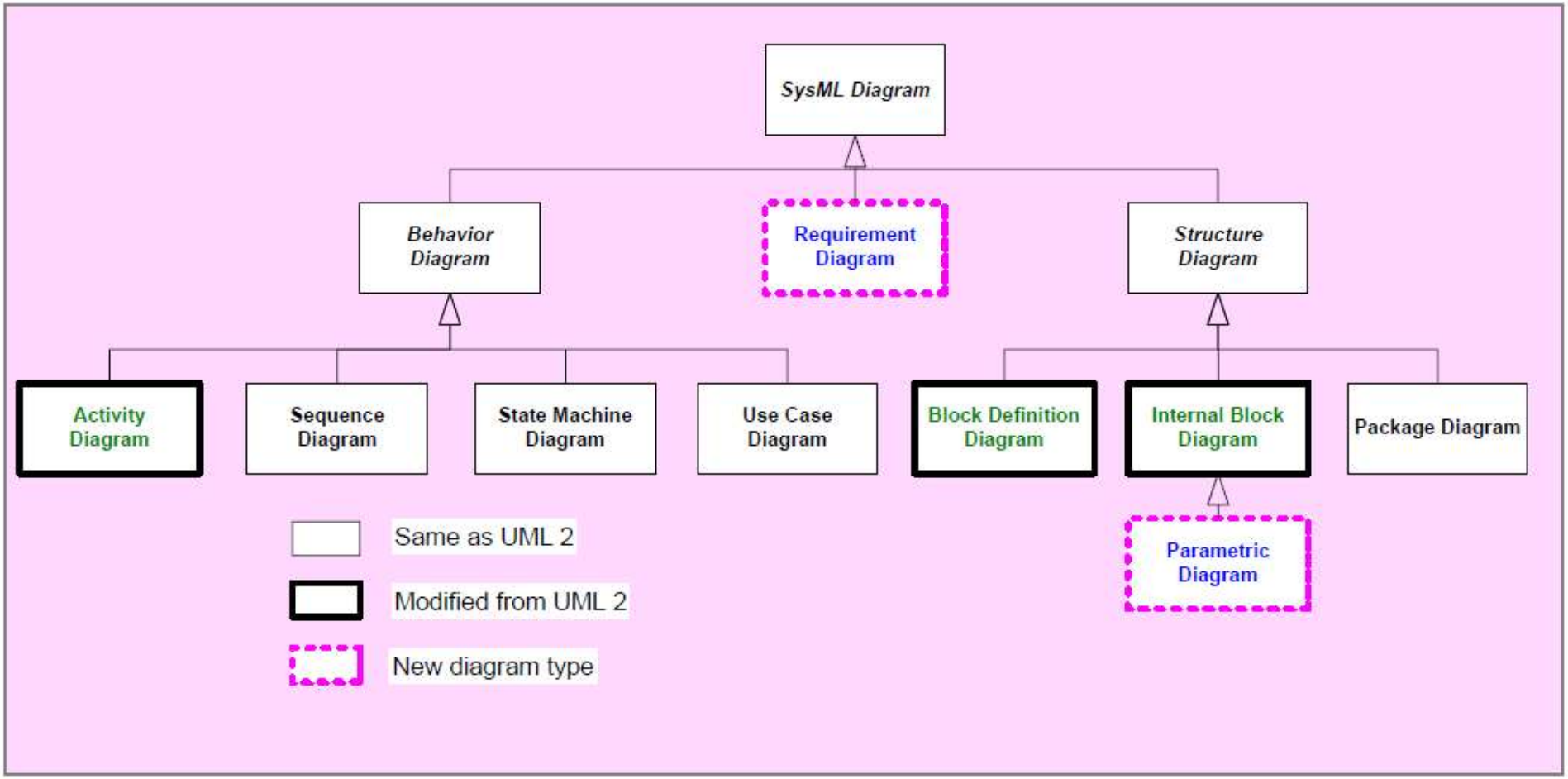
Systems Engineering

mariaiulianadascalu@gmail.com

SysML Diagram Taxonomy



Content

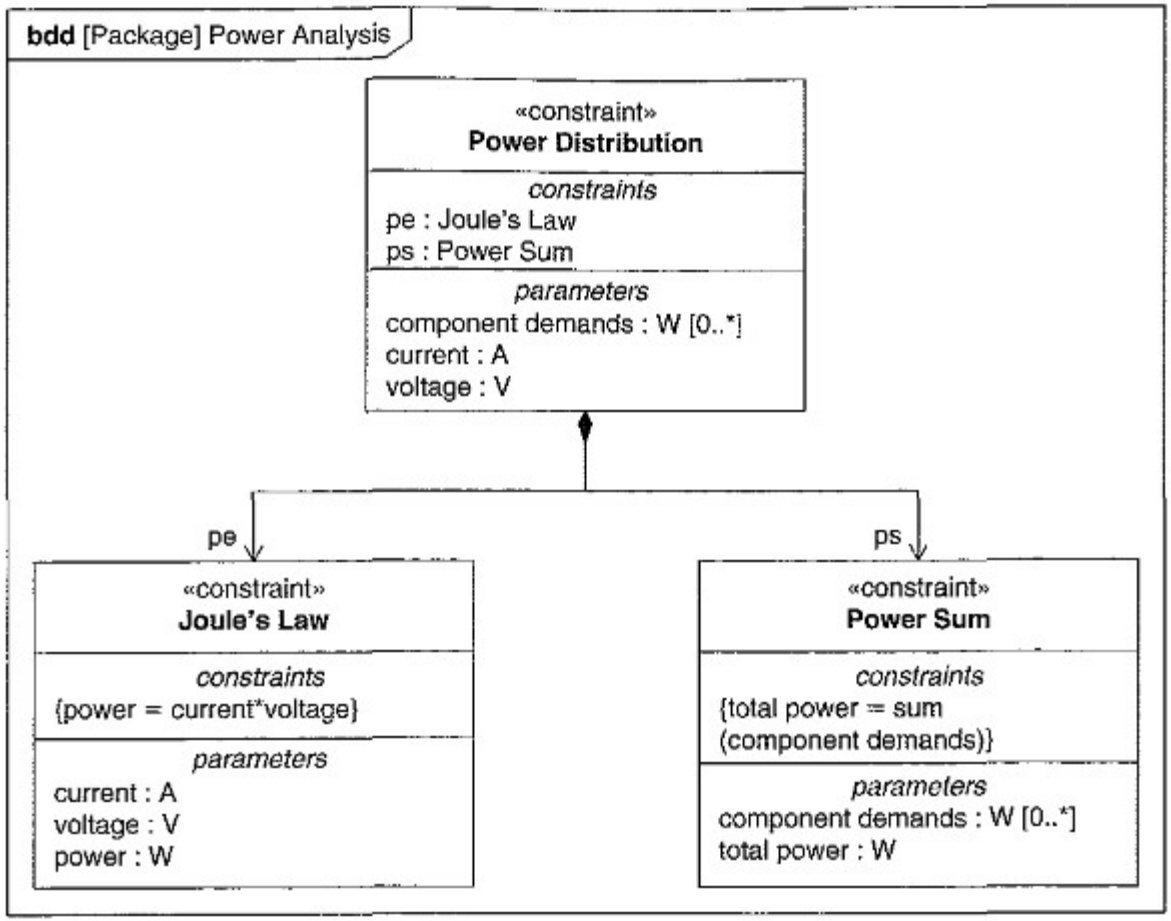
- Parametric models
- Modeling behavior

Parametric Models

- Used do capture constraints on the performance and physical properties of systems
- The constraints are expressed as equations whose parameters are bound to the properties of systems
- Used to make analysis (trade-off studies, design optimization,…)
- Supported by SysML through constraint blocks:
  - a special kind of block used to define equations so that they can be reused and interconnected
  - have 2 main features:
    - a set of parameters
    - an expression that constrains the parameters

Defining constraints with…

- Bdd

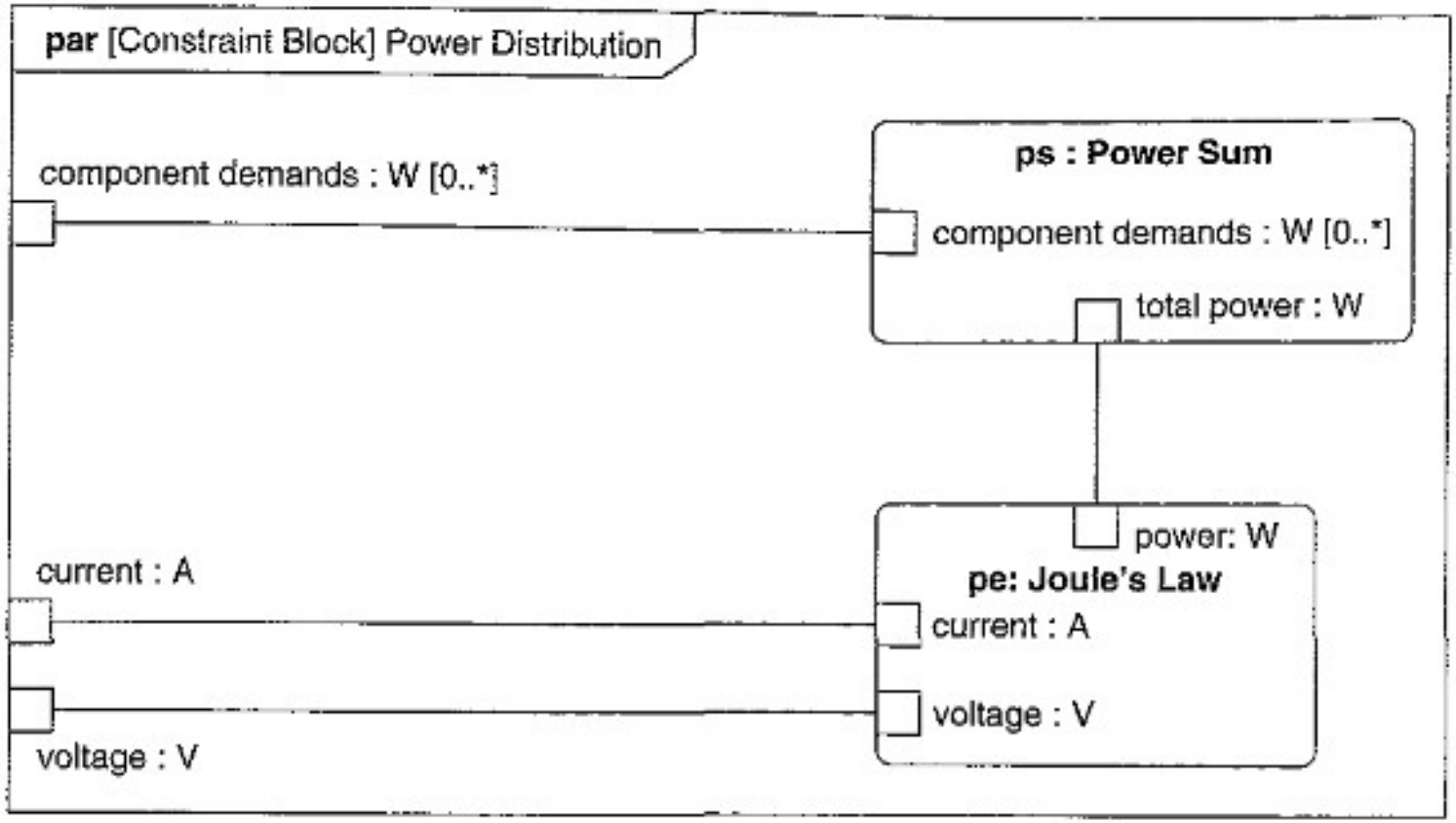


- Parametric diagram

Parametric diagram

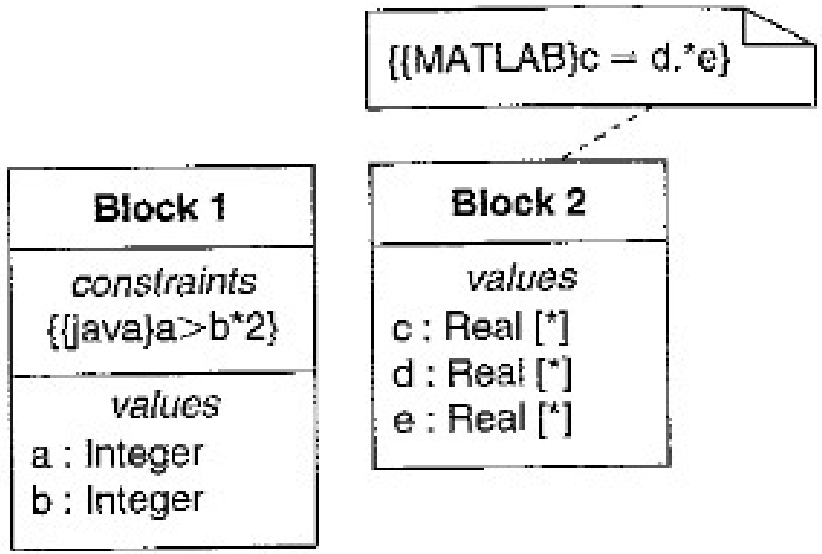
- Creates systems of equations that can constrain the properties of a block
- Header:

par [model element type] model element name [diagram name]



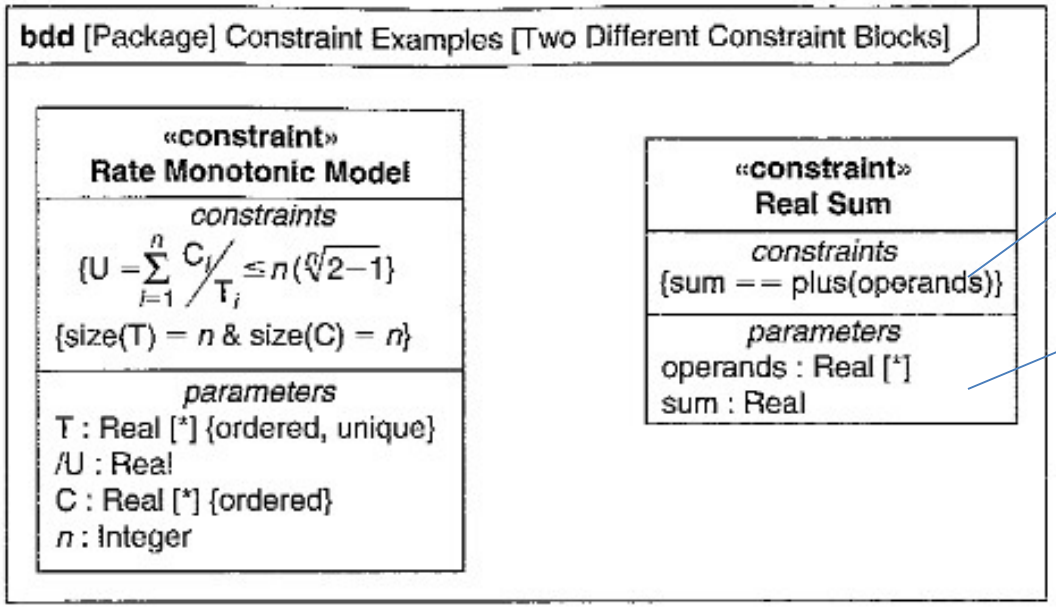
Using Constraint Expressions to Represent System Constraints

- SysML includes a generic mechanism for expressing constraints on a system, not a build-in constraint language to evaluate the constraints
- The definition of a constraint can include the language used to enable the constraint to be evaluated
- Constraints may be owned by any element in a namespace



Encapsulating Constraints in Constraint Blocks

- Why?
  - To enable reuse

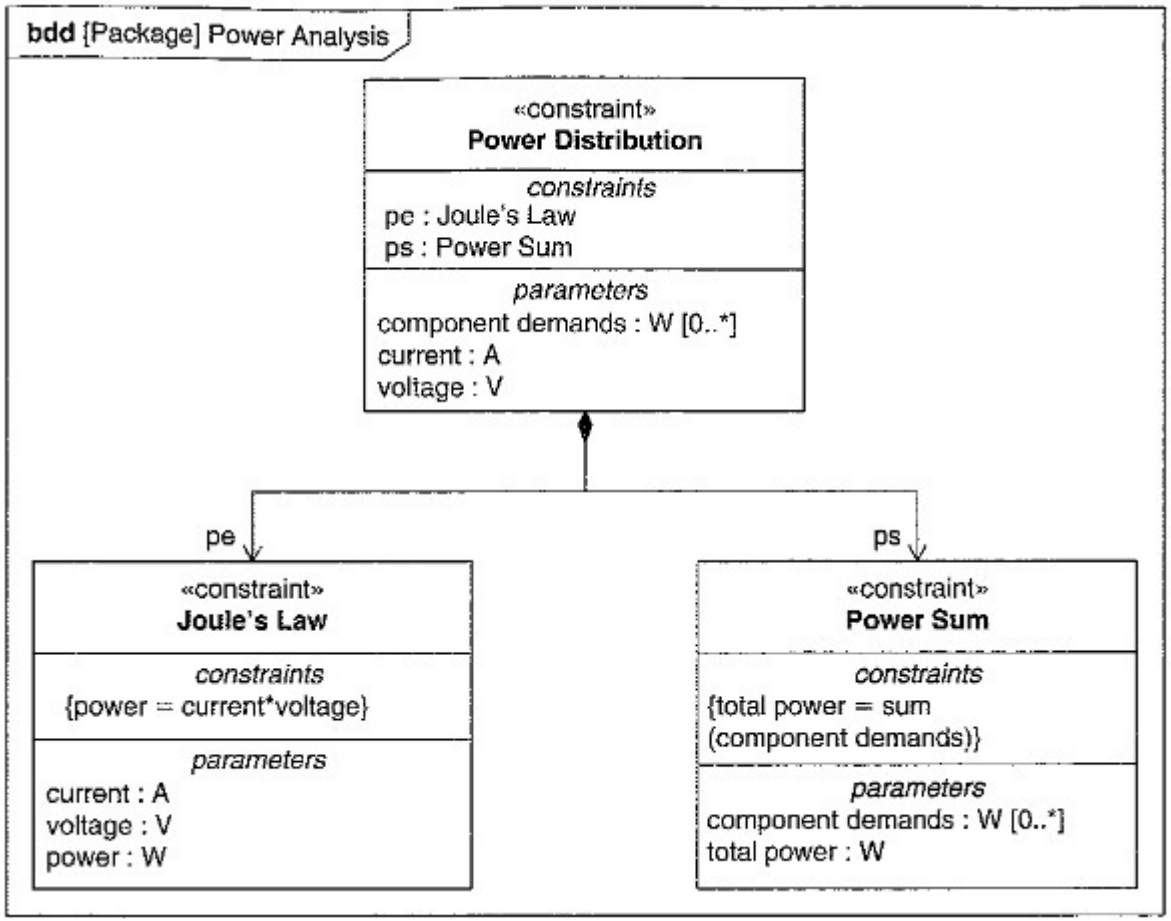


Constraint expression

Constraint parameters

Using Composition to Build Complex Constraint Blocks

- Modelers can compose complex constraint blocks from existing constraint blocks on a bdd



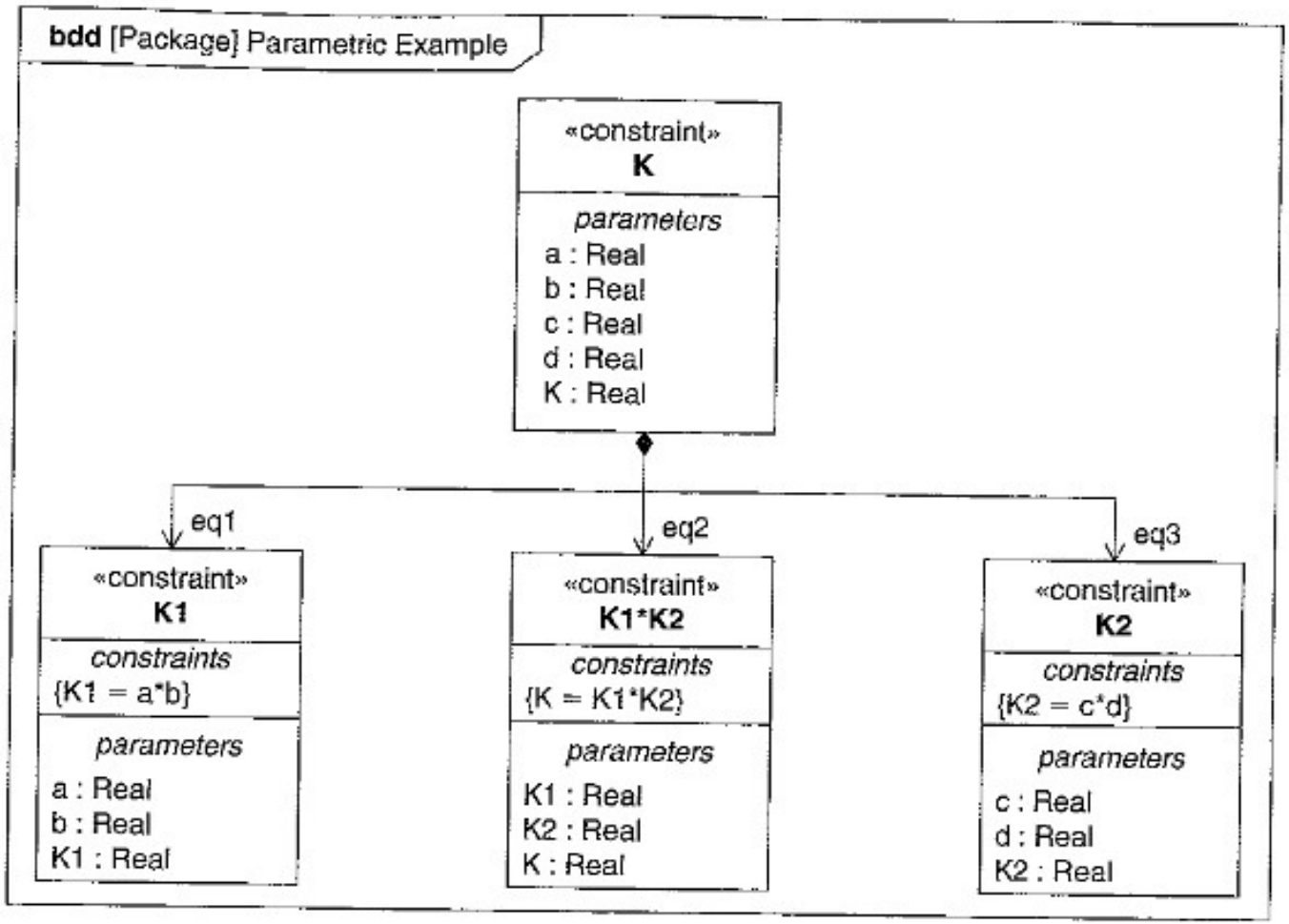
Using a Parametric Diagram to Bind Parameters of Constraint Blocks

- A bdd is used to define constraint blocks

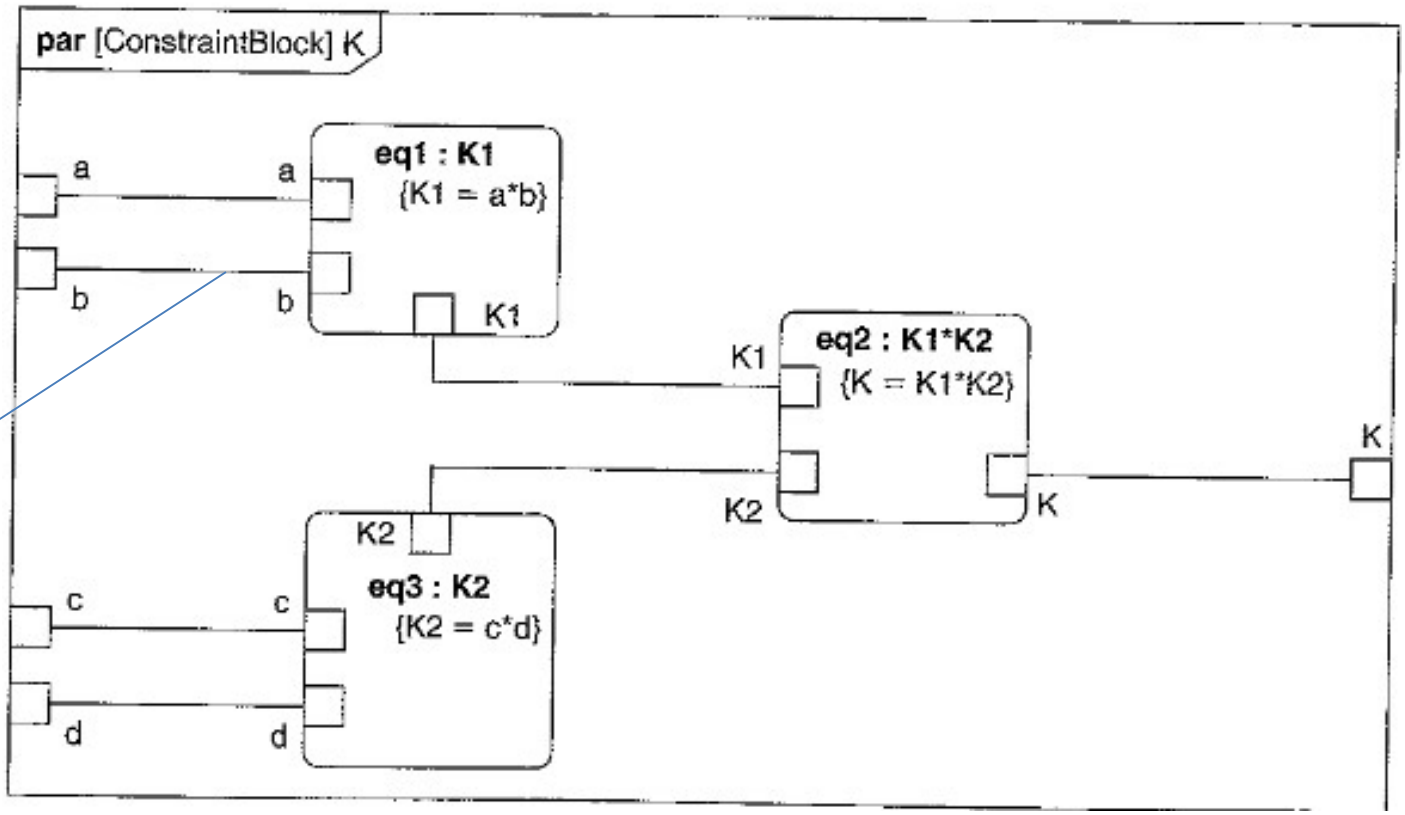
but

- A **parametric diagram** represents the usage of constrain blocks in a particular context (similar to the usage of blocks as parts in an ibd)

Example par (1)

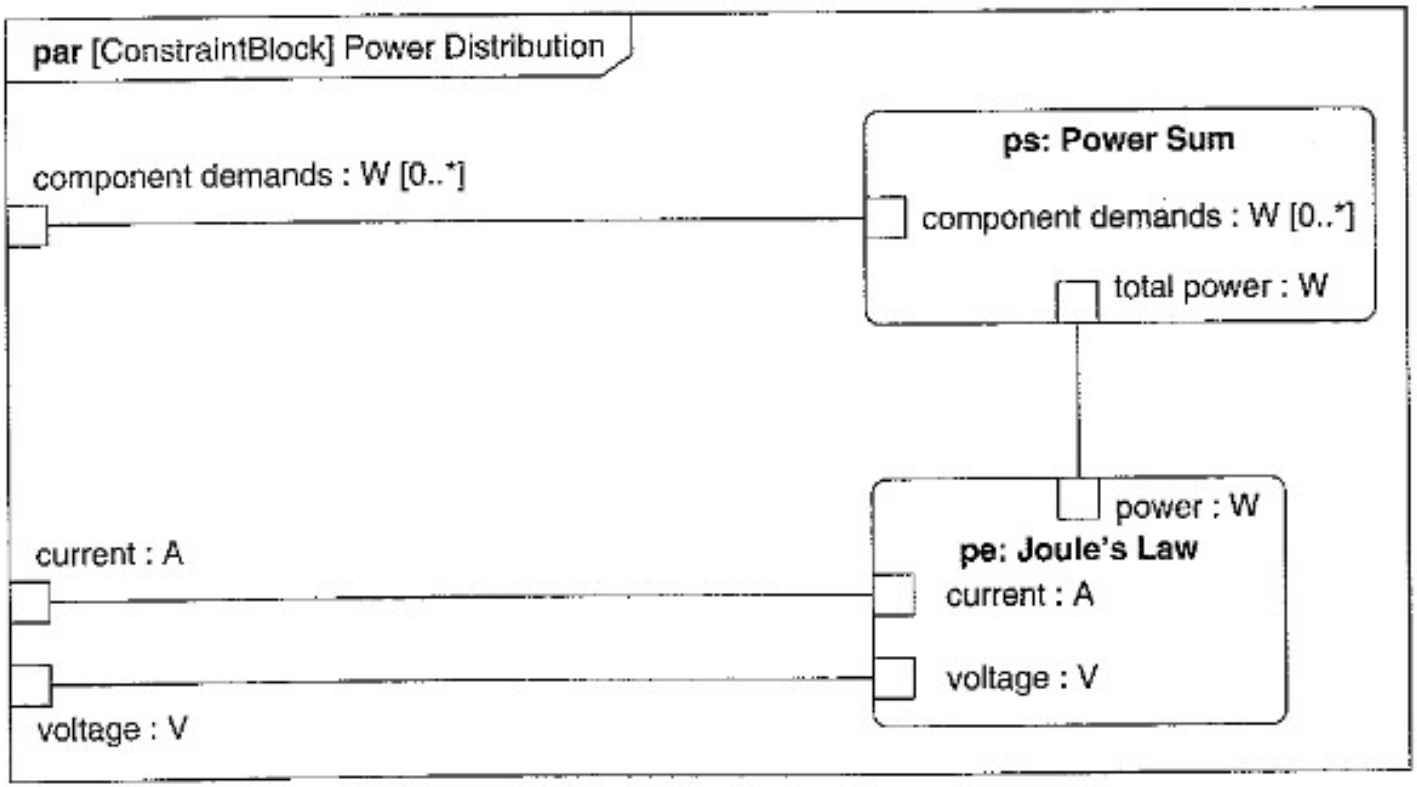


Example par (2)



Binding connector

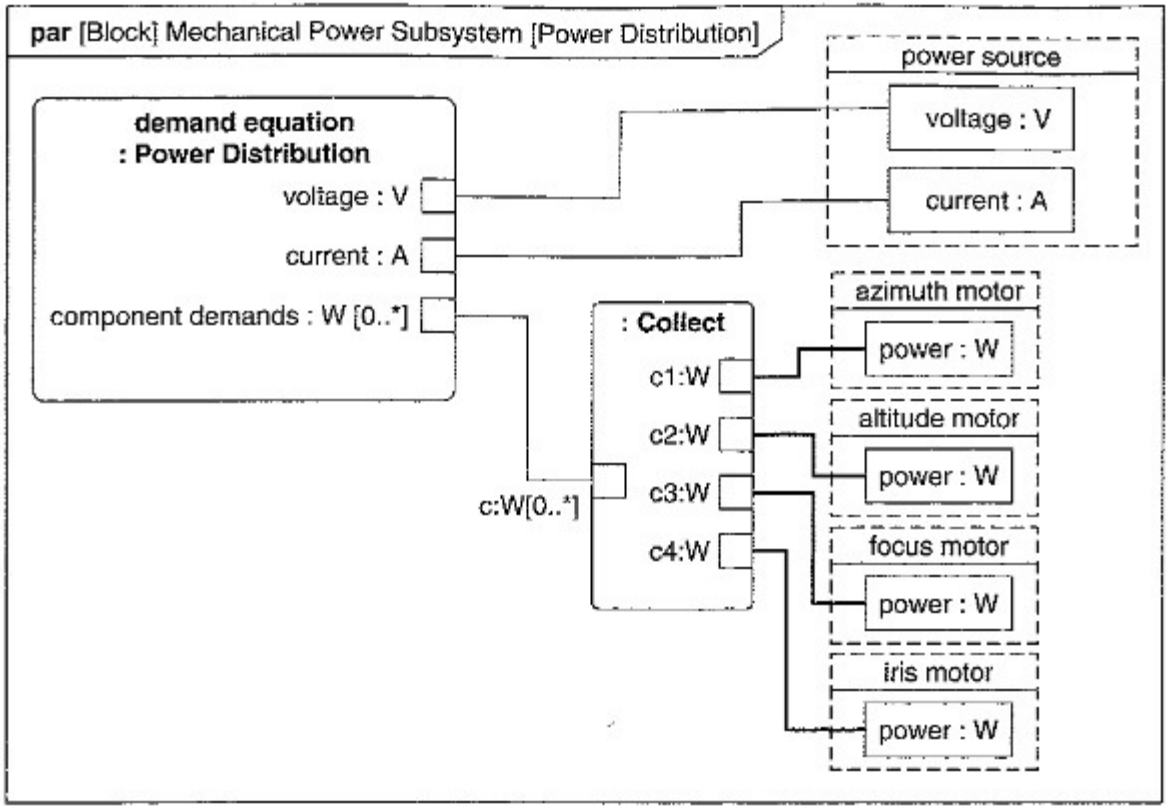
Example par (3)



Constraining Value Properties of a Block

- In a bdd, draw composite associations between the block whose values are being constrained and the required constraint blocks

- In a par, the block represents the enclosing frame and the constraint properties represent usages of the constraint blocks



Summary so far

- Constraint blocks can be defined in model libraries to facilitate specific types of analysis (performance, weight, thermal etc)
- Constraint blocks can be used by blocks to constrain the values of their properties
- Constraint properties are usages of constraint blocks
- Constraint properties bind to one another and to the value properties of blocks through parameters in parametric diagrams, using binding connectors
- An analysis context is a block that provides the context for a system or component that is subject to analysis:
  - Constraint blocks
  - References to the system being analyzed (the frame of the par)
- The analysis context can be passed to an engineering analysis tool to perform computational analysis
- Form of analysis: trade studies (compare alternatives)

References

- http://astah.net/tutorials/sysml/parametric
- http://astah.net/resources/documents/SySML-Tutorial.pdf)

Modeling Behavior in SysML

- with activities (flow-based behaviour)
- with interactions (message-based behaviour)
- with state machines (event-based behaviour)
- with use cases

SysML Activity Diagram

- an activity is a formalism for describing behaviour that specifies the transformation of inputs to outputs through a controlled sequence of actions
- an activity diagram is a representation for modelling flow-based behaviour

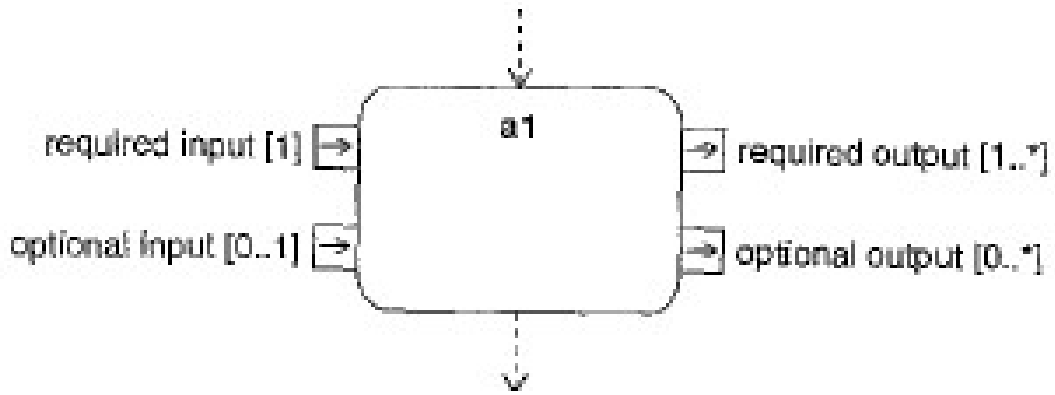
- header:

act [Activity] activity name [diagram name]

Actions-The Foundation of Activities

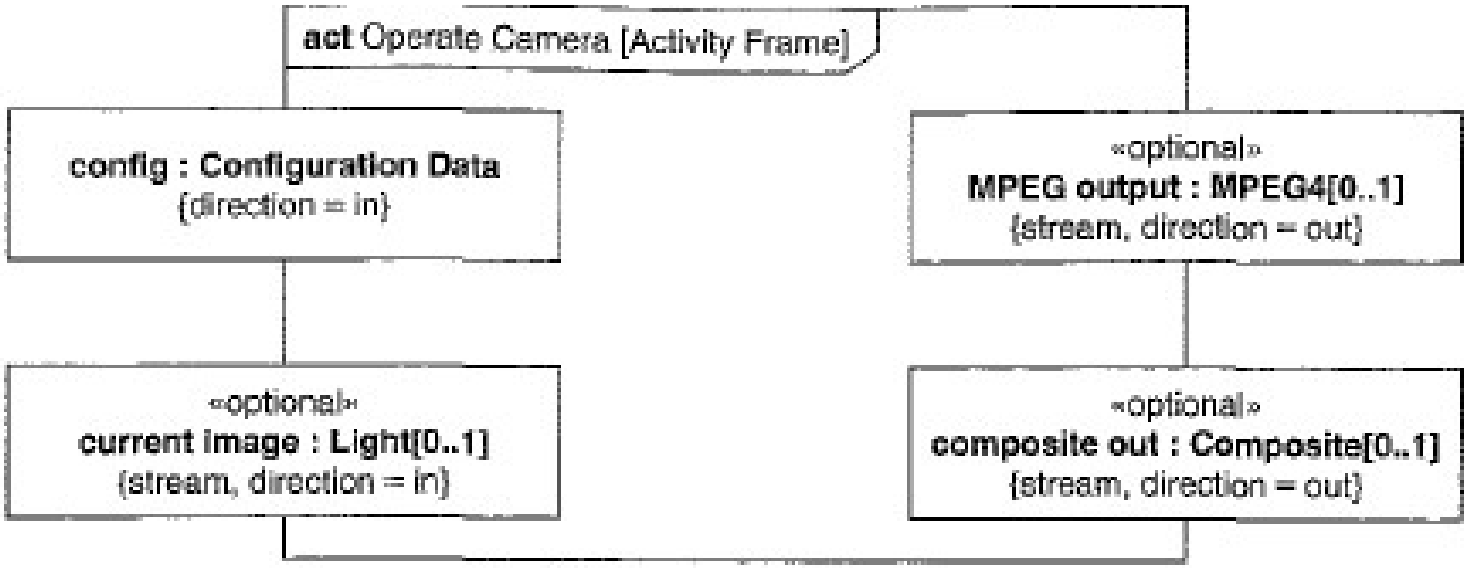
- an activity decomposes into a set of actions that describe how the activity executes and transforms its inputs to outputs

- an action processes tokens placed on its pins. Tokens on input pins are consumed, processed by the action, and placed on output pins for other actions to accept.



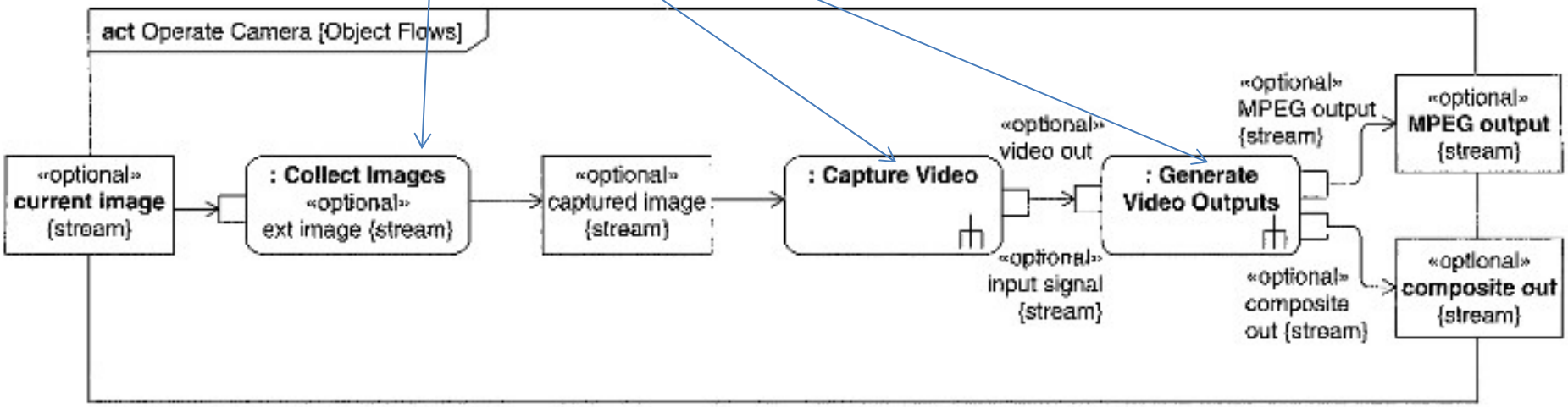
Parameters

- are inputs/outputs of activities
- each parameter may have:
  - a type
  - a direction
  - a multiplicity
  - a stream or not



Call Behavior Actions

- Invokes a behavior when it executes
- The called behavior is an activity/other types of SysML behavior
- The call behavior action owns a set of pins that mush match, in number and type, the parameters of the invoked behavior



Objects Flows

- Used to route input/output tokens that may represent information and/or physical items between object nodes (=parameter nodes and pins)

- Shown as a line connecting the source of the flow to the destination of the low, with an arrowhead at the destination

- Spot the object flows in the previous example

Routing Object Flows

- A **fork node** has one input flow and more than one output flow-it replicates every input token it receives onto each of its output flows.

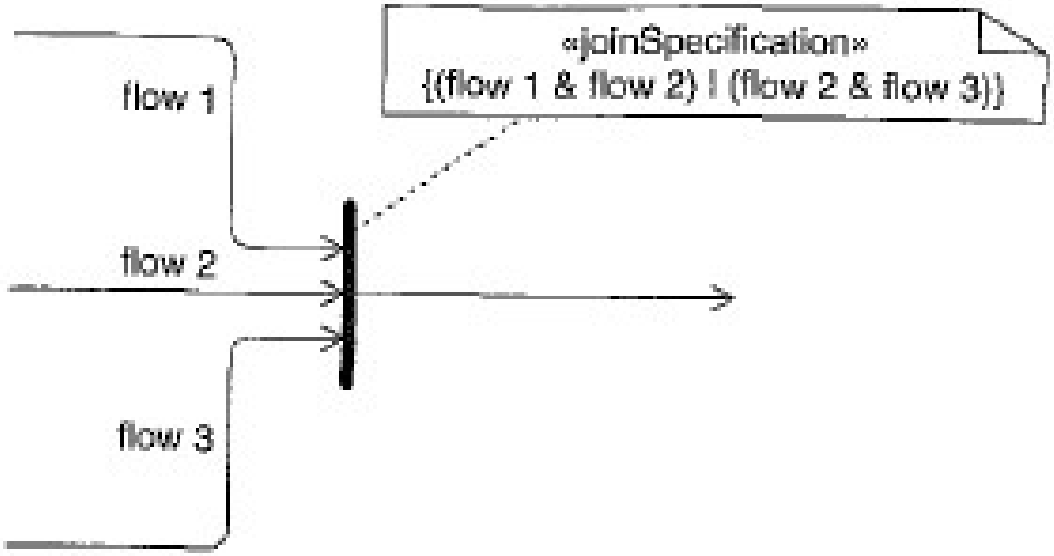
- A **join node** has one output flow and more than one input flow-its default behaviour for object flows is to produce output tokens only when an input token is available on each input flow.

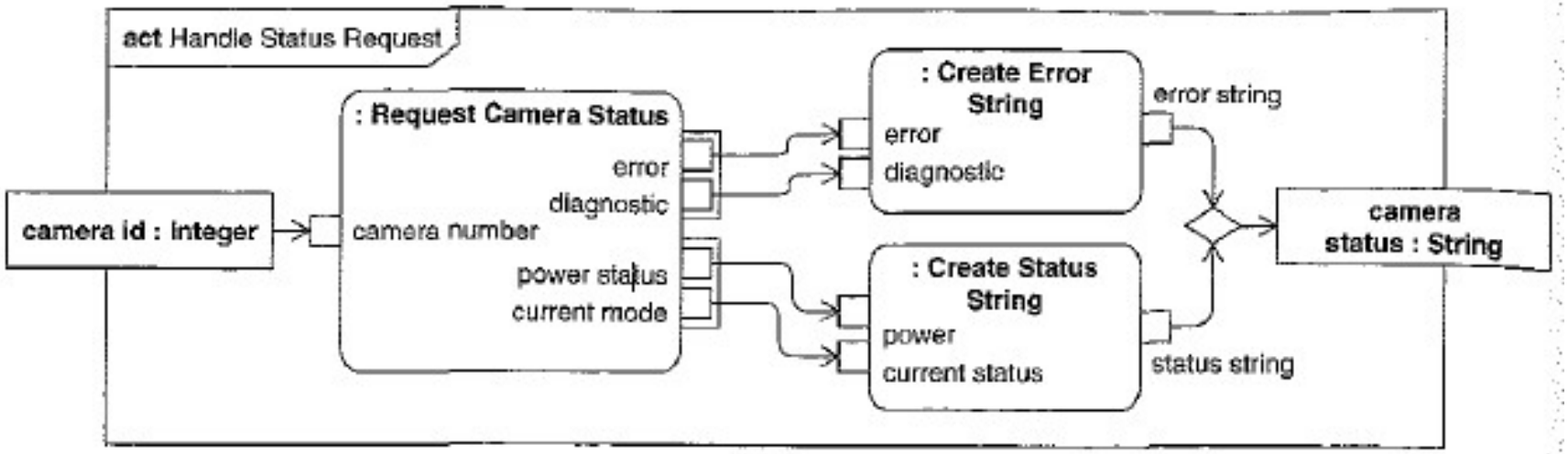
- A **decision node** has one input and more than one output flow-an input token can only traverse one output flow.

- A **merge node** has one output and more than one input flow-it routes each input token received on any input flow to its output flow. Unlike a join node, a merge node does not require tokens on all its input flows before offering them on its output flow: it offers tokens on its output flow as soon as it receives them.

- Fork and join symbols are shown as solid bars, typically aligned either horizontally or vertically. Decision and merge symbols are shown as diamonds.

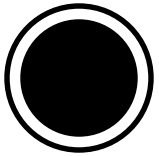
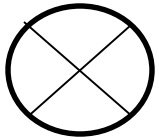
What type of nodes do we have here?



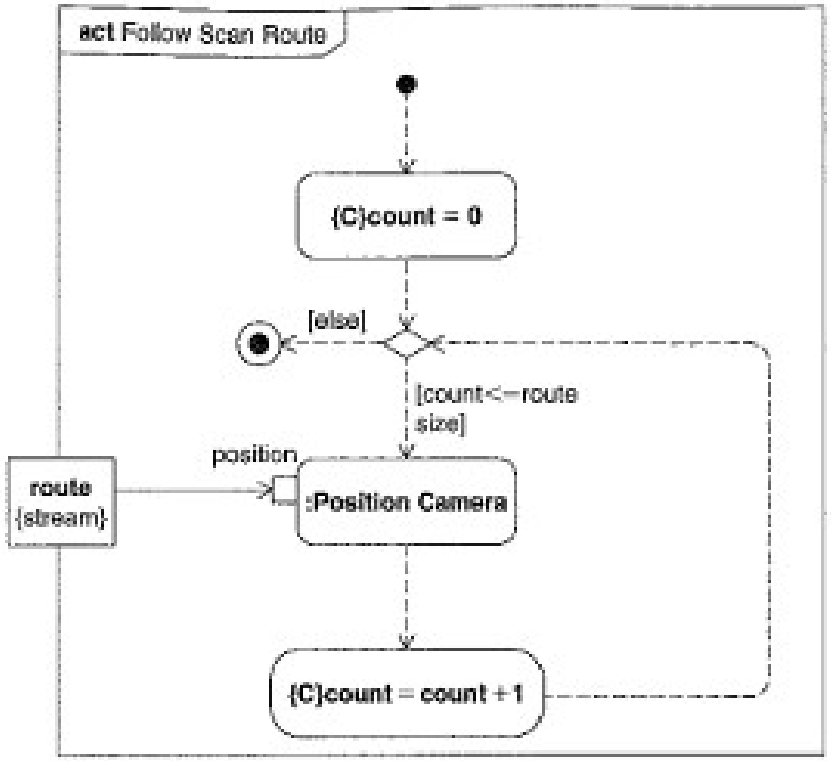


Control Flows

- Can be represented by using a solid (or dashed line) line with an arrowhead at the destination

- All the constructs used to route object flows can also be used to route control flows to represent control logic (e.g. join, merge,…) +
  - Initial node 
  - Activity final node 
  - Flow final node 

Example


act Follow Scan Route
{C}count = 0
[else]
[count<=route
size]
position
route
{stream}
:Position Camera
{C}count = count +1

Control Operators

- Special behaviors which produce control values via an output parameter – **ControlValue**, which enables/disables an activity

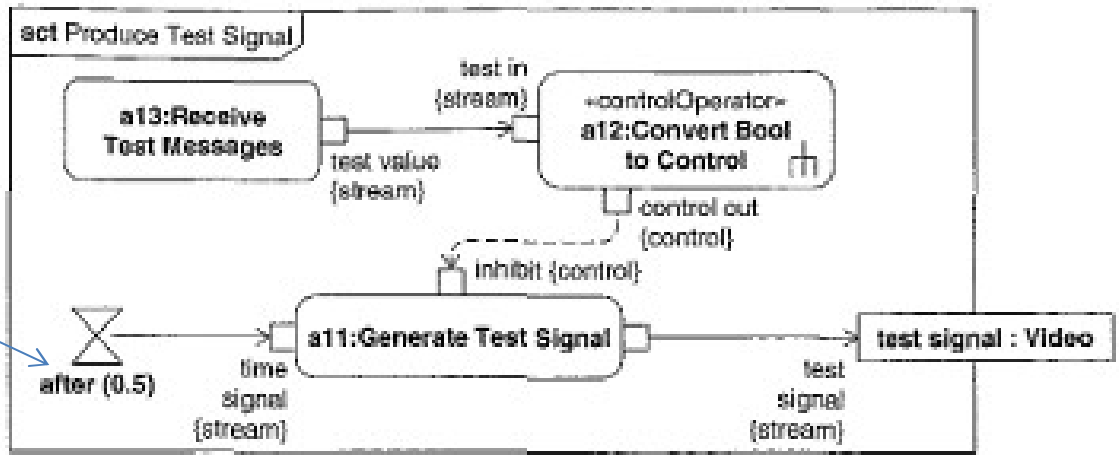
Signals

- Can be accepted by an activity, using an accept event action
- Can be sent, using a send signal action

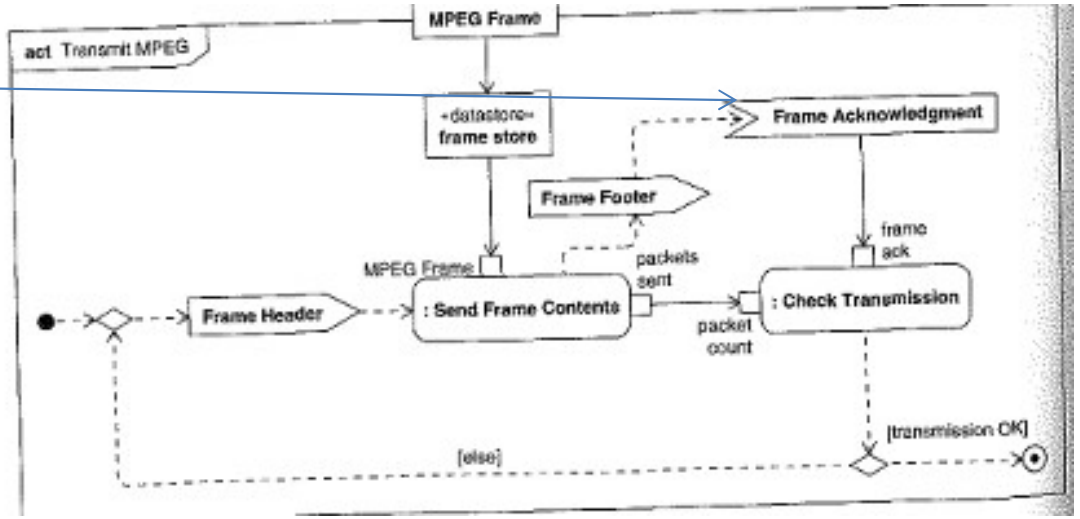
- An accept event action can accept other kind of events:
  - Time events
  - Change events

Examples

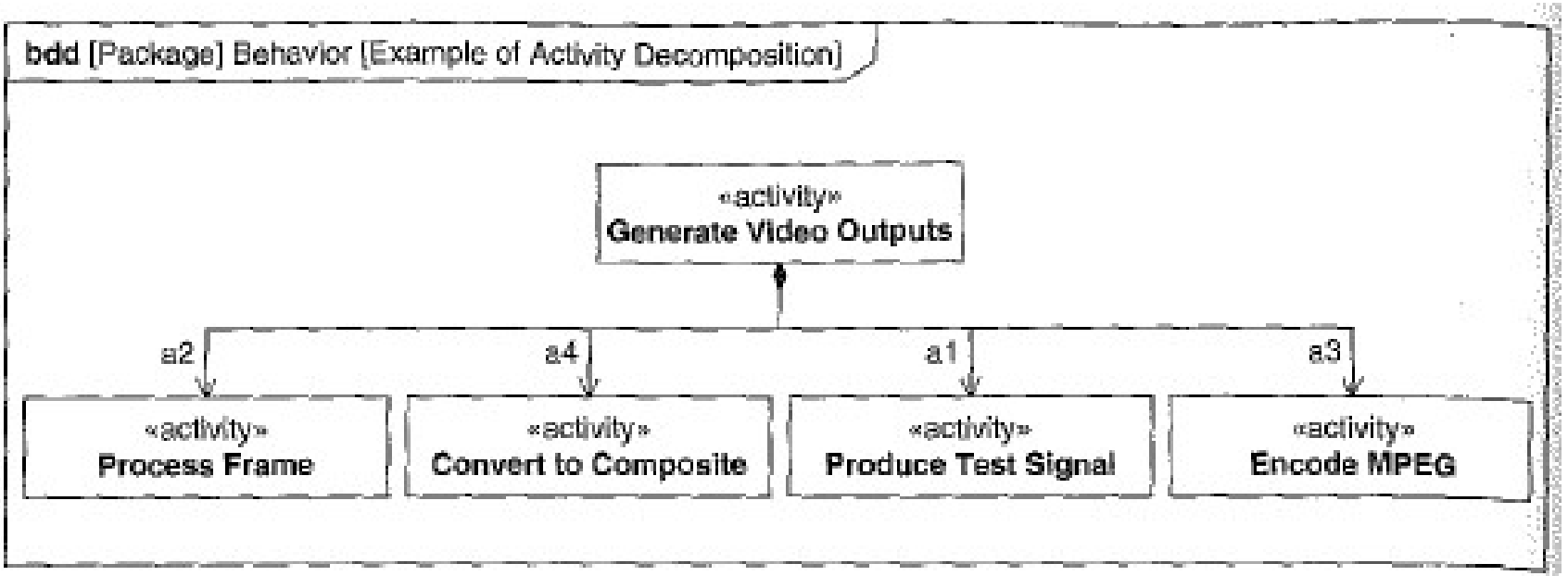
Time event



Change event



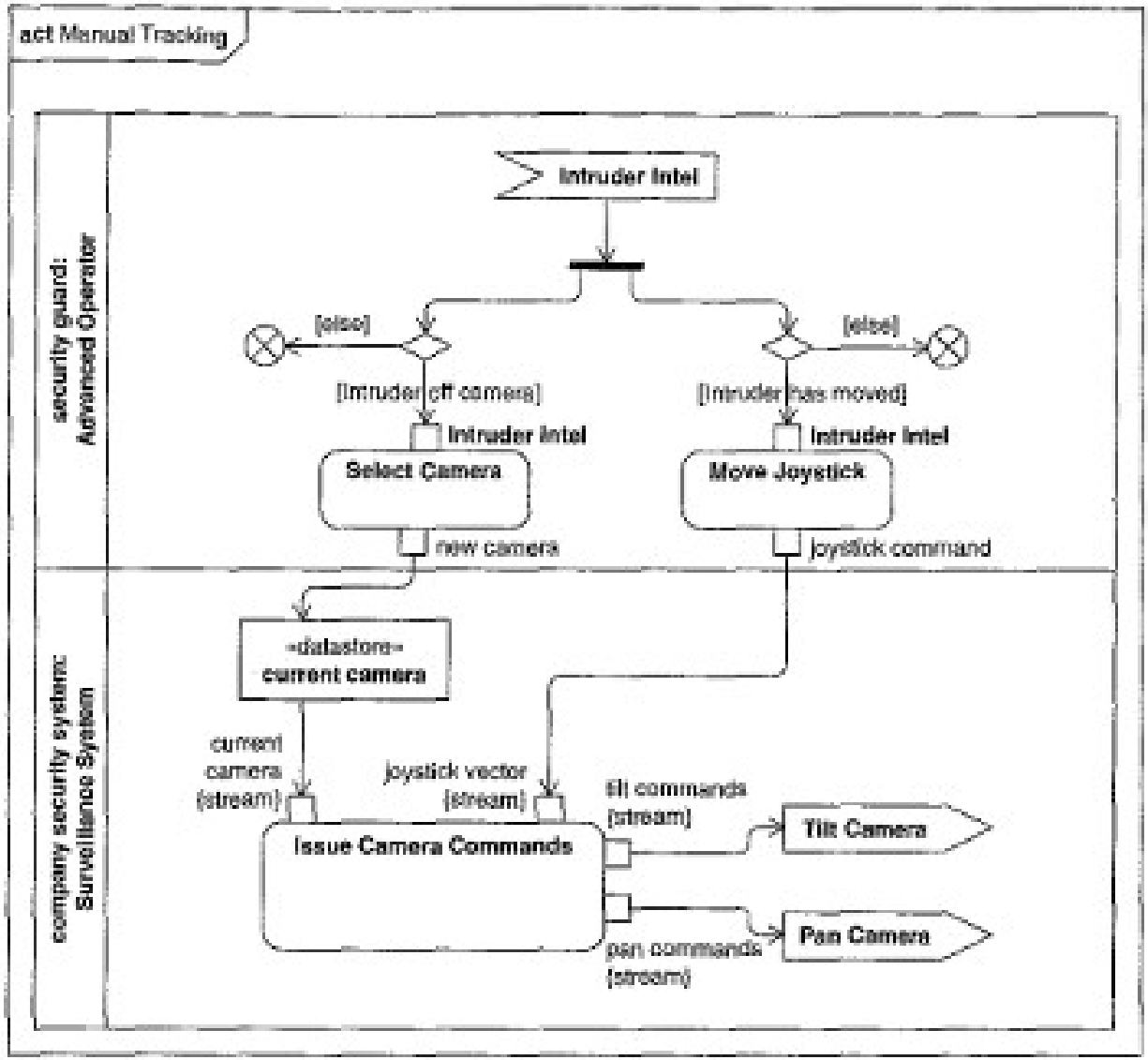
Activity Hierarchy



Relating Activities to Blocks

- Use an activity partition(swimlane) to assert that a given block (or part) is responsible for the execution of a set of actions:
  - A rectangular which contains the set of actions and has a header containing the name of the header/model responsible for the actions
- Make a block to own an activity and use this as a basis for specifying the block's behavior:
  - Use accept signals to connect the block with the activity

Example with swimlane



Summary of Activities

- An activity represents a controlled sequence of actions that transforms its inputs to its outputs
- Inputs and outputs are called parameters
- Activities are composed of actions
- Actions consume input tokens and produce output tokens via pins
- Actions are connected by flows:
  - Object flows: route tokens from input to output pins
  - Control flows: transfer control from one action to another

SysML Sequence Diagram

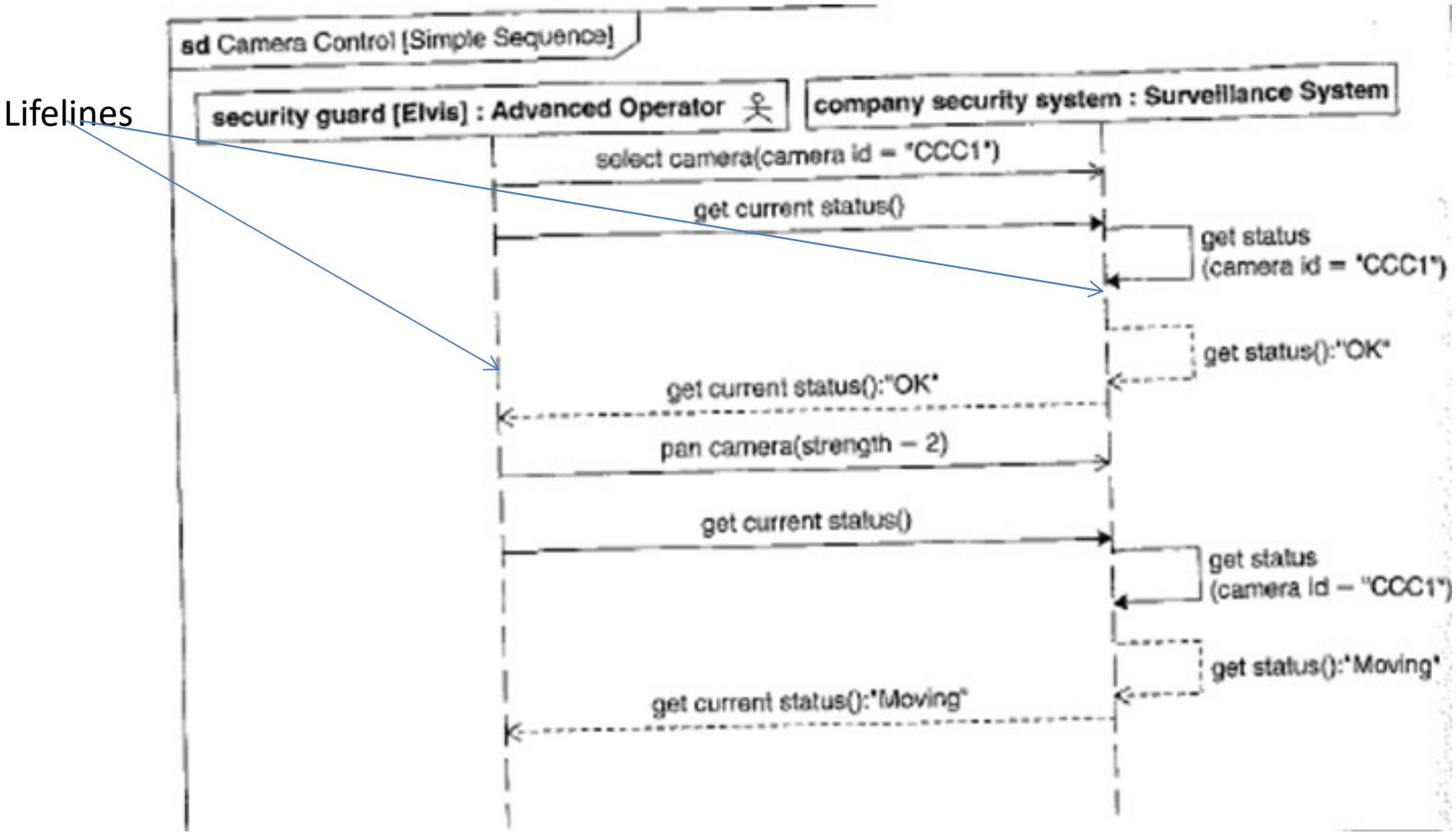
- models how parts of a block interact by exchanging messages or how the system interacts with the environment

- a message can represent the invocation of a service on a system component or the sending of a signal (synchronous vs. asynchronous)

- Header:

sd [Interaction] interaction name [diagram name]

Example


Lifelines
sd Camera Control [Simple Sequence]
security guard [Elvis] : Advanced Operator
company security system : Surveillance System
select camera(camera id = "CCC1")
get current status()
get status
(camera id = "CCC1")
get status():"OK"
get current status():"OK"
pan camera(strength = 2)
get current status()
get status
(camera id = "CCC1")
get status():"Moving"
get current status():"Moving"

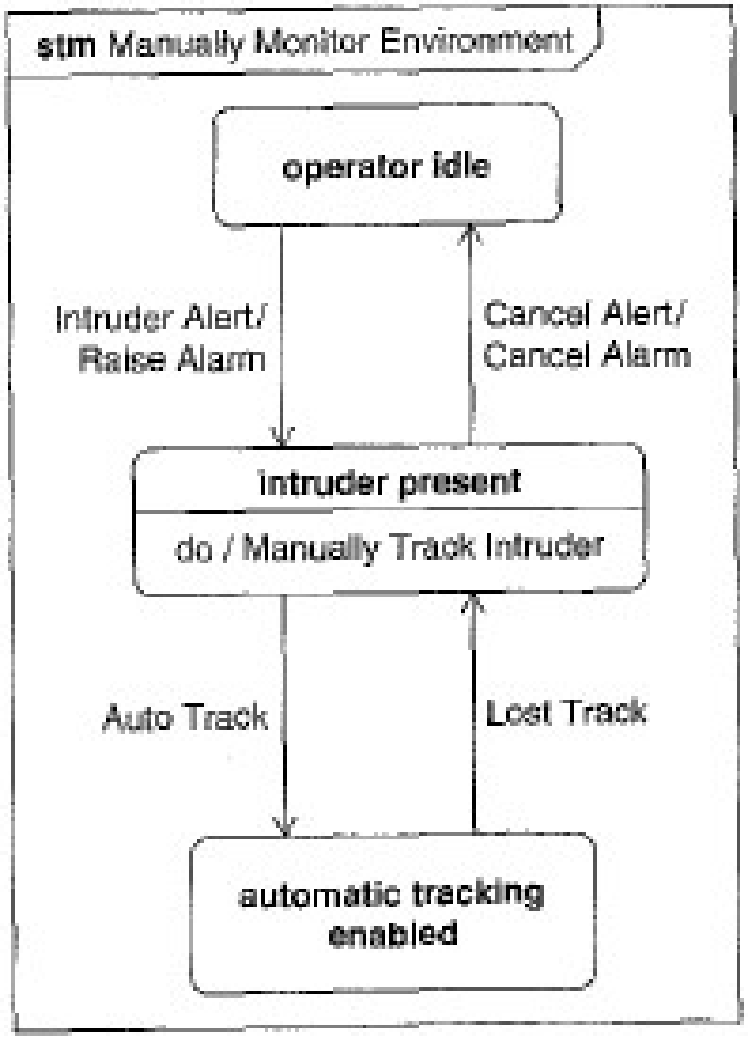
State Machine Diagram

- Describes the state-dependent behavior of a block throughout its lifecycle in terms of its states and transitions between them

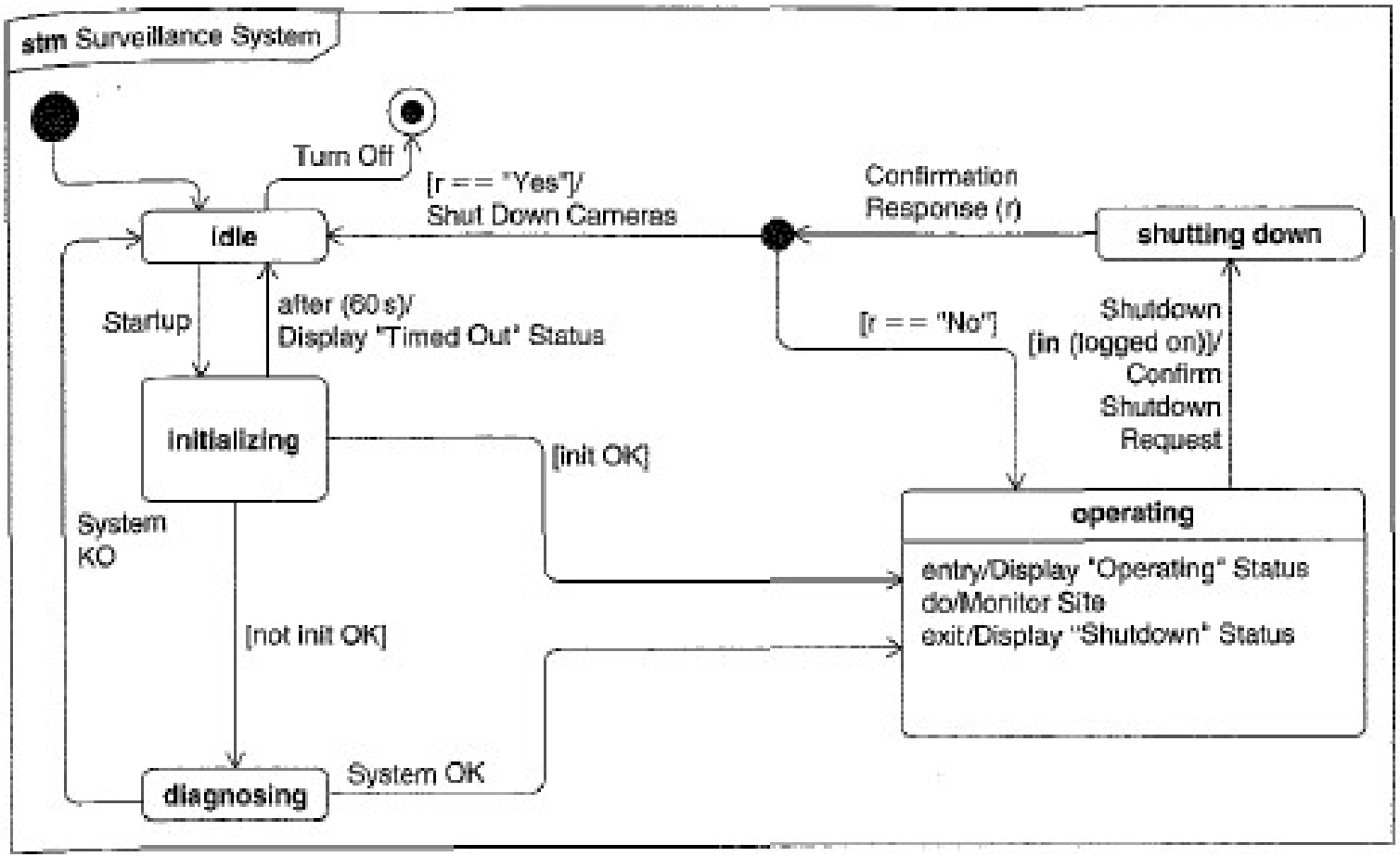
- Header:

stm [State-Machine] state machine name [diagram name]

Example


stm Manually Monitor Environment
operator idle
Intruder Alert/
Raise Alarm
Cancel Alert/
Cancel Alarm
intruder present
do / Manually Track Intruder
Auto Track
Lost Track
automatic tracking
enabled

Example

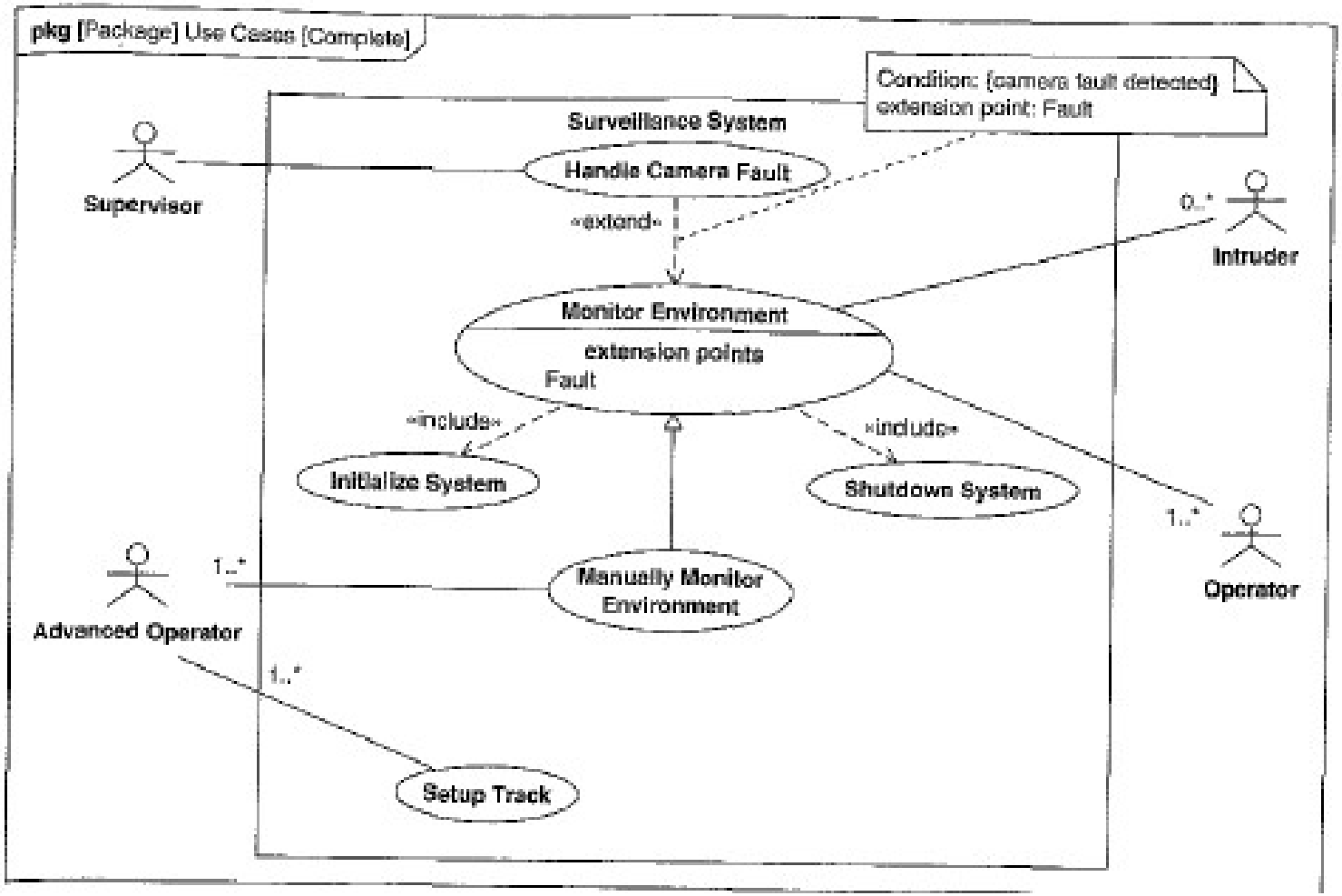

stm Surveillance System
Turn Off
[r == "Yes"]/
Shut Down Cameras
Confirmation
Response (r)
Idle
shutting down
Startup
after (60 s)/
Display "Timed Out" Status
[r == "No"]
Shutdown
[in (logged on)]/
Confirm
Shutdown
Request
initializing
[init OK]
operating
System
KO
entry/Display "Operating" Status
do/Monitor Site
exit/Display "Shutdown" Status
[not init OK]
diagnosing
System OK

Use Case Diagram

- describes the functionality of a system in terms of how its users use that system to achieve their goals
- the frame is a package or block, containing:
  - Set of actors
  - Use cases
  - Relationships
    - Actor-use case
    - Use case-use case
      - Inclusion
      - Extension
      - Classification
  - Descriptions (as notes): conditions that must occur for the use case to begin, postconditions,...

- Header: uc[model element type] model element name [diagram name]

Example


pkg [Package] Use Cases [Complete]
Condition: {camera fault detected}
extension point: Fault
Surveillance System
Handle Camera Fault
Supervisor
«extend»
0..*
Intruder
Monitor Environment
extension points
Fault
«include»
«include»
Initialize System
Shutdown System
1..*
Operator
1..*
Manually Monitor Environment
Advanced Operator
1..*
Setup Track