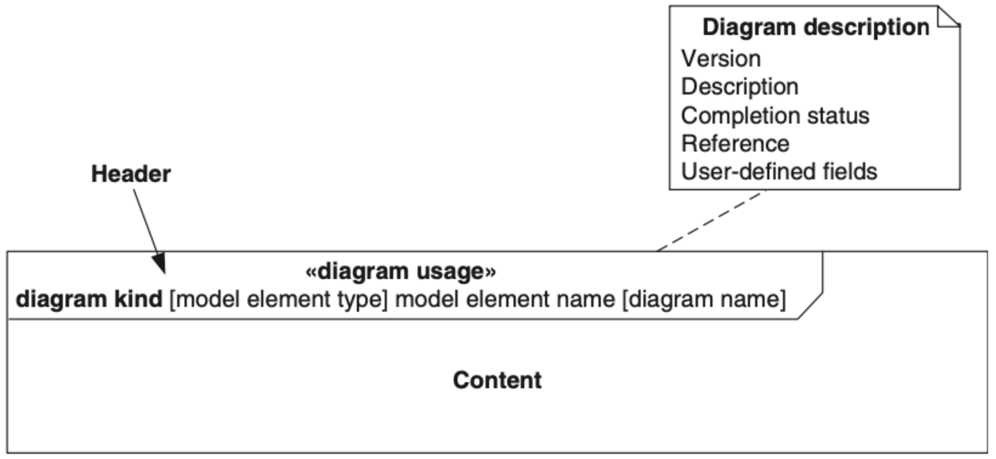


FIGURE 5.5

A diagram frame.

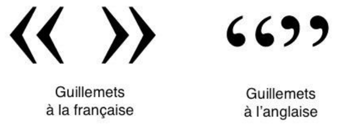


Diagram Description

- The **diagram description** is an optional note attached inside or outside the frame of the diagram.
- It is intended to allow the modeler to capture additional information about the diagram.

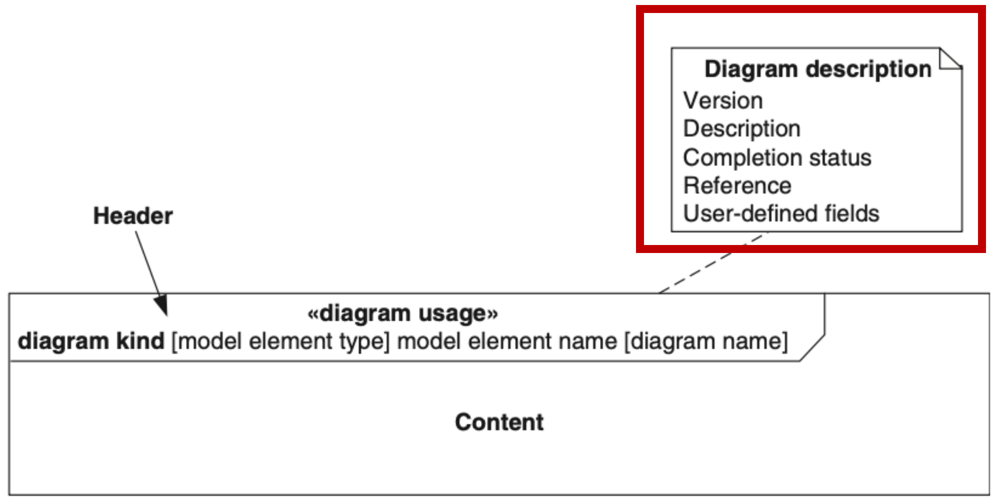


FIGURE 5.5

A diagram frame.

Diagram Content

- The **diagram content area**, sometimes called a canvas, contains elements that graphically represent elements of the model.

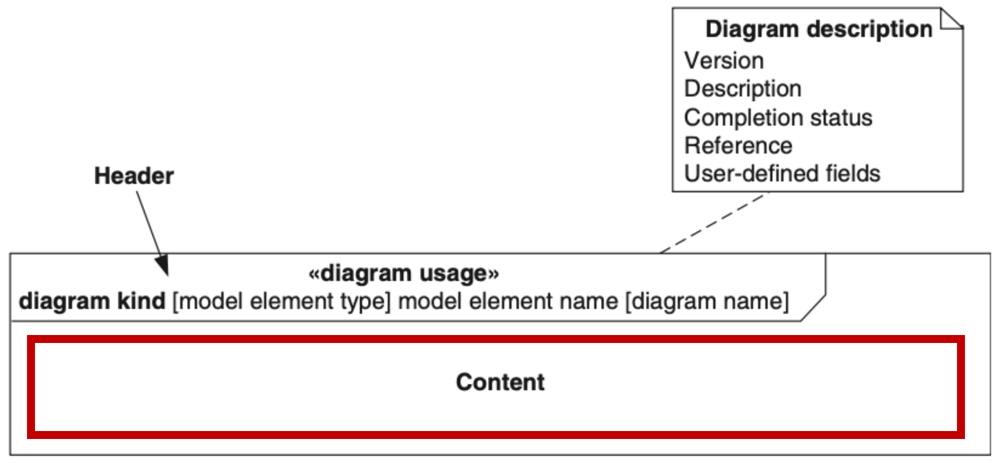


FIGURE 5.5

A diagram frame.

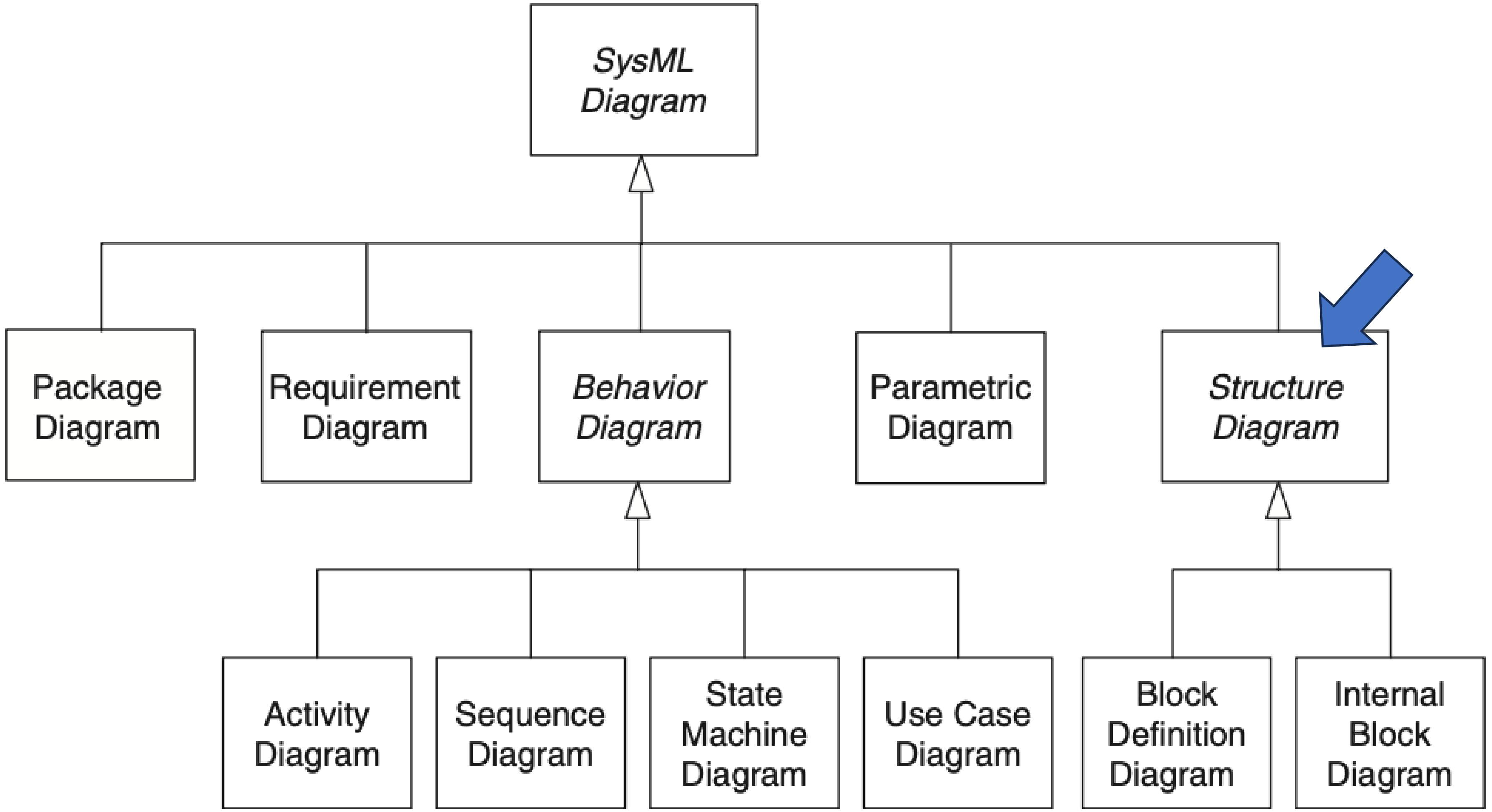


FIGURE 3.1

SysML diagram taxonomy.

Block Diagrams

MODELING STRUCTURE WITH BLOCKS – CHAPTER 7

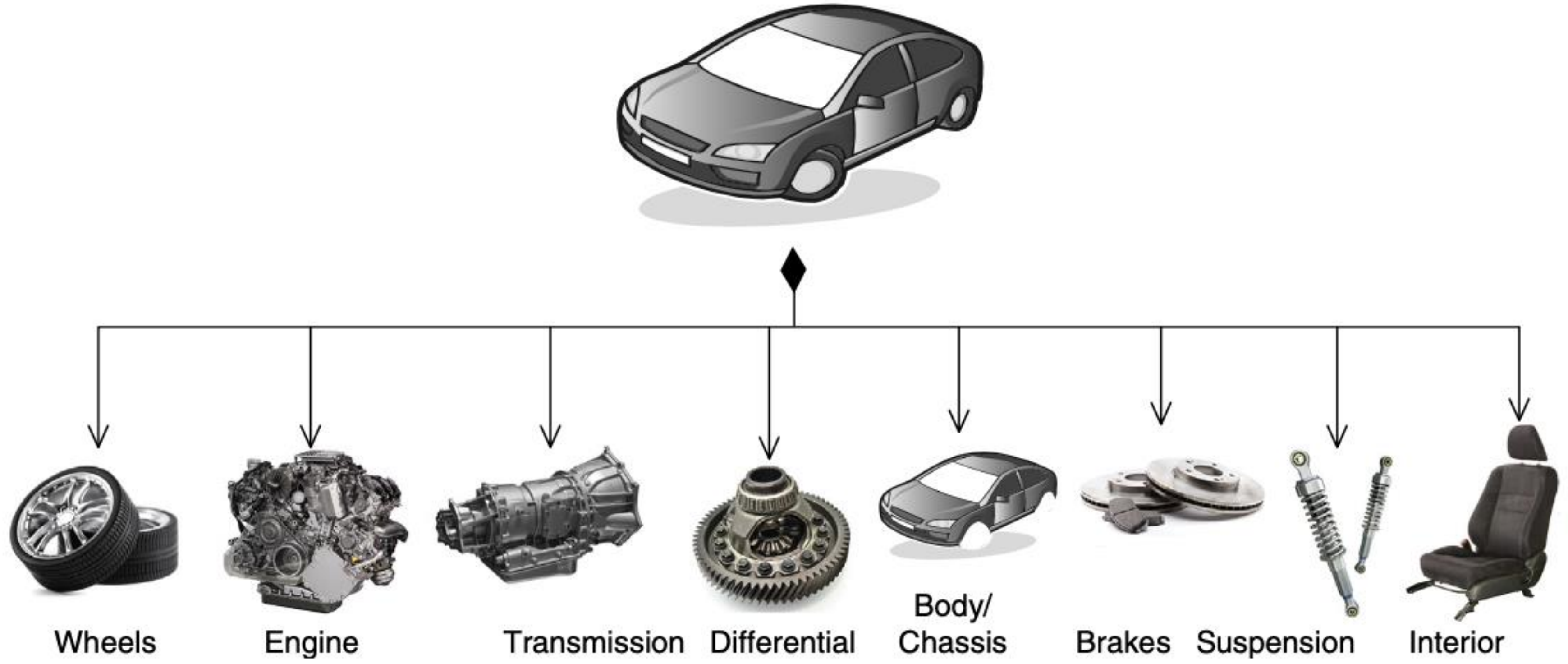


FIGURE 1.5

Automobile system decomposition into its components.

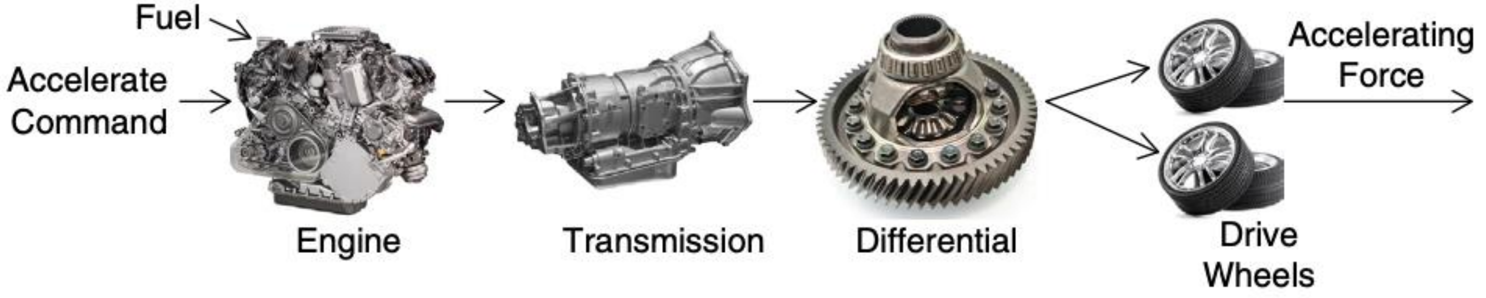


FIGURE 1.6

Interaction among components to achieve the system functional and performance requirements.

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

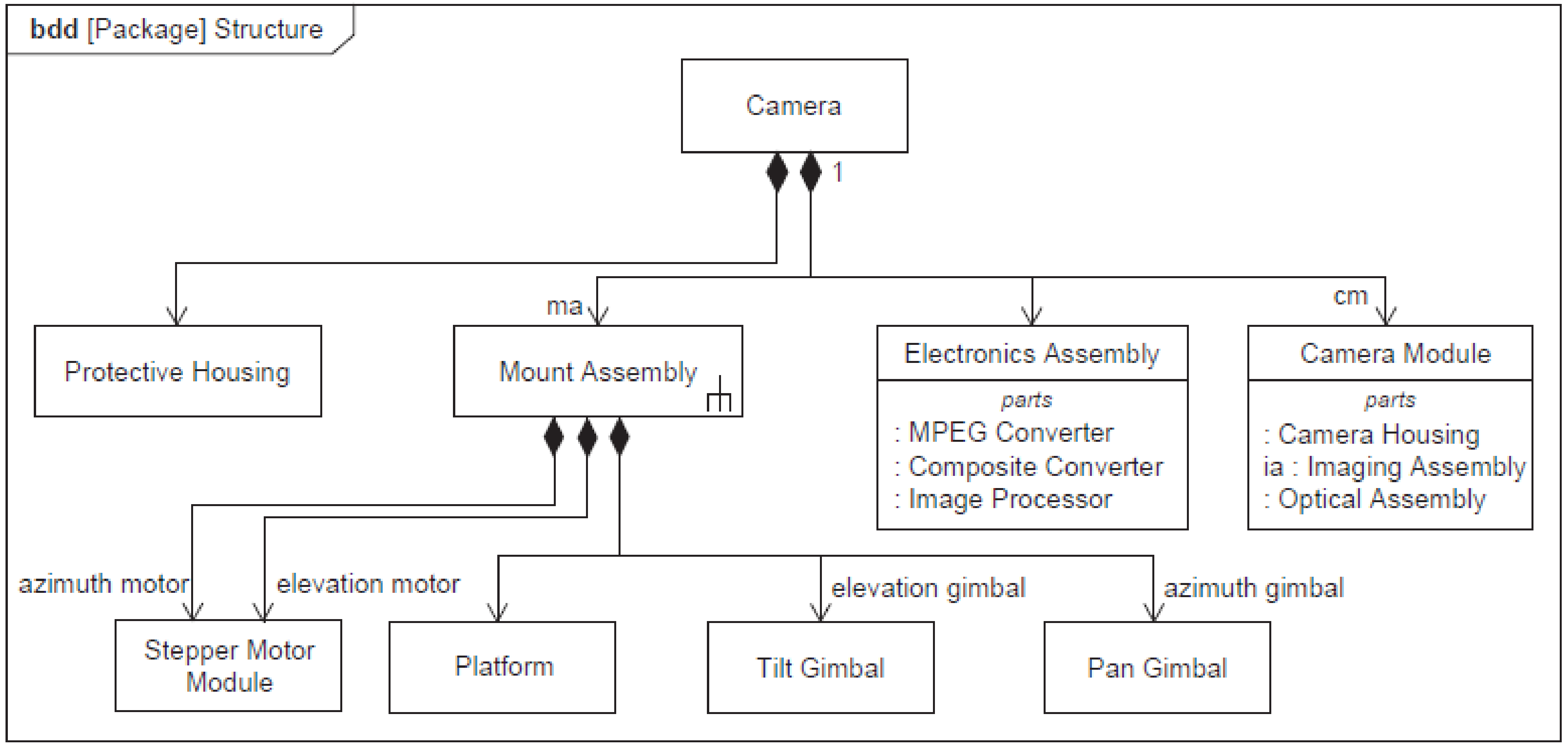


FIGURE 7.1

Example block definition diagram.

Example internal block diagram

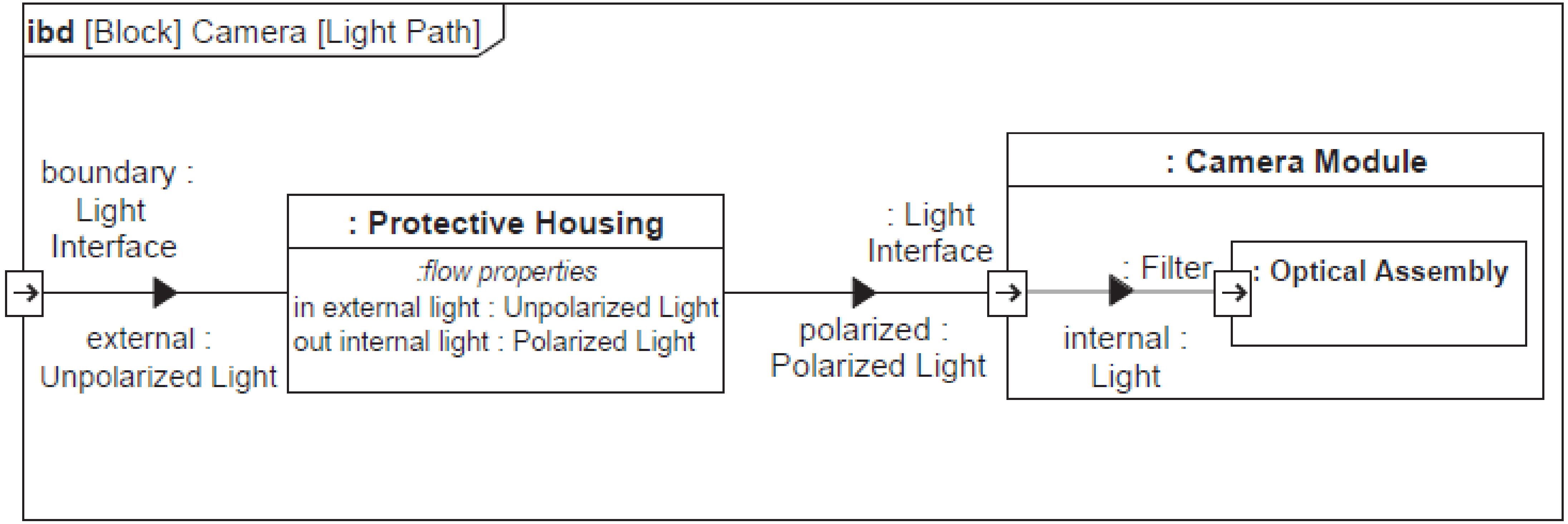

ibd [Block] Camera [Light Path]
boundary : Light Interface
external : Unpolarized Light
: Protective Housing
:flow properties
in external light : Unpolarized Light
out internal light : Polarized Light
: Light Interface
polarized : Polarized Light
: Camera Module
: Filter
internal : Light
: Optical Assembly

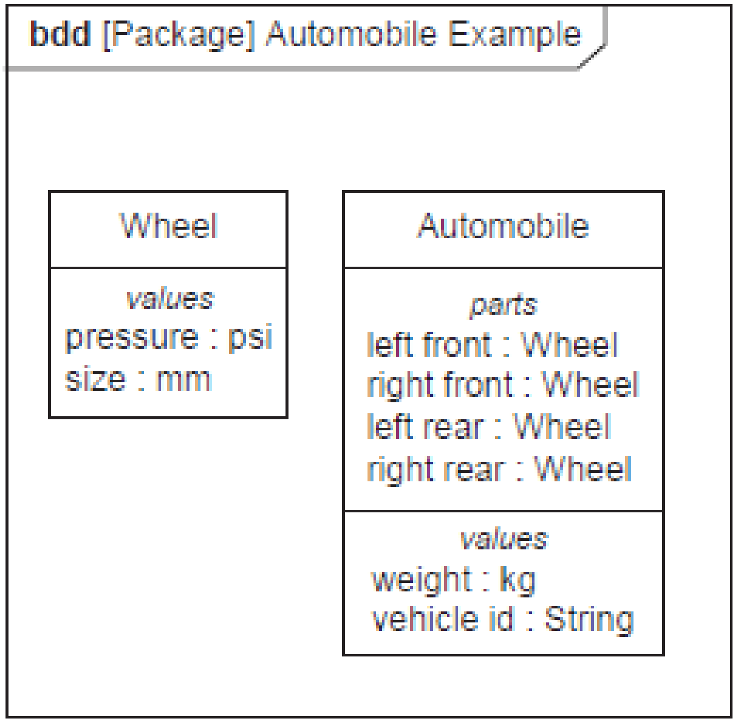
FIGURE 7.2

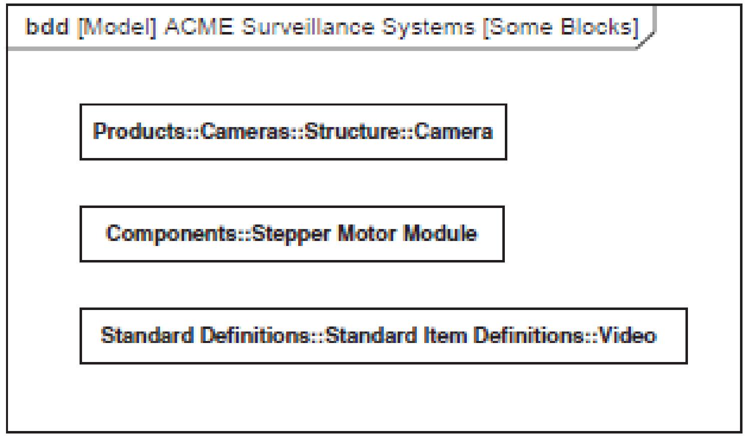
Example internal block diagram.

Block basics

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.*



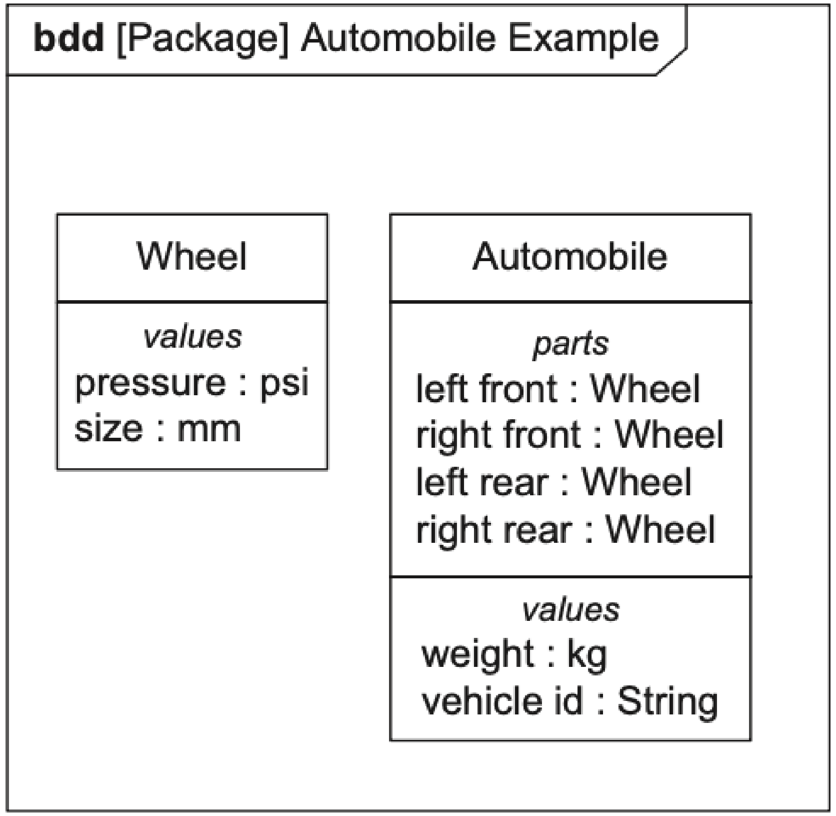

bdd [Model] ACME Surveillance Systems [Some Blocks]
Products::Cameras::Structure::Camera
Components::Stepper Motor Module
Standard Definitions::Standard Item Definitions::Video

Properties

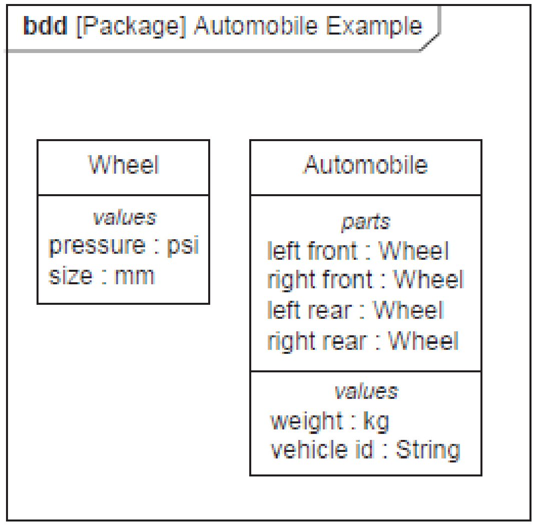
- Properties are structural features of a block. A property has a type that defines its characteristics, which may be another block, or some more basic type such as an integer.
  - **Part** properties (parts for short) describe the decomposition of a block into its constituent elements.
  - **Reference** 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

- **Parts** descrevem as relações de composição entre blocos.
  - Esse tipo de composição hierárquica de blocos é frequentemente visto em uma lista de materiais (também conhecida como lista de peças ou árvore de produtos).
  - Uma parte geralmente representa um bloco, embora também possa ser um ator.
  - A part é uma característica de um bloco e, como tal, pode ser listado em um compartimento de peças separado dentro de um bloco.
- Multiplicidade: 0..x (1-*)



Parts



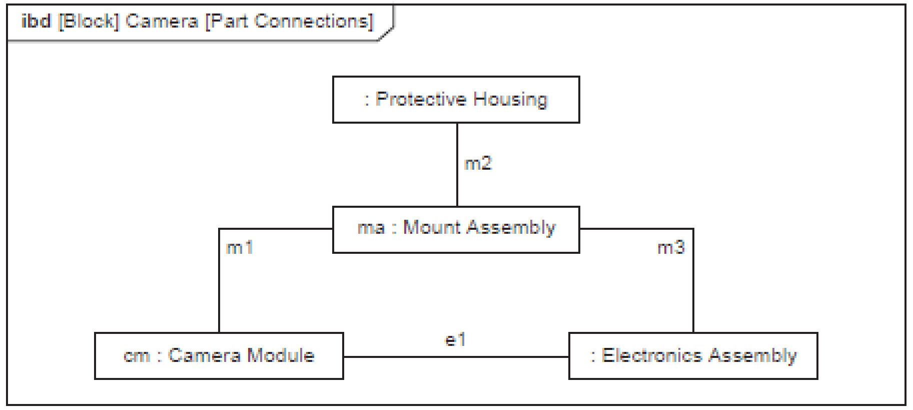


FIGURE 7.7

Connecting parts on an internal block diagram.

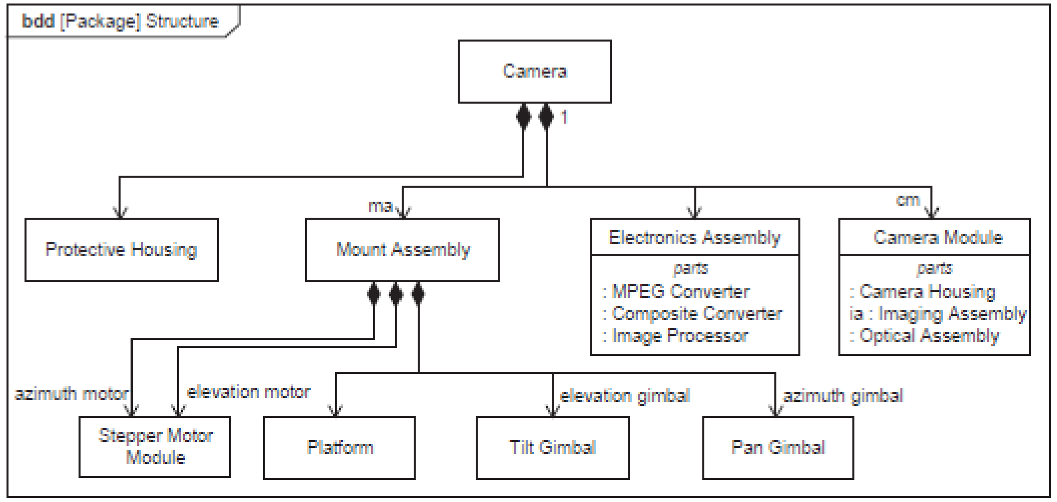


FIGURE 7.5

Showing a block composition hierarchy on a block definition diagram.

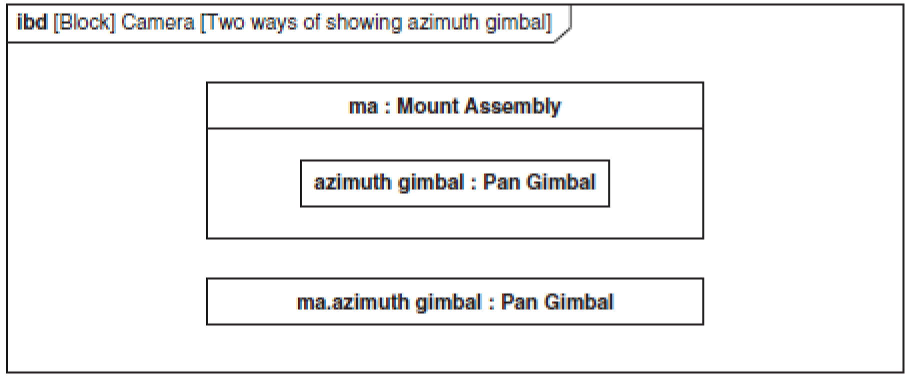

ibd [Block] Camera [Two ways of showing azimuth gimbal]
ma : Mount Assembly
azimuth gimbal : Pan Gimbal
ma.azimuth gimbal : Pan Gimbal

FIGURE 7.8

Showing deep-nested parts on an internal block diagram.

References

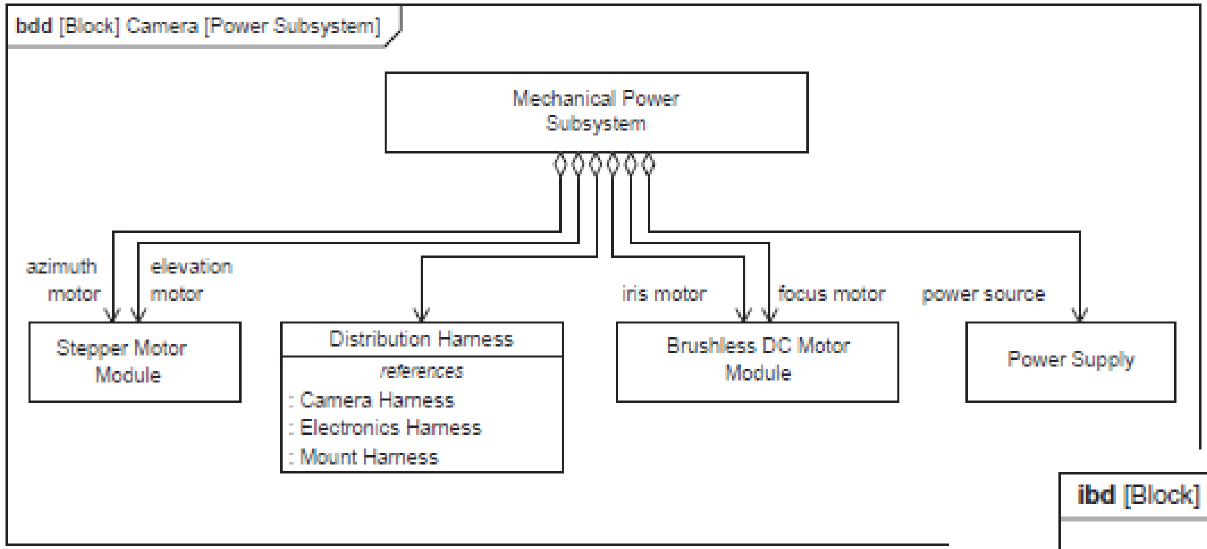


FIGURE 7.10

Reference associations on a block definition diagram.

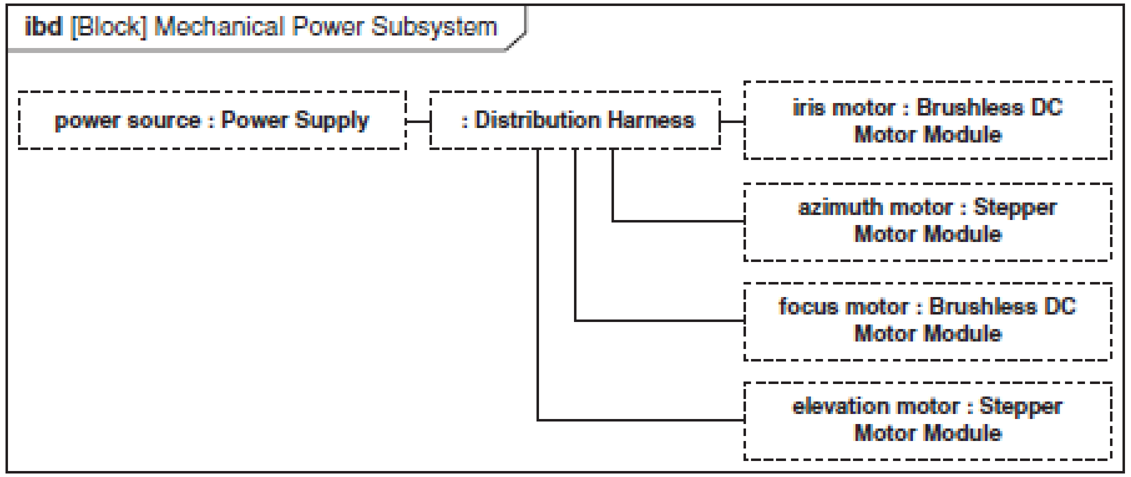

ibd [Block] Mechanical Power Subsystem
power source : Power Supply
: Distribution Harness
iris motor : Brushless DC Motor Module
azimuth motor : Stepper Motor Module
focus motor : Brushless DC Motor Module
elevation motor : Stepper Motor Module

FIGURE 7.11

Reference properties and their interconnections on an internal block diagram.

Values (Also called attribute)

- Value properties are used to model the quantitative characteristics of a block, such as its weight or velocity.
- The following are the different value type categories:
  - A **primitive** type supports the definition of scalar values. Integer, String, Boolean, and Real are predefined primitive types in SysML.
  - An **enumeration** defines a set of named values called literals. Examples of enumerations are colors and days of the week.
  - A **structured type** represents a specification of a data structure that includes more than one data element, each of which is represented by a value property.

Values

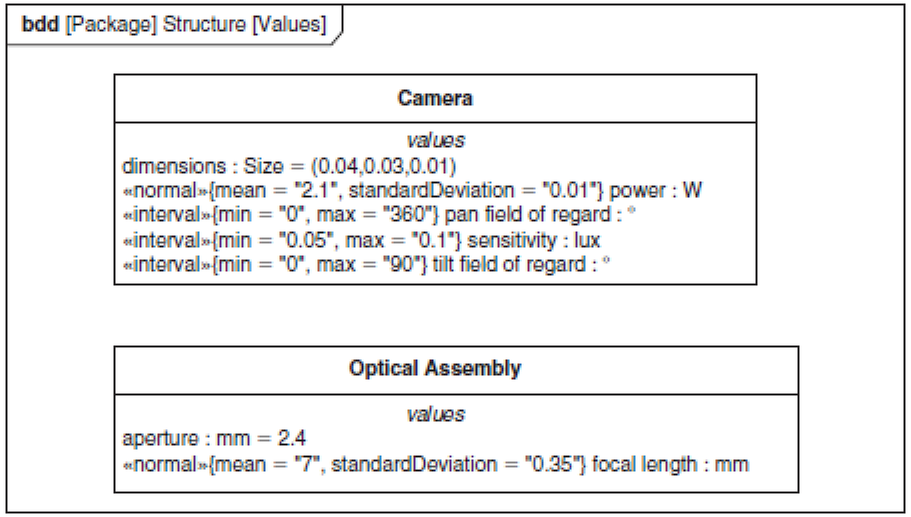

bdd [Package] Structure [Values]
Camera
values
dimensions : Size = (0.04,0.03,0.01)
«normal»{mean = "2.1", standardDeviation = "0.01"} power : W
«interval»{min = "0", max = "360"} pan field of regard : °
«interval»{min = "0.05", max = "0.1"} sensitivity : lux
«interval»{min = "0", max = "90"} tilt field of regard : °
Optical Assembly
values
aperture : mm = 2.4
«normal»{mean = "7", standardDeviation = "0.35"} focal length : mm

FIGURE 7.22

Examples of property values and distributions.

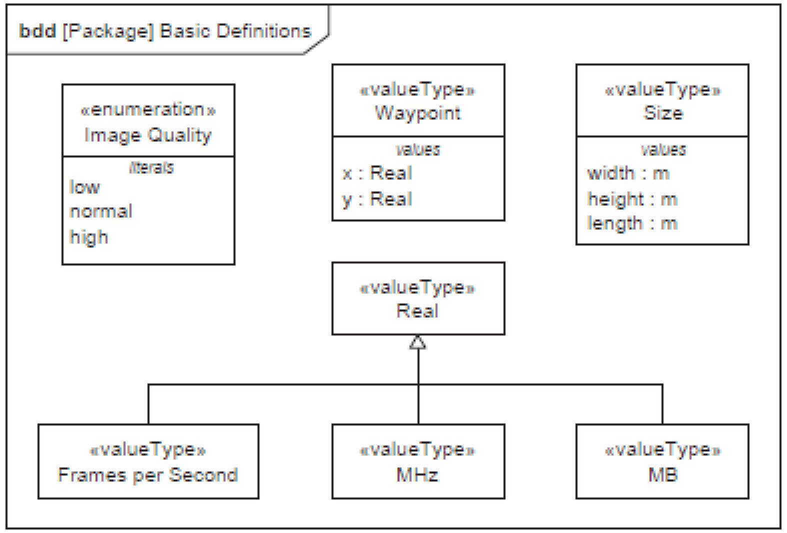


FIGURE 7.17

Definition of basic value types in a block definition diagram.

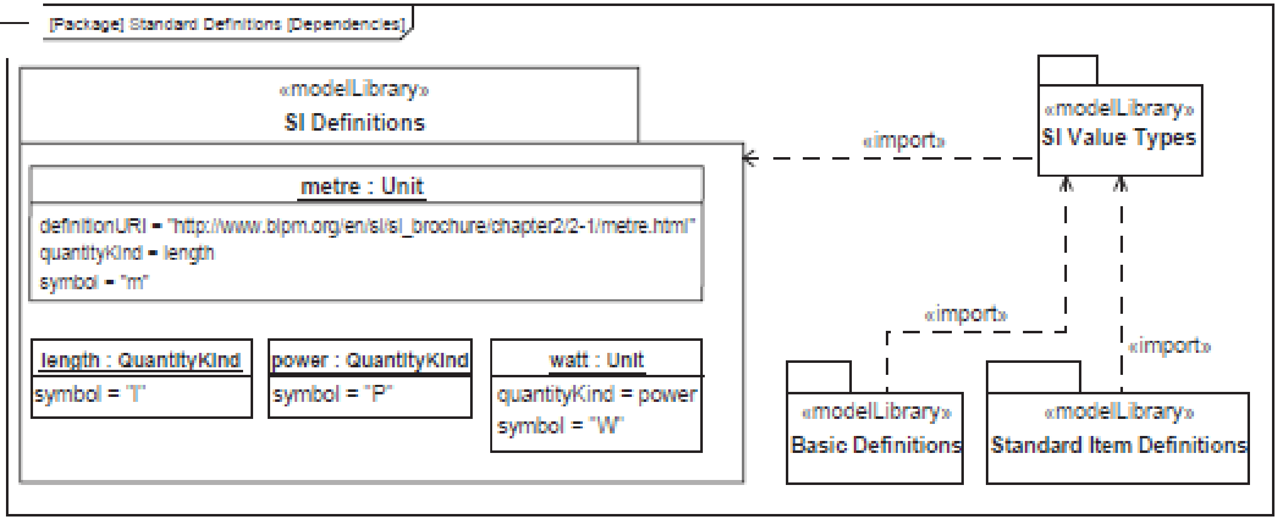

[Package] Standard Definitions [Dependencies]
«modelLibrary» SI Definitions
metre : Unit
definitionURI = "http://www.bipm.org/en/si/si_brochure/chapter2/2-1/metre.html"
quantityKind = length
symbol = "m"
length : QuantityKind
symbol = "l"
power : QuantityKind
symbol = "P"
watt : Unit
quantityKind = power
symbol = "W"
«import»
«modelLibrary» SI Value Types
«import»
«import»
«modelLibrary» Basic Definitions
«modelLibrary» Standard Item Definitions

FIGURE 7.18

Importing the SI definitions defined by SysML.

Behaviour

Behaviors

- Blocks provide a context for behaviors, which encompasses any and all descriptions of **how the block handles inputs, outputs, and changes to its internal state**.
- A block can designate a behavior as its primary behavior (classifier).
- These behaviors will be used in activities/state machines and interactions.
  - **Activities** turn inputs into outputs.
  - **State machines** are used to describe how the block responds to events.
  - **Sequences** describe how parts of a block interact with each other using message passing

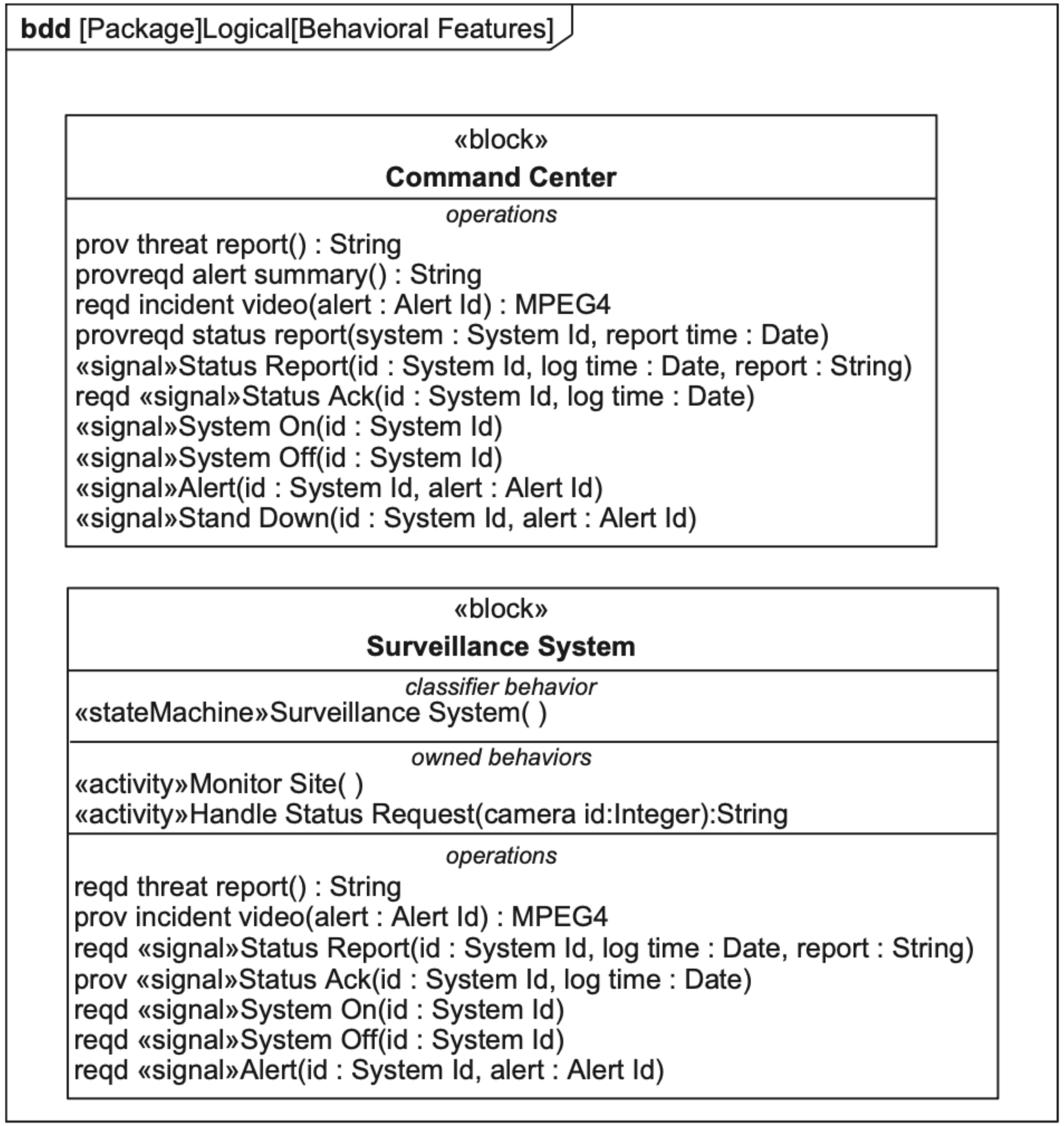

bdd [Package]Logical[Behavioral Features]
«block»
Command Center
operations
prov threat report() : String
provreqd alert summary() : String
reqd incident video(alert : Alert Id) : MPEG4
provreqd status report(system : System Id, report time : Date)
«signal»Status Report(id : System Id, log time : Date, report : String)
reqd «signal»Status Ack(id : System Id, log time : Date)
«signal»System On(id : System Id)
«signal»System Off(id : System Id)
«signal»Alert(id : System Id, alert : Alert Id)
«signal»Stand Down(id : System Id, alert : Alert Id)
«block»
Surveillance System
classifier behavior
«stateMachine»Surveillance System( )
owned behaviors
«activity»Monitor Site( )
«activity»Handle Status Request(camera id:Integer):String
operations
reqd threat report() : String
prov incident video(alert : Alert Id) : MPEG4
reqd «signal»Status Report(id : System Id, log time : Date, report : String)
prov «signal»Status Ack(id : System Id, log time : Date)
reqd «signal»System On(id : System Id)
reqd «signal»System Off(id : System Id)
reqd «signal»Alert(id : System Id, alert : Alert Id)

FIGURE 7.30

Blocks with behavioral features.

Block Relationships

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

- A composition relates two blocks in a relationship of part to whole.
  - It has two ends, one describing the whole and the other describing the part.
  - The upper bound of multiplicity at the integer end is always 1 because an instance of a part can only exist in a whole at any given time.
- A composition is shown as a line between two blocks; the end of a composite association is adorned by a **black diamond**.
- Each end of the composition can show **a name and a multiplicity**.

Composition example

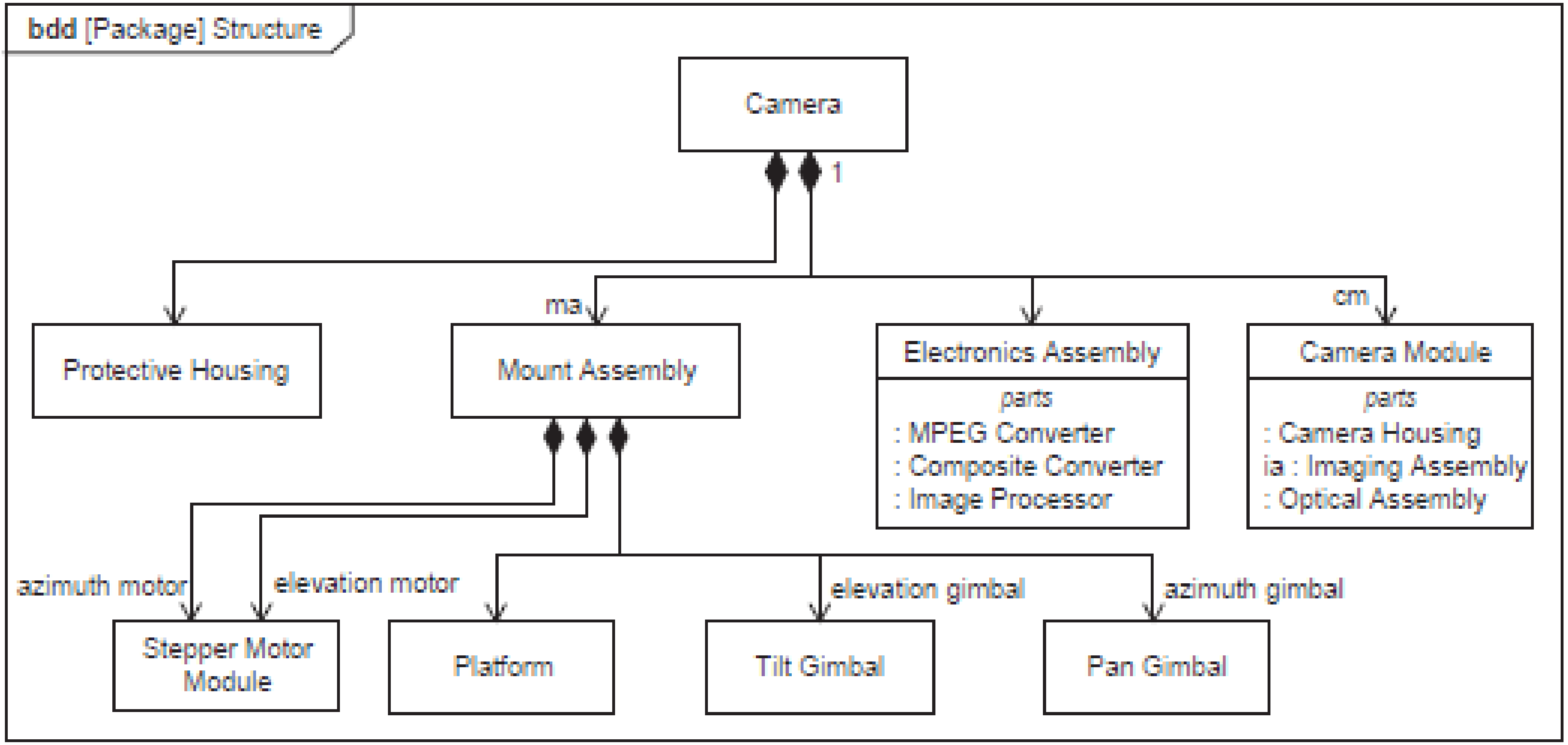


FIGURE 7.5

Showing a block composition hierarchy on a block definition diagram.

Reference

- References, enable an instance of a block that contains the reference property to refer to an instance of the block that types the reference property.
- Reference associations are used in a block definition diagram to capture relationship between blocks.
- A reference association is represented as a line between two blocks. One end of an association can be represented by a **white diamond** (aggregation).
  - **When there is a reference property at only one end, the line has an open arrowhead at the end of the binding pointing** from the owner of the reference property to the referenced type.
  - There **is no arrowhead** at the end of the binding that owns the reference property.
  - If the reference binding is **bidirectional** (that is, it has reference properties at both ends), **there are no arrowheads at either end**.

Reference example

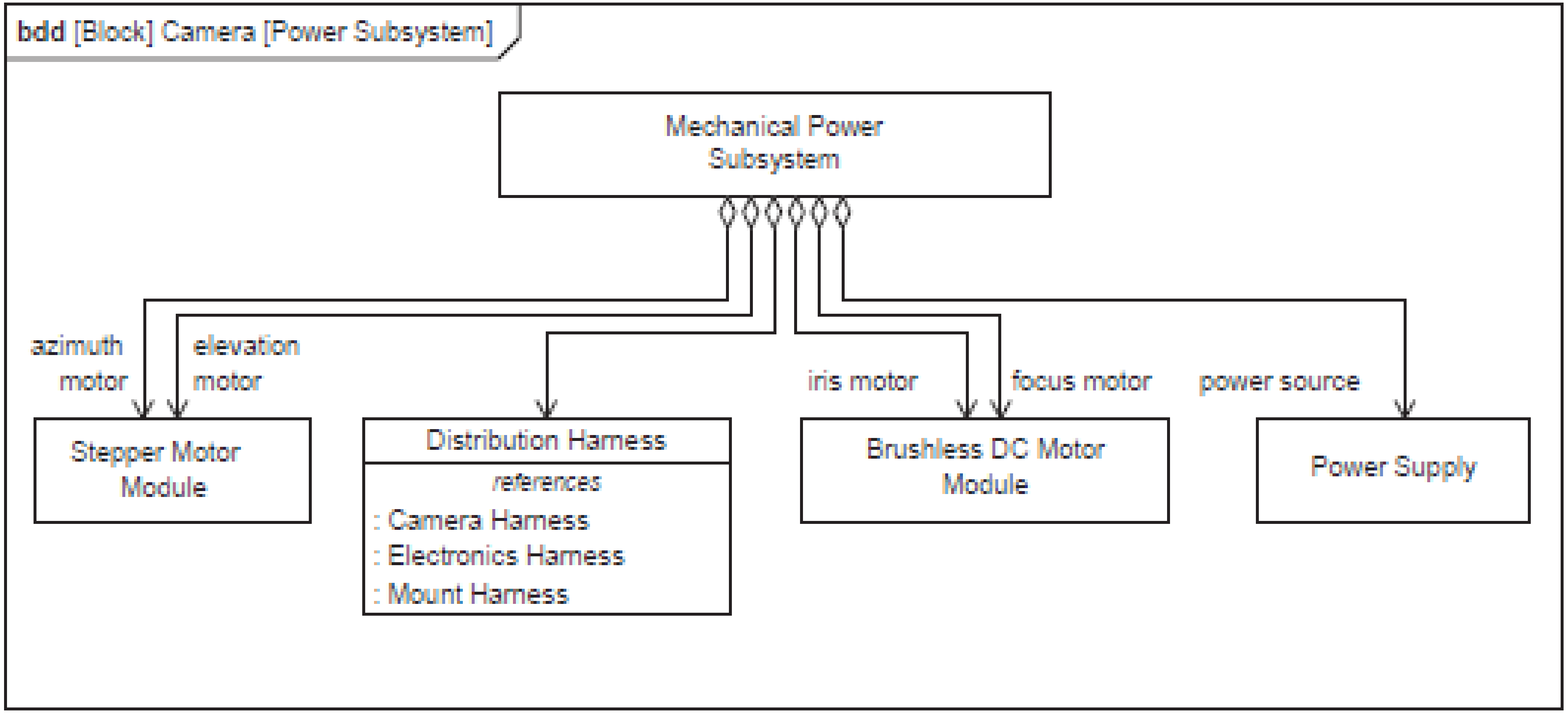


FIGURE 7.10

Reference associations on a block definition diagram.

Heritage

- In SysML, a classifier is a type that can be used as the basis for more specific types.
- A more specialized classifier will inherit the common features of the more general classifier and may contain additional features unique to it.
- The relationship between the general classifier and the specialized classifier is called generalization or specialization.
- Different blocks in a classification hierarchy can have different structural characteristics.
- When a resource in a superclass is reset into a subclass, the original resource in the superclass is no longer available to the subclass.

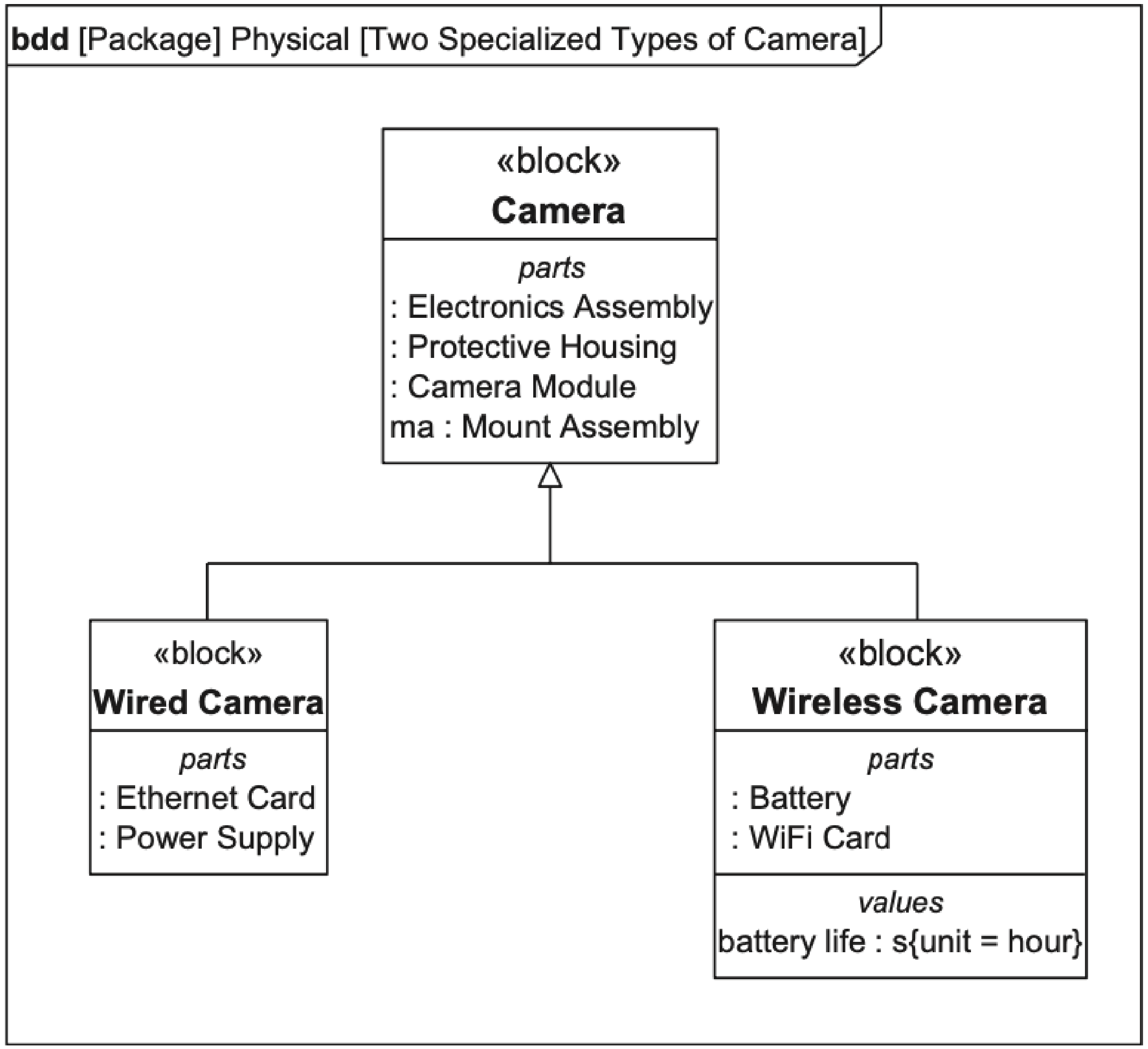


FIGURE 7.48

Example of block specialization.

Properties (values) especialization

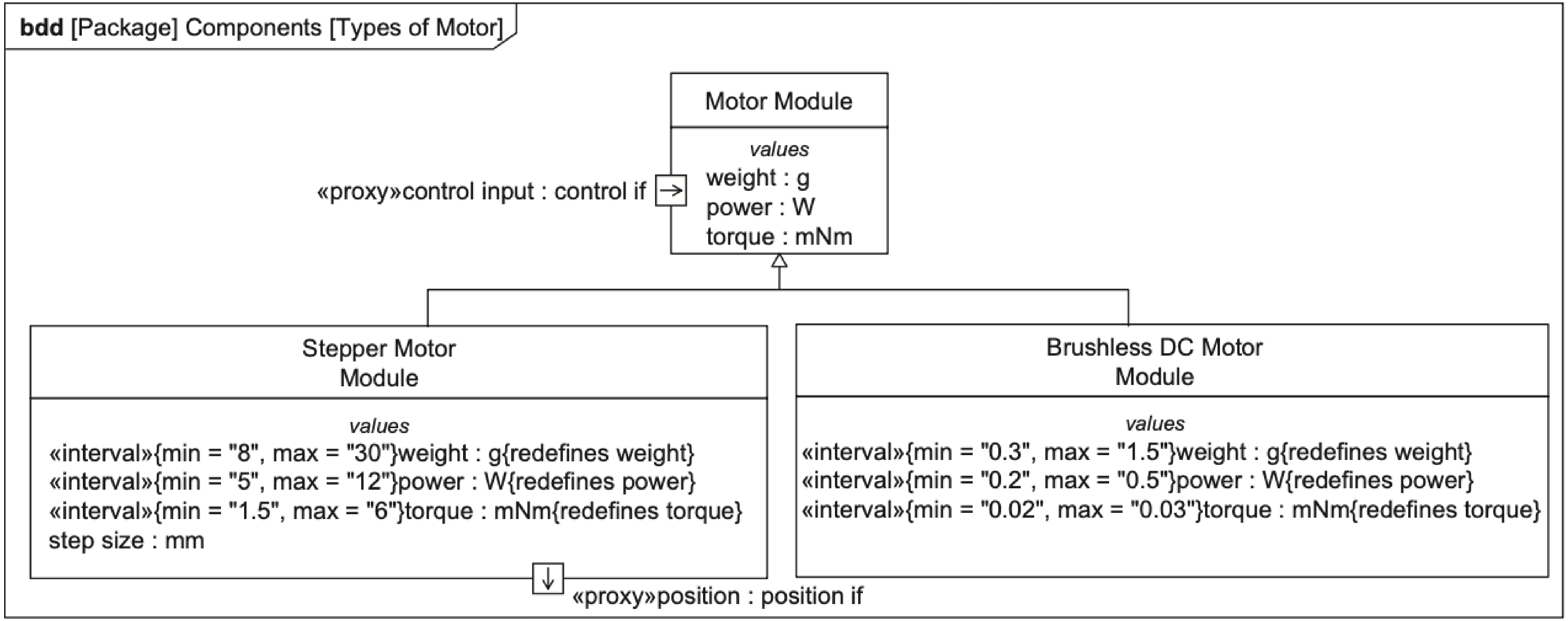

bdd [Package] Components [Types of Motor]
Motor Module
values
weight : g
power : W
torque : mNm
«proxy»control input : control if
Stepper Motor Module
values
«interval»{min = "8", max = "30"}weight : g{redefines weight}
«interval»{min = "5", max = "12"}power : W{redefines power}
«interval»{min = "1.5", max = "6"}torque : mNm{redefines torque}
step size : mm
«proxy»position : position if
Brushless DC Motor Module
values
«interval»{min = "0.3", max = "1.5"}weight : g{redefines weight}
«interval»{min = "0.2", max = "0.5"}power : W{redefines power}
«interval»{min = "0.02", max = "0.03"}torque : mNm{redefines torque}

FIGURE 7.49

Showing a classification hierarchy on a block definition diagram.

Polymorphism example

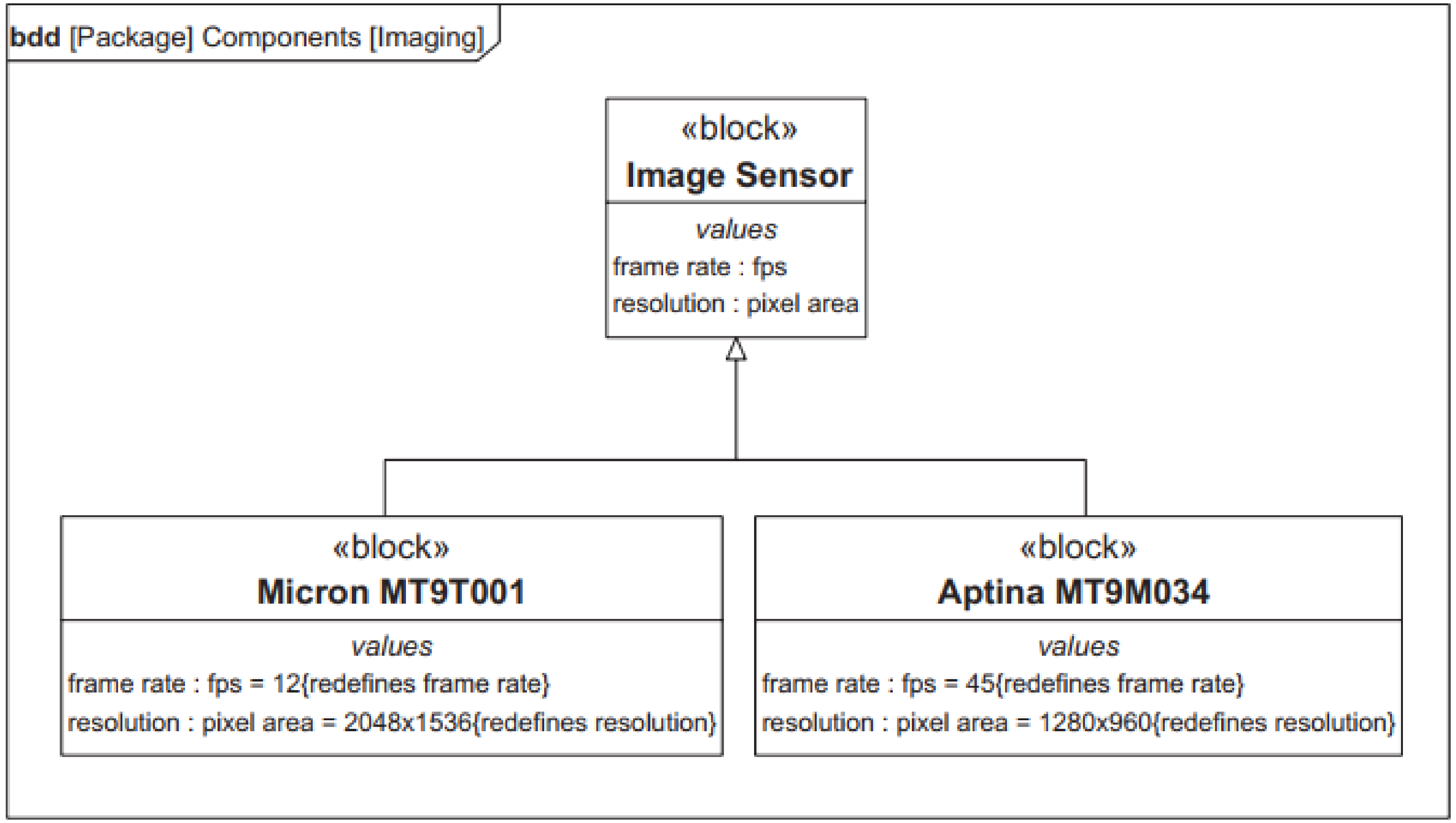
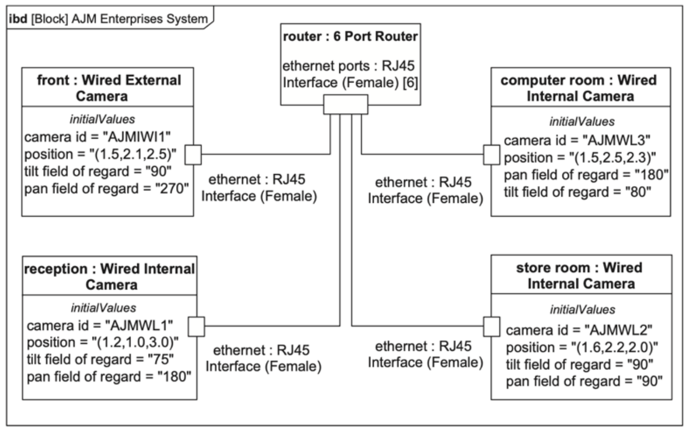

bdd [Package] Components [Imaging]
«block»
Image Sensor
values
frame rate : fps
resolution : pixel area
«block»
Micron MT9T001
values
frame rate : fps = 12{redefines frame rate}
resolution : pixel area = 2048x1536{redefines resolution}
«block»
Aptina MT9M034
values
frame rate : fps = 45{redefines frame rate}
resolution : pixel area = 1280x960{redefines resolution}

FIGURE 7.52

Two kinds of Imaging Assembly.

Block Configuration

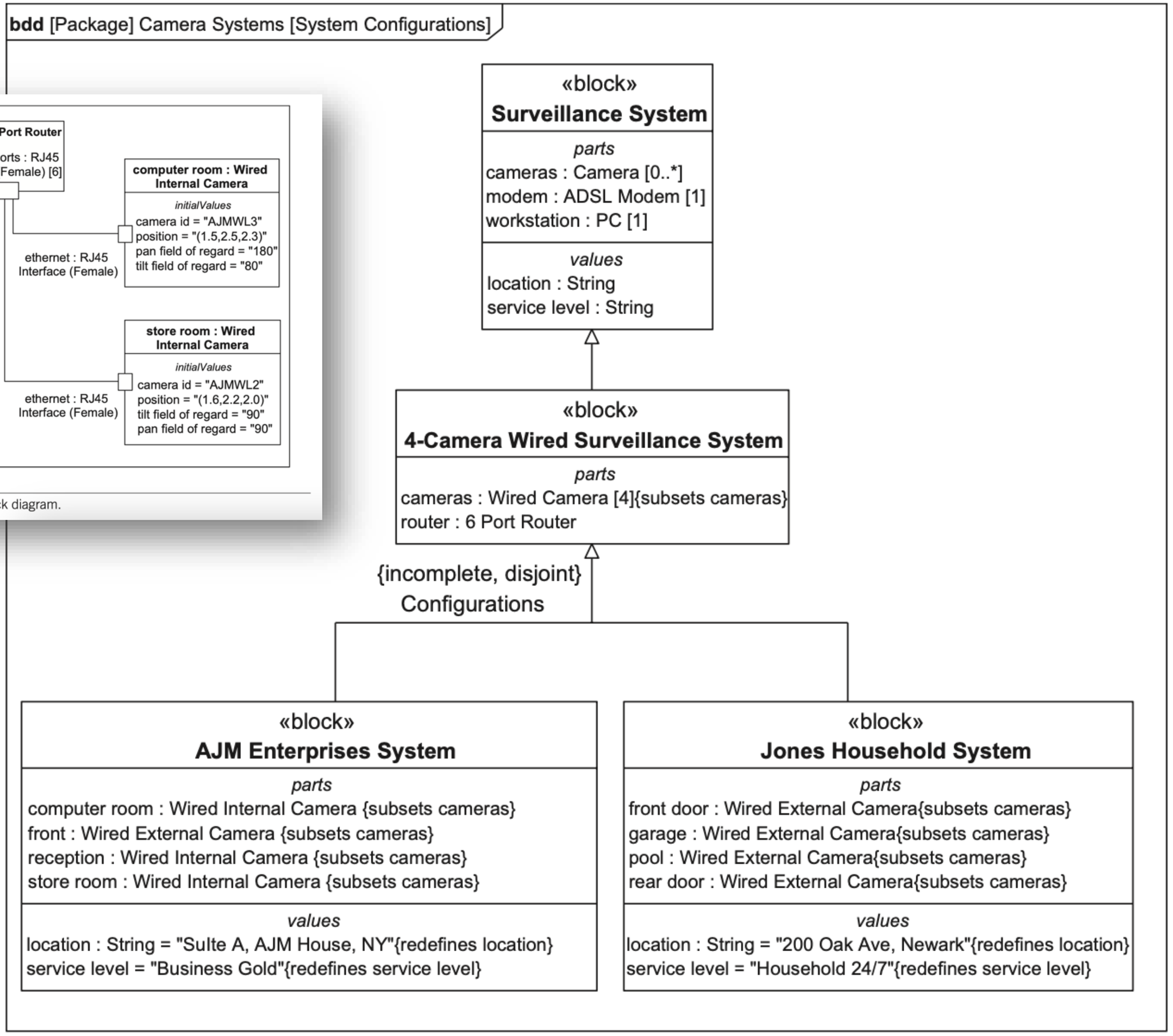
- A block configuration is constructed using the generalization relationship.
- The configuration becomes a subclass of the block for which it is a configuration.



ibd [Block] AJM Enterprises System
router : 6 Port Router
ethernet ports : RJ45 Interface (Female) [6]
front : Wired External Camera
initialValues
camera id = "AJMIWI1"
position = "(1.5,2.1,2.5)"
tilt field of regard = "90"
pan field of regard = "270"
computer room : Wired Internal Camera
initialValues
camera id = "AJMWL3"
position = "(1.5,2.5,2.3)"
pan field of regard = "180"
tilt field of regard = "80"
reception : Wired Internal Camera
initialValues
camera id = "AJMWL1"
position = "(1.2,1.0,3.0)"
tilt field of regard = "75"
pan field of regard = "180"
store room : Wired Internal Camera
initialValues
camera id = "AJMWL2"
position = "(1.6,2.2,2.0)"
tilt field of regard = "90"
pan field of regard = "90"
ethernet : RJ45 Interface (Female)
ethernet : RJ45 Interface (Female)
ethernet : RJ45 Interface (Female)
ethernet : RJ45 Interface (Female)

FIGURE 7.59

Showing the configuration of a block on an internal block diagram.



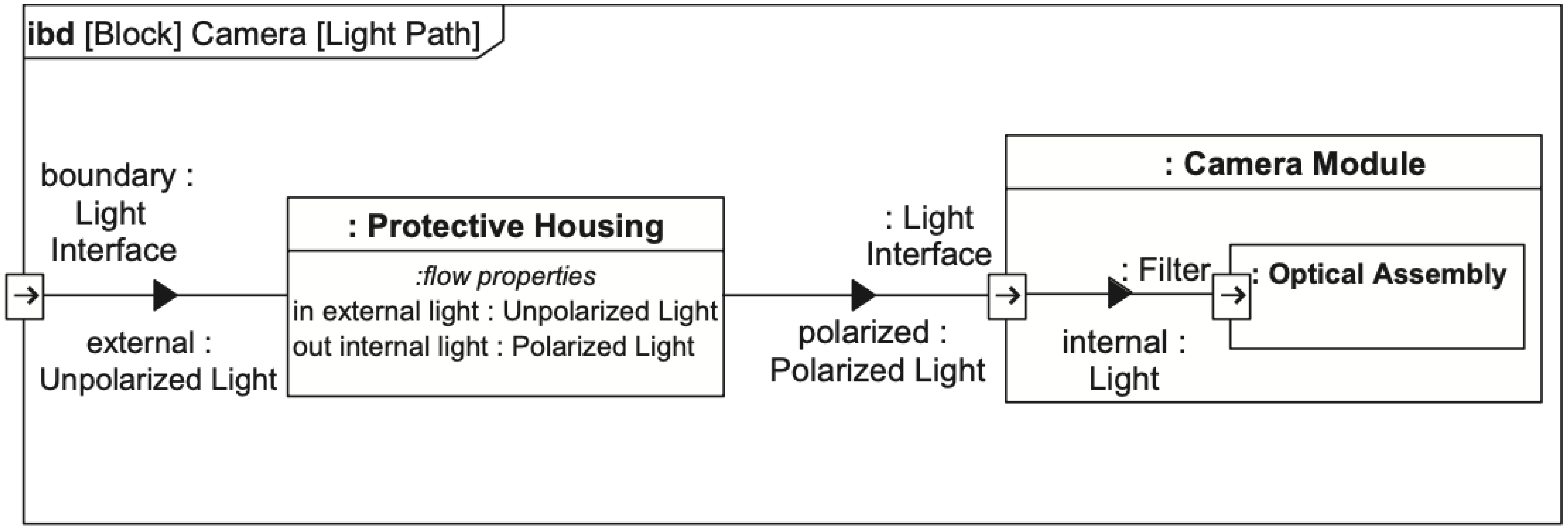
bdd [Package] Camera Systems [System Configurations]
«block»
Surveillance System
parts
cameras : Camera [0..*]
modem : ADSL Modem [1]
workstation : PC [1]
values
location : String
service level : String
«block»
4-Camera Wired Surveillance System
parts
cameras : Wired Camera [4]{subsets cameras}
router : 6 Port Router
{incomplete, disjoint}
Configurations
«block»
AJM Enterprises System
parts
computer room : Wired Internal Camera {subsets cameras}
front : Wired External Camera {subsets cameras}
reception : Wired Internal Camera {subsets cameras}
store room : Wired Internal Camera {subsets cameras}
values
location : String = "Sulte A, AJM House, NY"{redefines location}
service level = "Business Gold"{redefines service level}
«block»
Jones Household System
parts
front door : Wired External Camera{subsets cameras}
garage : Wired External Camera{subsets cameras}
pool : Wired External Camera{subsets cameras}
rear door : Wired External Camera{subsets cameras}
values
location : String = "200 Oak Ave, Newark"{redefines location}
service level = "Household 24/7"{redefines service level}

FIGURE 7.58

Modeling different configurations of a block on a block definition diagram.

Block Internal Interconnection (ibd)

Internal Block Diagram



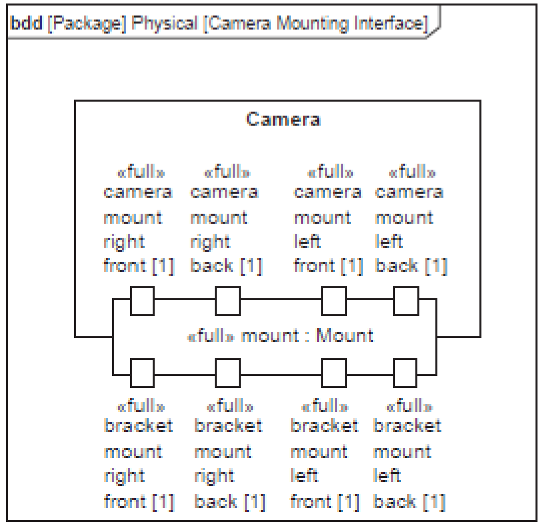
ibd [Block] Camera [Light Path]
boundary : Light Interface
external : Unpolarized Light
: Protective Housing
:flow properties
in external light : Unpolarized Light
out internal light : Polarized Light
: Light Interface
polarized : Polarized Light
: Camera Module
: Filter
internal : Light
: Optical Assembly

FIGURE 7.2

Example internal block diagram.

Ports

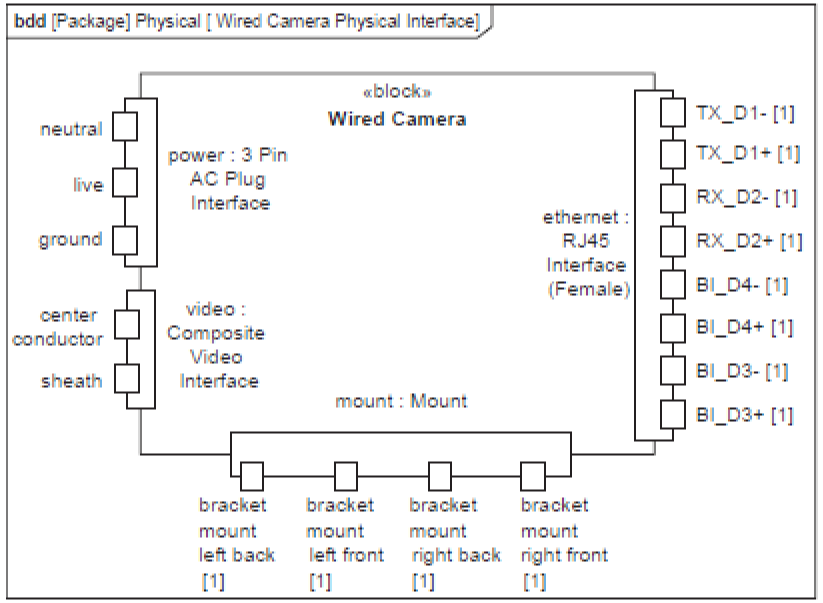
- A port represents an access point on the **boundary of a block** and on the boundary of any part or reference typed by that block.
- A block may have many ports that specify different access points. Ports can be connected to one another by connectors on an internal block diagram to support the interaction 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**.



bdd [Package] Physical [Camera Mounting Interface]
Camera
«full» camera mount right front [1]
«full» camera mount right back [1]
«full» camera mount left front [1]
«full» camera mount left back [1]
«full» mount : Mount
«full» bracket mount right front [1]
«full» bracket mount right back [1]
«full» bracket mount left front [1]
«full» bracket mount left back [1]

FIGURE 7.32

A full port with nested ports.



bdd [Package] Physical [Wired Camera Physical Interface]
«block»
Wired Camera
neutral
live
ground
power : 3 Pin AC Plug Interface
center conductor
sheath
video : Composite Video Interface
ethernet : RJ45 Interface (Female)
TX_D1- [1]
TX_D1+ [1]
RX_D2- [1]
RX_D2+ [1]
BI_D4- [1]
BI_D4+ [1]
BI_D3- [1]
BI_D3+ [1]
mount : Mount
bracket mount left back [1]
bracket mount left front [1]
bracket mount right back [1]
bracket mount right front [1]

FIGURE 7.34

A block with nested ports.

Conector

- A **connector** is used to connect two parts and provides the opportunity for those parts to interact, although the connector alone says nothing about the nature of the interaction.
- The **interaction may include the flow of inputs and outputs between parts**, the invocation of operations on parts, or the sending and receiving of signals between parts, or it may be specified by constraints on the properties of the parts at each end.
- **A connector can have an association block that allows for further definition of the characteristics of the connection**.
- In an ibd, the connector between two parts is represented as a line connecting two part symbols.

Connecting ports

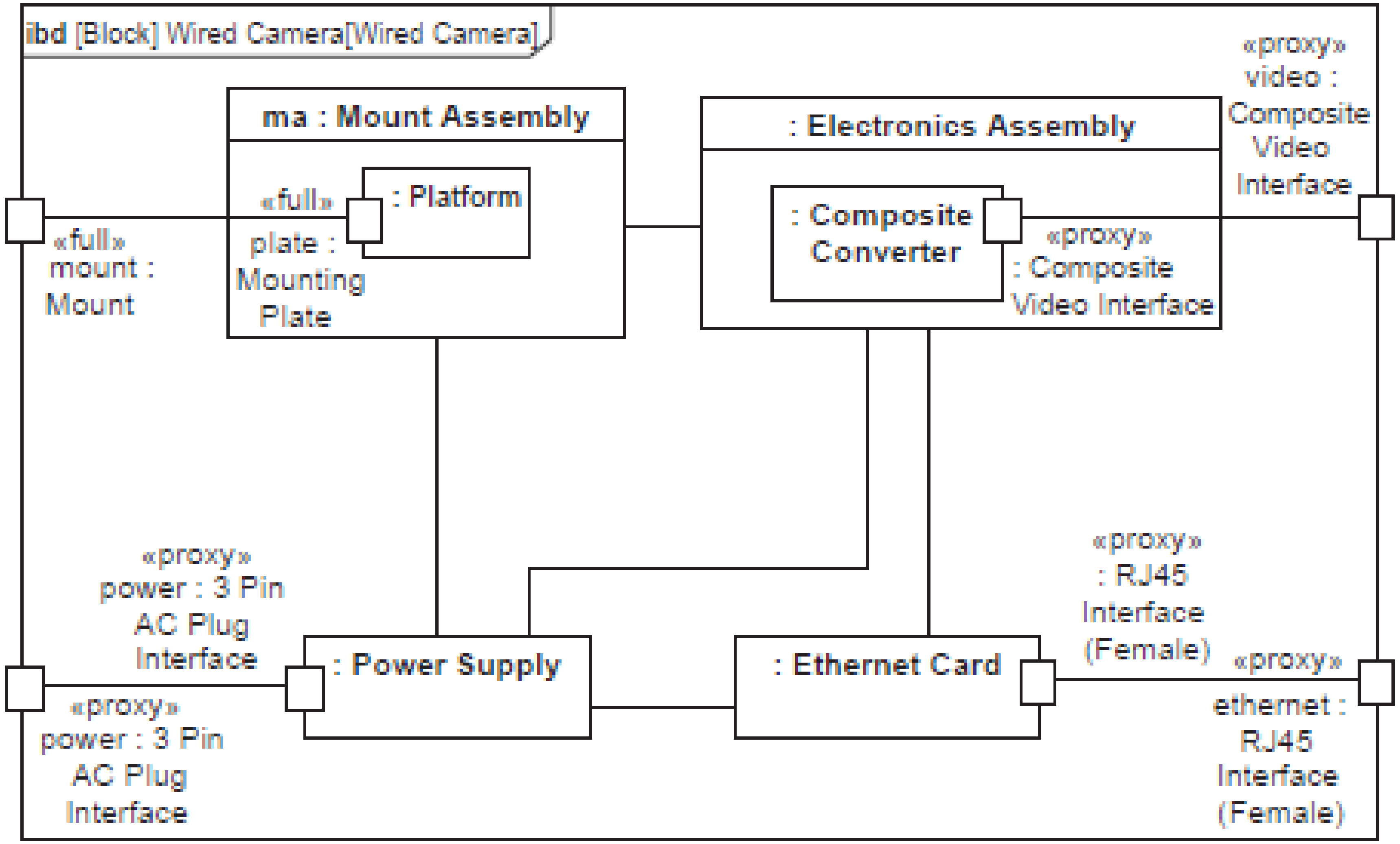

ibd [Block] Wired Camera[Wired Camera]
ma : Mount Assembly
«full»
: Platform
plate : Mounting Plate
«full» mount : Mount
: Electronics Assembly
: Composite Converter
«proxy» : Composite Video Interface
«proxy» video : Composite Video Interface
«proxy» power : 3 Pin AC Plug Interface
: Power Supply
«proxy» power : 3 Pin AC Plug Interface
: Ethernet Card
«proxy» : RJ45 Interface (Female)
«proxy» ethernet : RJ45 Interface (Female)

FIGURE 7.35

Connecting ports internally to a block.

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.

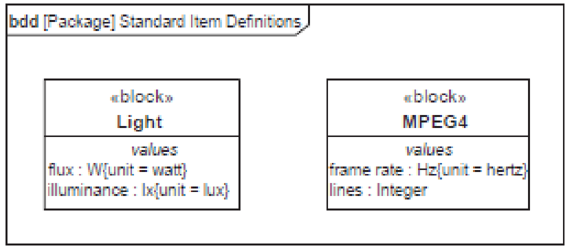

bdd [Package] Standard Item Definitions
«block»
Light
values
flux : W{unit = watt}
illuminance : lx{unit = lux}
«block»
MPEG4
values
frame rate : Hz{unit = hertz}
lines : Integer

FIGURE 7.23
Items that flow in the *Camera* system.

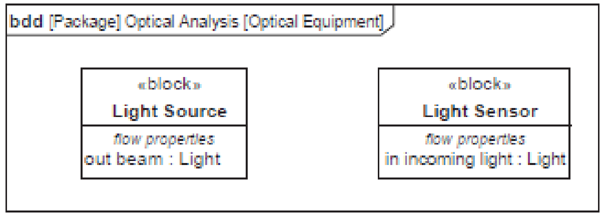


FIGURE 7.24
Flow properties on blocks.

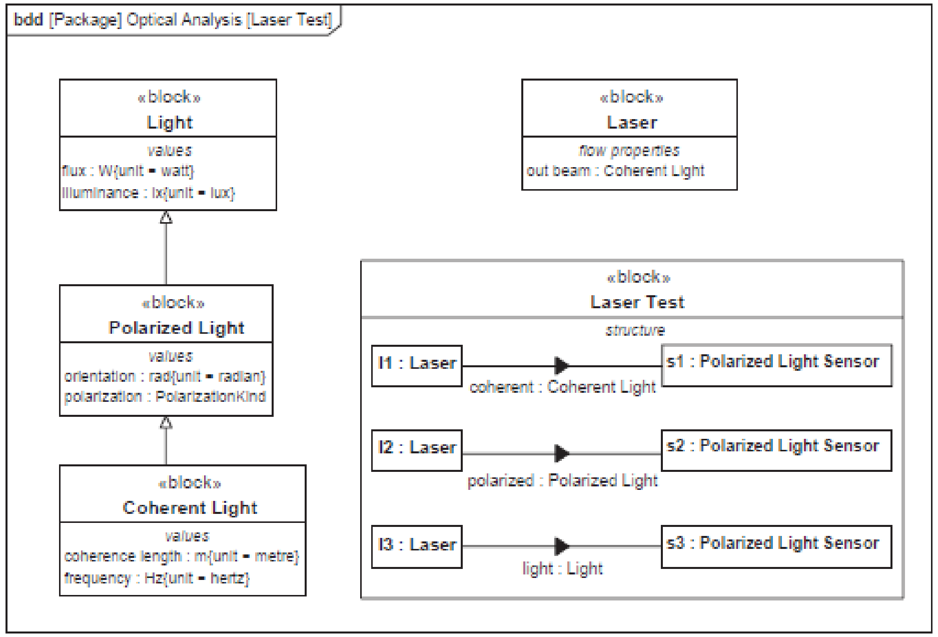

bdd [Package] Optical Analysis [Laser Test]
«block»
Light
values
flux : W{unit = watt}
illuminance : lx{unit = lux}
«block»
Polarized Light
values
orientation : rad{unit = radian}
polarization : PolarizationKind
«block»
Coherent Light
values
coherence length : m{unit = metre}
frequency : Hz{unit = hertz}
«block»
Laser
flow properties
out beam : Coherent Light
«block»
Laser Test
structure
l1 : Laser
coherent : Coherent Light
s1 : Polarized Light Sensor
l2 : Laser
polarized : Polarized Light
s2 : Polarized Light Sensor
l3 : Laser
light : Light
s3 : Polarized Light Sensor

FIGURE 7.27
Item flows between parts.

Item flows between ports.

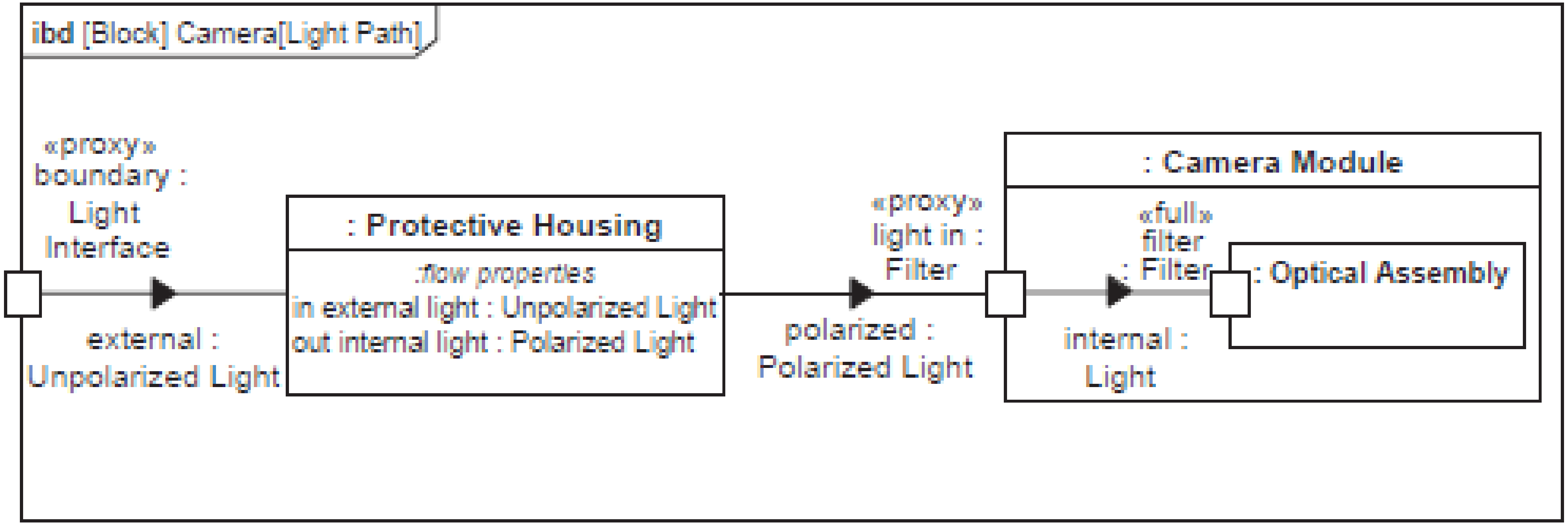

ibd [Block] Camera[Light Path]
«proxy» boundary : Light Interface
external : Unpolarized Light
: Protective Housing
:flow properties
in external light : Unpolarized Light
out internal light : Polarized Light
«proxy» light in : Filter
polarized : Polarized Light
: Camera Module
«full» filter : Filter
: Optical Assembly
internal : Light

FIGURE 7.44

Item flows between ports.