COLLEGE of CHARLESTON

Systems *Engineering: Design and Development*

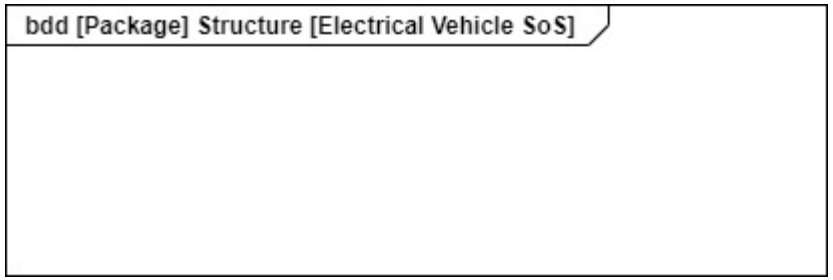
ENGR 387

Agenda

- **Purpose and Use of Blocks and Block Definition Diagrams**
- **Purpose of Use of Block features**
- **Purpose and Use of Association Relationships**
- **Purpose and Use of Generalization Relationships**
- **Purpose and Use of Dependency Relationships**
- **Purpose and Use of Actors**
- **Purpose and Use of Value Types**
- **Purpose and Use of Constraint Blocks**
- **Purpose and Use of Comments**
- **Default Multiplicities**

Purpose and Use of Block Definition Diagrams

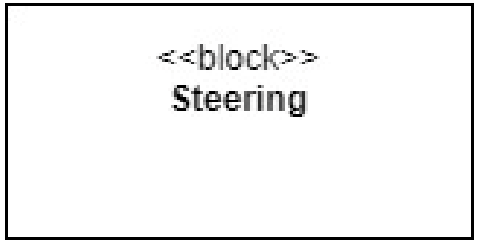
- BDD: express information about a system's structure
- BDD model elements: blocks, actors, value types, constraint blocks, flow specifications, & interfaces
  - Elements that appear on a BDD = elements of definition
- BDDs display the structural relationships among elements of definition



- The BDD frame
  - Diagram kind abbreviation: bdd
  - Model element type that the diagram frame represents: package, model, modelLibrary, view, block, constraintBlock
  - Model element the diagram represents: namespace (model element that contains other model elements)

Purpose and Use of Blocks and Block Definition Diagrams

- Elements of Definition vs. Elements of Usage
  - Definition of types
    - Notation: name only
      - *DesktopWorkstation*
  - Usage (instances) of types
    - Notation: name and type; separate by colon
      - *SDX1205LJD : DesktopWorkstation*
- Block: basic unit of structure in SYSML
  - Represents a type of entity, not an instance
- Block notation: rectangle with stereotype <<block>> preceding the name in the name compartment

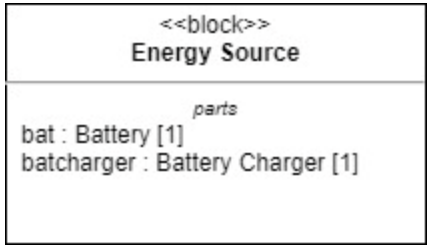


Purpose and Use of Block Features

- Features come in two varieties: **Structural** and **Behavioral**
- 5 kinds of structural features (aka properties)
  - Part properties
  - Reference properties
  - Value properties
  - Constraint properties
  - Ports
- 2 kinds of behavioral features
  - Operations
  - Receptions

Purpose and Use of Block Features – Part Properties

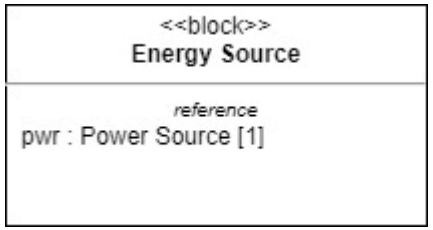
- Represents a structure that is internal to the block
- Represents ownership (a part property can belong to only one composite structure at a time)
- 'typed by' a "block"
- Notation = <part name> : <type> [<multiplicity>]



- Multiplicity: constraint on the number of instances that the part property can represent within the composite
  - Expressed as a single integer OR a range of integers
  - Unconstrained number of instances: [0..*] or [*]
  - If no multiplicity is shown for a part property, the default is 1; [1..1]

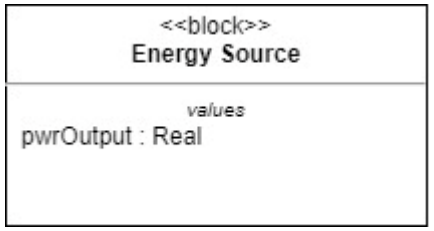
Purpose and Use of Block Features – Reference Properties

- Represents a structure that is external to the block
- Does NOT convey ownership (i.e. it's an external structure); represents a "needs" relationship
  - implies some type of connection must exist
- Notation = <reference name> : <type> [<multiplicity>]
  - Typed by a block or actor somewhere in the system model
- Multiplicity: constraint on the number of instances that the reference property can represent within the composite
  - Expressed as a single integer OR a range of integers
  - Unconstrained number of instances: [0..*] or [*]
  - If no multiplicity is shown for a reference property, the default is 1; [1..1]

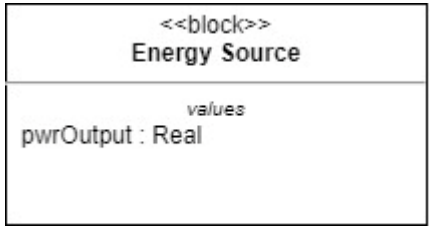


Purpose and Use of Block Features – Value Properties

- Represents a quantity, a Boolean, or a string (often- something you assign a number to)
- Notation = <value name> : <type> [<multiplicity>] = <default value>
  - Typed by a value type somewhere in the system model
  - <default value> is optional



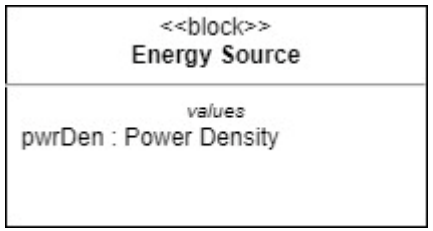
- Multiplicity: constraint on the number of instances that the value property can represent within the composite
  - Expressed as a single integer OR a range of integers
  - Unconstrained number of instances: [0..*] or [*]
  - If no multiplicity is shown for a value property, the default is 1; [1..1]
  - Value property referred to as a collection when it's multiplicity has an upper bounder greater than 1
- Can hold values that are both assigned or derived (calculated)
  - A forward slash (/) in front of the value name conveys value property is derived



COLLEGE of CHARLESTON
1770

Purpose and Use of Block Features – Constraint Properties

- Represents a mathematical relationship that is imposed on a set of value properties
- Notation = <constraint name> : <type>



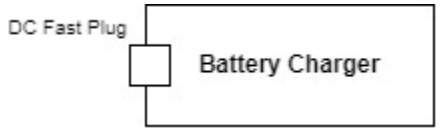
  - Typed by a constraint block somewhere in the system model
    - constraint blocks generally encapsulate mathematical equations
- Blocks can own constraint properties to constrain value properties

Purpose and Use of Block Features – Ports

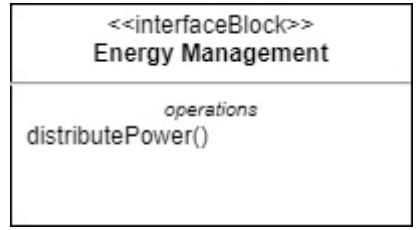
- Represents a distinct interaction point at the boundary of a structure through which external entities can interact with that structure – to:
  - Provide or request a service
  - Exchange matter, energy, or data
- Decouples a block's clients from any particular internal implementation
- Can represent any type of interaction point
  - Physical object (fuel nozzle, HDMI jack)
  - Software object (message queue, GUI)
  - Interaction between business organizations (purchase order, website)
- Two kinds of ports (v1.2 and earlier)
  - Standard port: specify the interaction with a focus on services that a block provides or requires
  - Flow port: specify an interaction with a focus on types of matter, energy, or data that can flow in and out of a block

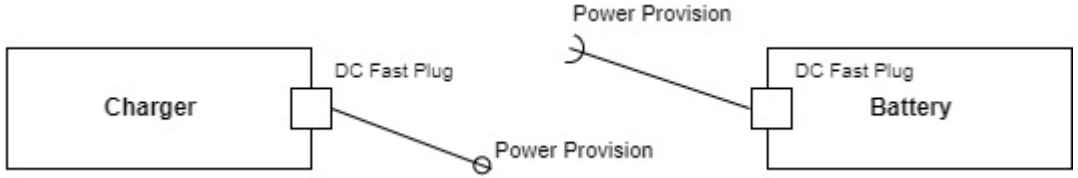
Purpose and Use of Block Features – Standard Ports and Interfaces

- Standard Ports
  - (most often) Displayed as a small square straddling the border of a block
  - Can have a modeler defined name
  - Can have one or more types
    - Types are the interfaces assigned



- Interfaces
  - Element of definition
  - Defines a set of operations and receptions that clients and providers will conform to
  - Assigned as either a provided interface or a required interface
    - Provided interface is displayed with the ball notation
    - A block that provides an interface must implement all of the interface's operations and receptions
    - Required interface is displayed using the socket notation
    - A block that requires an interface may invoke one or more of it's operations and receptions (but not necessarily all)

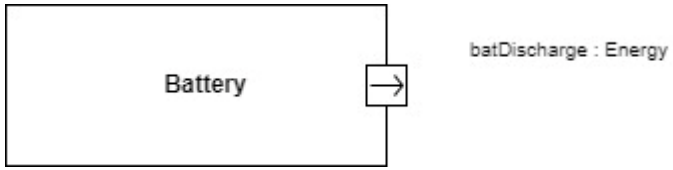




COLLEGE of CHARLESTON
1770

Purpose and Use of Block Features – Flow Ports

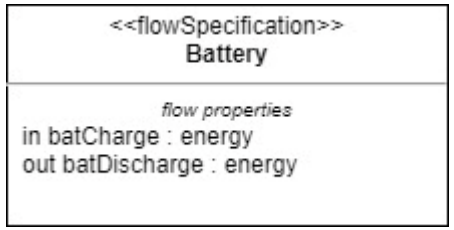
- (most often) Displayed as a small square straddling the border of a block and a symbol inside the small square



- Can have a modeler defined name and type; name and type shown as a string floating near the flow port
  - Format = <name> : <type>
- Two types of flow ports: nonatomic and atomic

Purpose and Use of Block Features – Nonatomic Flow Ports

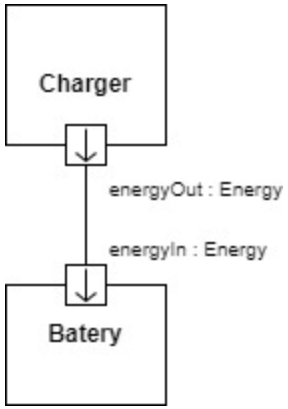
- Symbolized as <>: multiple types of items can flow in and out of that port
- Typed by a flow specification defined somewhere in the system model
- Flow specification- element of definition; defines a set of flow properties that can flow in or out of nonatomic flow port



- Flow property- specific item that can flow in or out of a block via a flow port (has a direction, name, and type)
  - Notation = <direction> <name> : <type>
    - Direction can be in, out, inout
    - Name is modeler defined
    - Type is a value type, block, signal created somewhere in the system model
- Conjugated (~): directions of the flow properties are reversed

Purpose and Use of Block Features – Atomic Flow Ports

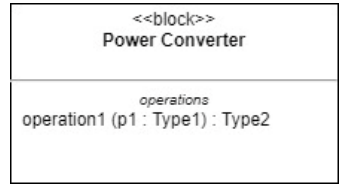
- Symbolized as an arrow conveying direction of flow
- Single type of item can flow in or out via that port



-
  - Type is a value type, block, or signal created somewhere in the system model

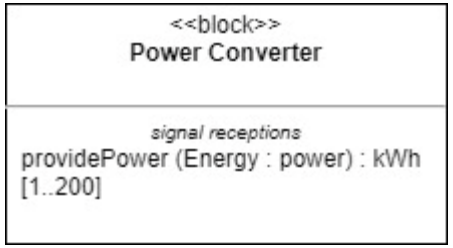
Purpose and Use of Block Features – Operations

- Behavior that a block performs when a client calls it; invoked by a call event
  - Most often synchronous, but doesn't have to be
- Displayed in the operations compartment of a block
  - Notation = <operation name> (<parameter list>) : <return type> [<multiplicity>]
    - Operation name is modeler defined
    - Parameter list: comma separated list of parameters
      - Parameters: inputs or outputs of the operation
      - Parameter notation: <direction> <parameter name> : <type> [<multiplicity>] = <default value>
        - Direction: in, out, inout
        - Parameter name is modeler defined
        - Type: value type or block created somewhere in the system model
        - Multiplicity: constraint on the number of instances of the type that the parameter can represent
        - Default value: value assigned to the parameter if no value is specified as an argument when the operation is called
    - Return type: value type or block created somewhere in the system model
    - Multiplicity: # of instances of the return type the operation can return to the caller when it completes
- Good practice: use a verb phrase to name an operation (it's a behavior!)



Purpose and Use of Block Features – Receptions

- Behavior that a block performs when a client sends a signal that triggers it; invoked by a signal event
  - Always represents an asynchronous behavior
- Signal: can represent any type of matter, energy, data that one part of a system sends to another part
  - Signals can own properties (ex. Data carried from client to target)
  - Signal properties become inputs to the reception
  - The client can add values for properties to the instance of a signal
  - The reception must have a parameter with a compatible type for each property of the signal
- Reception notation: <<signal>> <reception name> (<parameter list>)
  - Reception name must match the name of the signal that triggers it
  - Parameter notation: <parameter name> : <type> [<multiplicity>] = <default value>
    - Parameter name is modeler defined
    - Type: value type or block that exists somewhere in the system model
    - Multiplicity: constraint on the number of instances of the type that the parameter can represent
    - Default value: value assigned to the parameter if no value is provided in the corresponding property of the signal
- Receptions cannot have return types (b/c they are asynchronous)
  - Parameters can only be inputs and never outputs



COLLEGE of CHARLESTON

Purpose and Use of Association Relationships

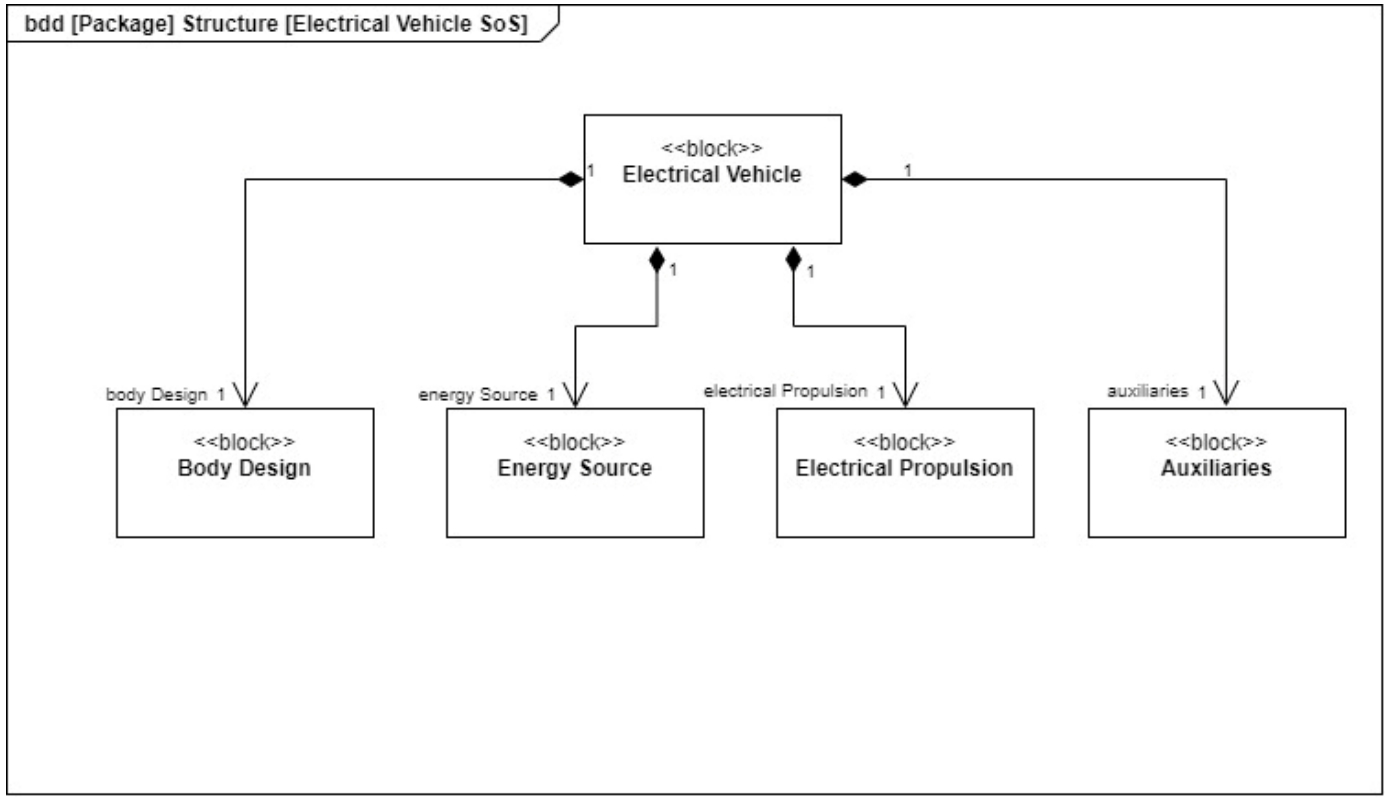
- Two types of association relationships: reference and composite
  - Reference association corresponds to a reference property
  - Composite association corresponds to a part property
- Associations are an alternative notation to convey these kinds of structural relationships within a system

Purpose and Use of Association Relationships – Reference Associations

- A reference association between two blocks means that a connection can exist between instances of those blocks in an operational system
  - These instances can access each other for some purpose across the connection
- Notation: solid line between two blocks
  - Open arrowhead on one end conveys unidirectional access
  - Absence of arrowheads on either end conveys bidirectional access
- Association labels:
  - Optionally display association name floating near the middle of the line
    - Association name is modeler defined
  - Optionally display a role name and multiplicity on either end of the line
    - Role name corresponds to a reference property owned by the block at the opposite end
    - Typed by the block that it's next to
    - Multiplicity show on the end of a reference association (near role name) corresponds to the multiplicity of that same reference property

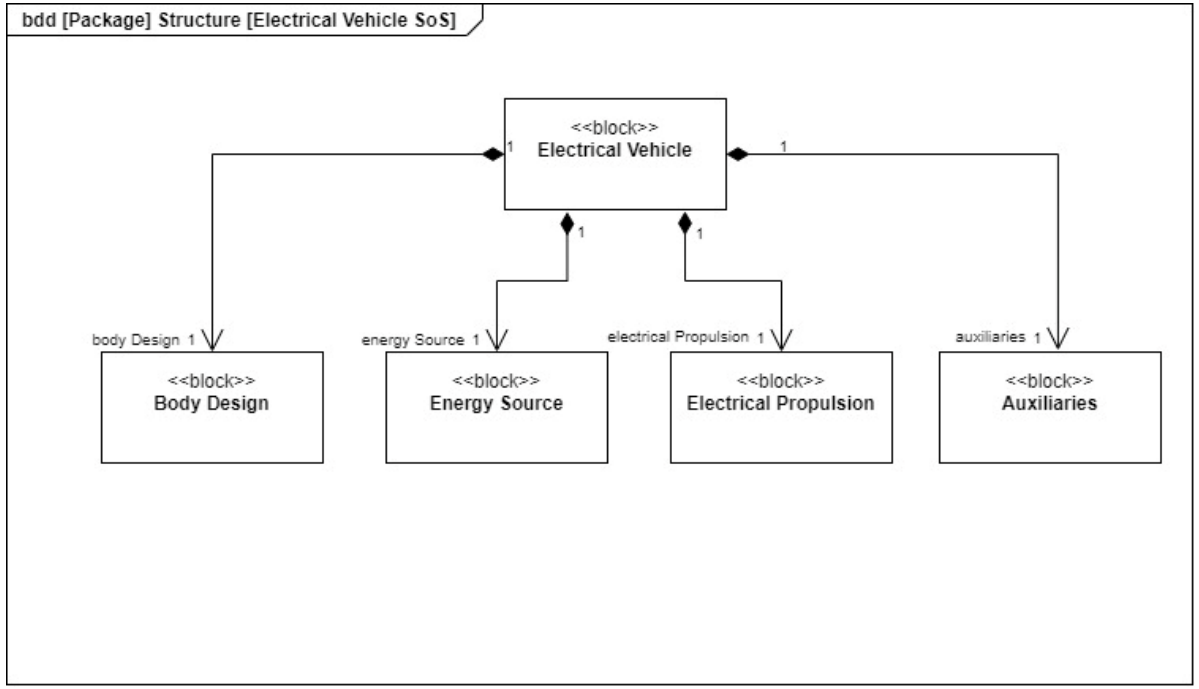
Purpose and Use of Association Relationships – Composite Relationships

- Conveys structural decomposition
- An instance of a block at the composite end is made up of some number of instances of the block at the part end
- Notation: solid line between two blocks with a solid diamond on the composite end
  - Open arrowhead at the part end conveys unidirectional access from the composite to its part
  - Absence of an arrowhead conveys bidirectional access



Purpose and Use of Association Relationships – Composite Relationships Cont.

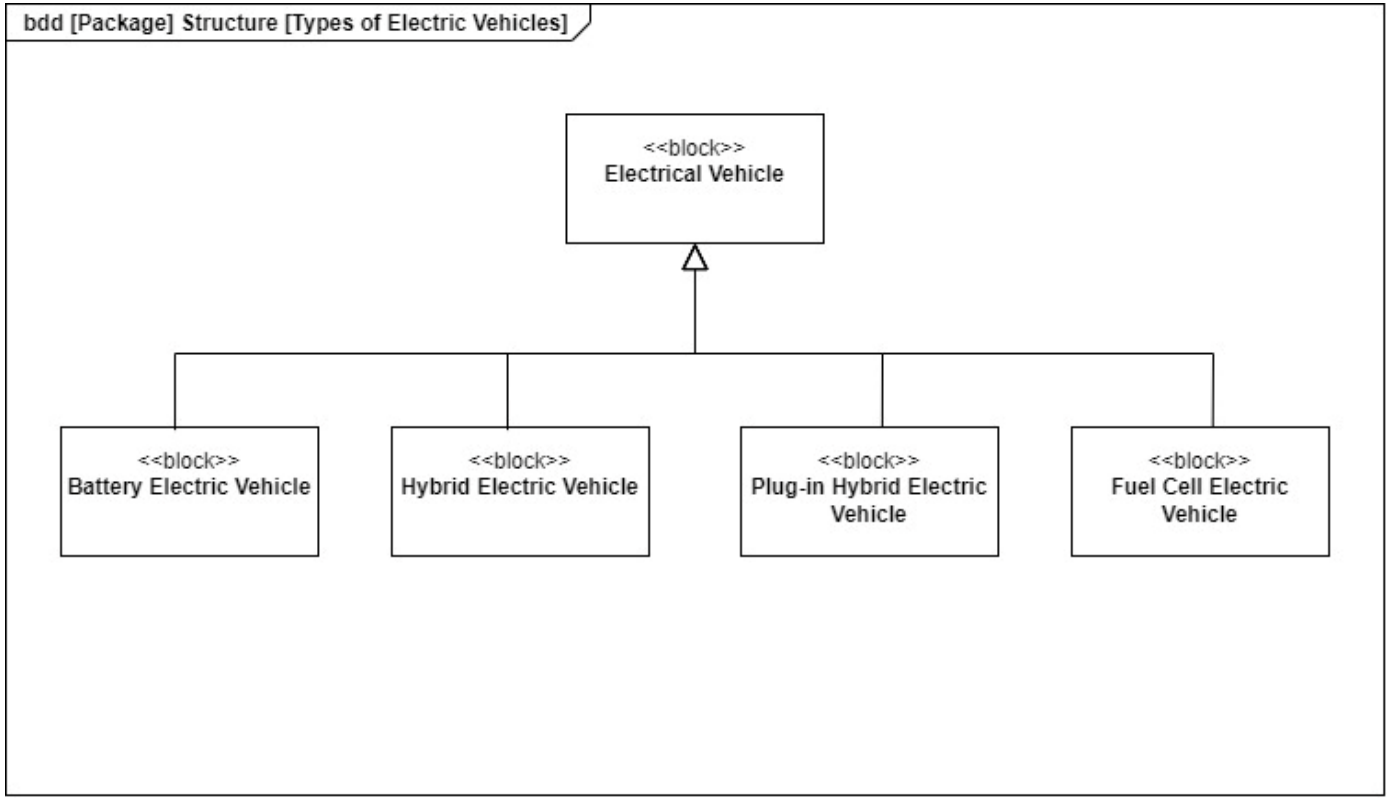
- Role name shown on the part end corresponds to the name of a part property- that's owned by the block at the composite end and typed by the block at the part end
- Multiplicity on the part end is not restricted- a composite structure can be made up of an arbitrary number of instances of parts
- Multiplicity on the composite end is restricted- a part can only belong to one composite at a time (upper bound on the composite end must always be 1)
  - Lower bound of 0 means a part can be removed from it's composite
  - Lower bound of 1 means a part cannot be removed from it's composite
  - DEFAULT multiplicity on a composite end of a composite association is [0..1]



COLLEGE of CHARLESTON

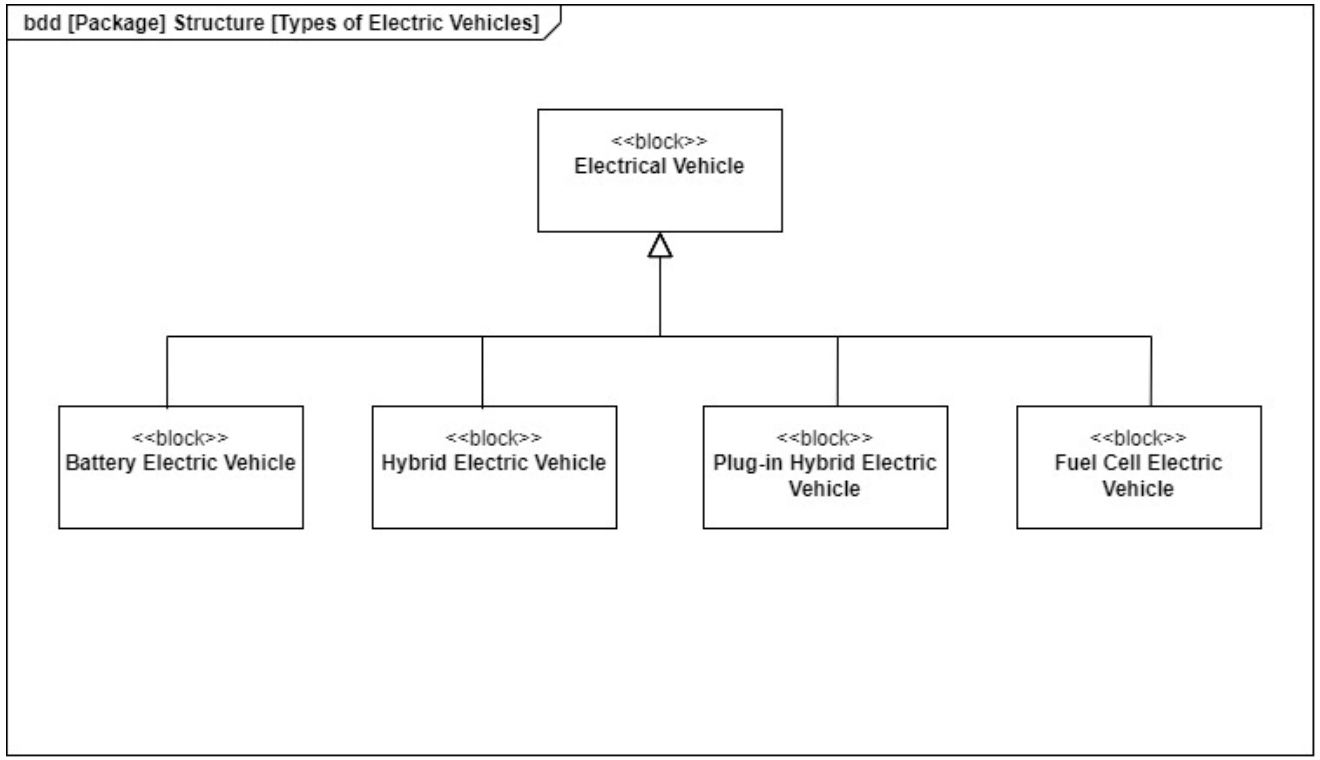
Purpose and Use of Generalization Relationships

- Generalization relationship: conveys inheritance between two elements; used to create classification trees
  - Supertype: generalized element
  - Subtype: specialized element
- Notation: solid line with a hollow, triangular arrow head on the end of the supertype
  - Read as “subtype is a type of supertype”



Purpose and Use of Generalization Relationships Cont.

- Generalizations are transitive – type hierarchies can be arbitrarily deep
- Subtypes inherit ALL the features of the supertype (structural and behavioral)
- Subtypes are a specialization of it's supertype – can have other features its supertype does not have
- Generalizations can be used to define abstractions – factors out features that are common among the subtypes
  - Define a common feature in one place within the model (in the supertype) and that common feature propagates down to the type hierarchy to all subtypes
  - Abstractions convey substitutability – a subtype will be accepted wherever it's supertype is required



Purpose and Use of Dependency Relationships

- Dependency relationship: a client element relies on a supplier element; when the supplier element changes, the client element may also have to change
- Dependencies are a structure of the model, not of the system the model represents
- Notation (on a BDD): dashed line with an open arrowhead drawn from client to supplier

Purpose and Use of Actors

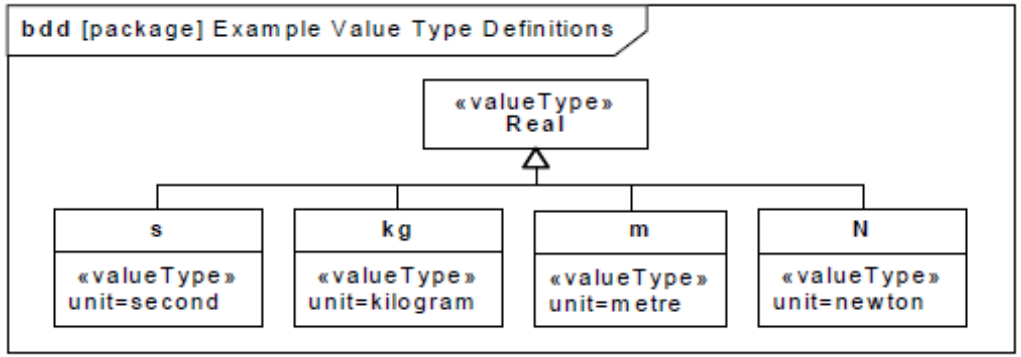
- Actor: represents someone or something that has an external interface with your system
- Name of an actor conveys a role played by a person, organization, or other system when it interacts with your system
- Notations:
  - Stick figure
  - Rectangle with keyword <<actor>> preceding the name
- Key ideas about generalizations, reference associations, and composite associations apply when actors are involved in these relationships – EXCEPT:
  - A generalization cannot be defined between an actor and a block
  - An actor cannot have parts (cannot appear at the composite end of a composite association – actors are regarded as a black box)

Purpose and Use of Value Types

- Value Type: element of definition which defines a type of quantity (except Boolean and string)
- Appear as types for the following:
  - Value Property (most common)
  - Atomic flow ports on blocks and actors
  - Flow properties in flow specifications
  - Constraint parameters in constraint blocks
  - Item flows and item properties on connectors
  - Return types of operations
  - Parameters of operations and receptions
  - Object nodes, pins, and activity parameters within activities

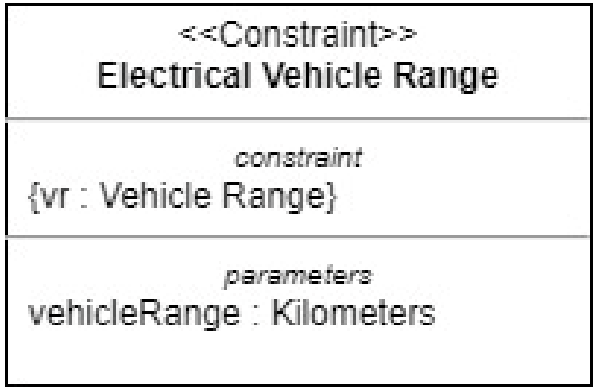
Purpose and Use of Value Types Cont.

- 3 types: primitive, structured, and enumerated
  - Primitive: no internal structure
    - Notation: rectangle with stereotype <<valueType>> preceding the name
    - 4 types: String, Boolean, Integer, Real (can define specializations of these supertypes)
  - Structured: has an internal structure (two or more value properties)
    - Notation: rectangle with stereotype <<valueType>> preceding the name
    - 1 type: complex (two parts – realPart and imaginaryPart, both of type Real
  - Enumerated: defines a set of literals (legal values)
- Value Types can be related to one another using generalization
  - Generalizations are transitive
  - Substitutability principle applies



Purpose and Use of Constraint Blocks

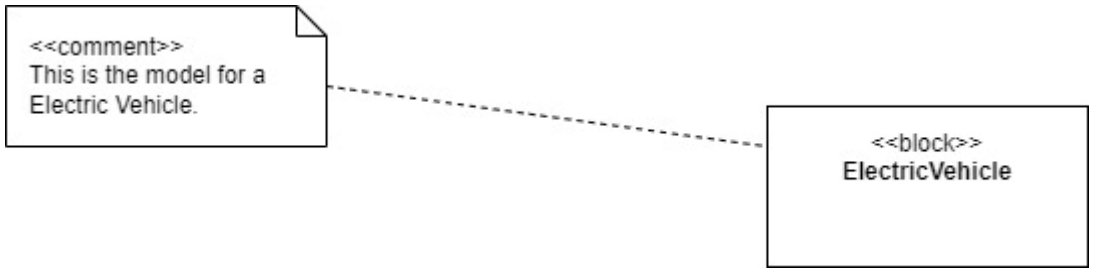
- Special kind of block (element of definition)- used to encapsulate a reusable constraint expression
  - Constraint expression: usually an equation or an inequality
    - Variables are called constraint parameters (often represent quantities and are typed by value types)
    - Constraint parameters receive their values from the value properties they're bound to (displayed on a parametric diagram)
- Notation: rectangle with the stereotype <<constraint>> preceding the name
  - Constraint expression appears between curly brackets { } in the constraint compartment
  - Constraint parameters are listed in the parameters compartment
- Can create complex constraint blocks from a set of simpler constraint blocks
  - Complex constraint block displays constituent parts as a list of constraints properties in the constraints compartment
  - Can also use composite associations to convey that one constraint block is composed of other, simpler ones



COLLEGE of CHARLESTON
1770

Purpose and Use of Comments

- Comment: model element used to express information on a diagram in an unconstrained way as a block of text
- Consists of a single attribute: a string of text called the body
- You can attach a comment to other elements on a diagram (and it can be attached to multiple elements)
- Notation: note symbol, rectangle whose upper right corner is bent
  - A dashed line is used to attach a comment to other elements
- Specialized kinds of comments: rationale, problem, diagram description
  - Appear as a note symbol with respective stereotype preceding the body of the comment (ex. <<rationale>>)



Default Multiplicities

- Association BDDs
  - SysML defines defaults for multiplicities on the ends of specific types of associations. A part or shared association has a default multiplicity of [0..1] on the black or white diamond end. A unidirectional association has a default multiplicity of 1 on its target end. These multiplicities may be assumed if not shown on a diagram. To avoid confusion, any multiplicity other than the default should always be shown on a diagram.*

*Per OMG Standards

Questions

COLLEGE of CHARLESTON
1770

References

- Additional information can be obtained by reviewing:
  - SysML Distilled (Delligatti)
    - Chapter 3: Block Definition Diagrams

Summary