Introduction to Systems Engineering

Dr. Juan R. Pimentel
President
LeSoft
Retired Professor
Electrical and Computer Engineering
Kettering University

Introduction to Systems Engineering

Day 1 – Introduction to systems engineering

- Introduction to course
- Systems life cycle
- Introduction to Systems Engineering
- The V model of Systems Engineering
- Major Phases: Requirements, Conceptual Design, Preliminary and Detailed Design, Construction, Production, and Utilization phases
- The four pillars of SysML: Structure, Behavior, Requirements, and Parameters
- Demo of SysML: Structure, behavior, requirements, and parametrics for an Anti-Lock Braking System (ABS) model using Modelio.
- Introduction to Self-Driving Vehicles
- Introduction to the Safety of Self-Driving Vehicles

Introduction to Course

- Currently, many systems and products are very complex or very large or both.
- One example is automated vehicles with a high degree of safety.
- AV Safety is a major concern for:
  - OEMs (vehicle manufacturers) and their Suppliers
  - Drivers
  - Insurance companies
  - Governments
  - General public
  - Other stakeholders

- Thus the importance of designing and developing an autonomous vehicle with a high degree of safety.
- Systems engineering is a methodology and set of techniques to design systems that are very complex and/or very large.

Why does Safety matters?

- Uber accident killing one pedestrian in Arizona (March 2018)
- Series of Tesla accidents involving AUTOPILOT (Early 2018)
- Tesla accident killing one driver on a Florida highway (April 2016)
- Tesla accident involving rear end collision with a parked truck (2017)
- Accident in Las Vegas (2017)
- Problems with airbag activated without command (Japanese Supplier 2014)
- Problems with vehicle recognition for automated driving (American OEM 2016)

Introduction to Course

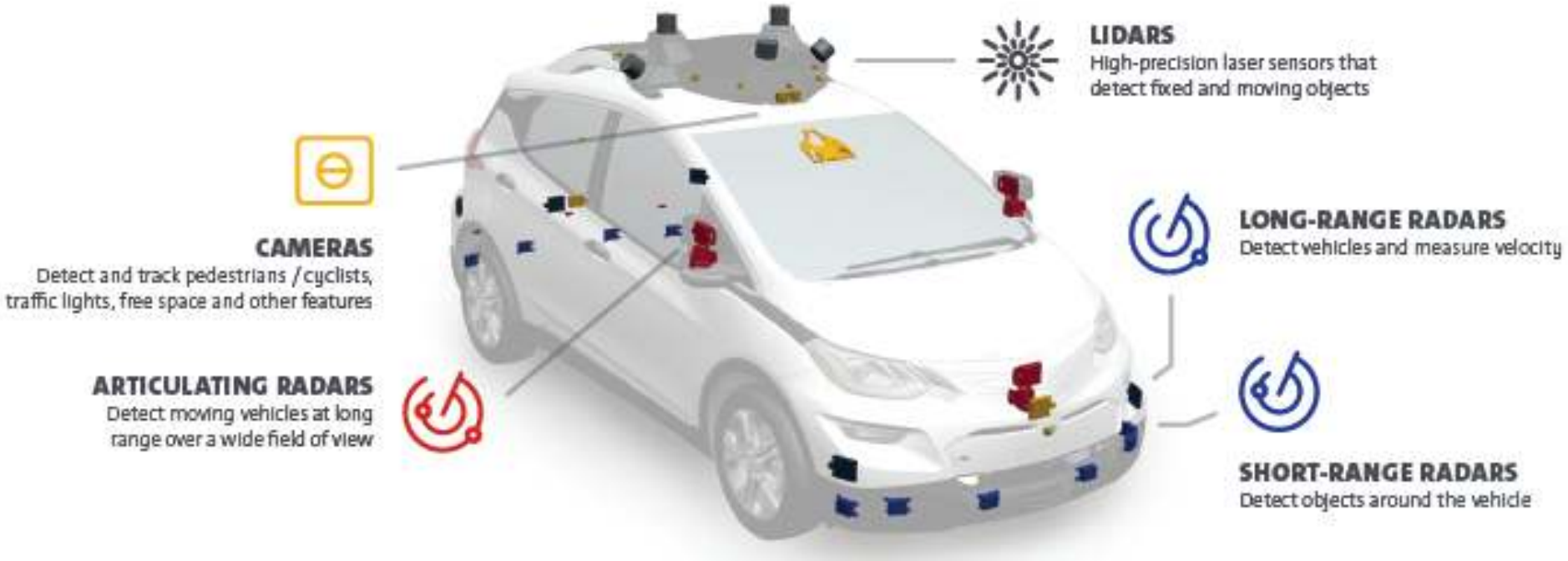
- Safety has been extensively studied and standards exist for several industrial areas such as:
- Nuclear, avionics, process, farmaceutical, manufacturing, and automotive.

- **Question:** Can we design a complex system without using a systems engineering approach?

- **Answer:** Yes but it would be much more difficult. There are many advantages of using a systems engineering approach.
- But safety is not the only reason to use systems engineering. There are others!!

Introduction to Course

Unique safety requirements of Avs, Oil & Gas.



- **Defining systems thinking**
- "Systems thinking is a discipline for seeing wholes.
- It is a framework for seeing interrelationships rather than things, for seeing patterns of change rather than static "snapshots."
- It is a set of general principles— distilled over the course of the twentieth century, spanning fields as diverse as the physical and social sciences, engineering, and management....
- During the last thirty years, these tools have been applied to understand a wide range of corporate, urban, regional, economic, political, ecological, and even psychological systems.
- And systems thinking is a sensibility — for the subtle interconnectedness that gives living systems their unique character."

- - Peter Senge, *The Fifth Discipline*

Systems Lifecycle

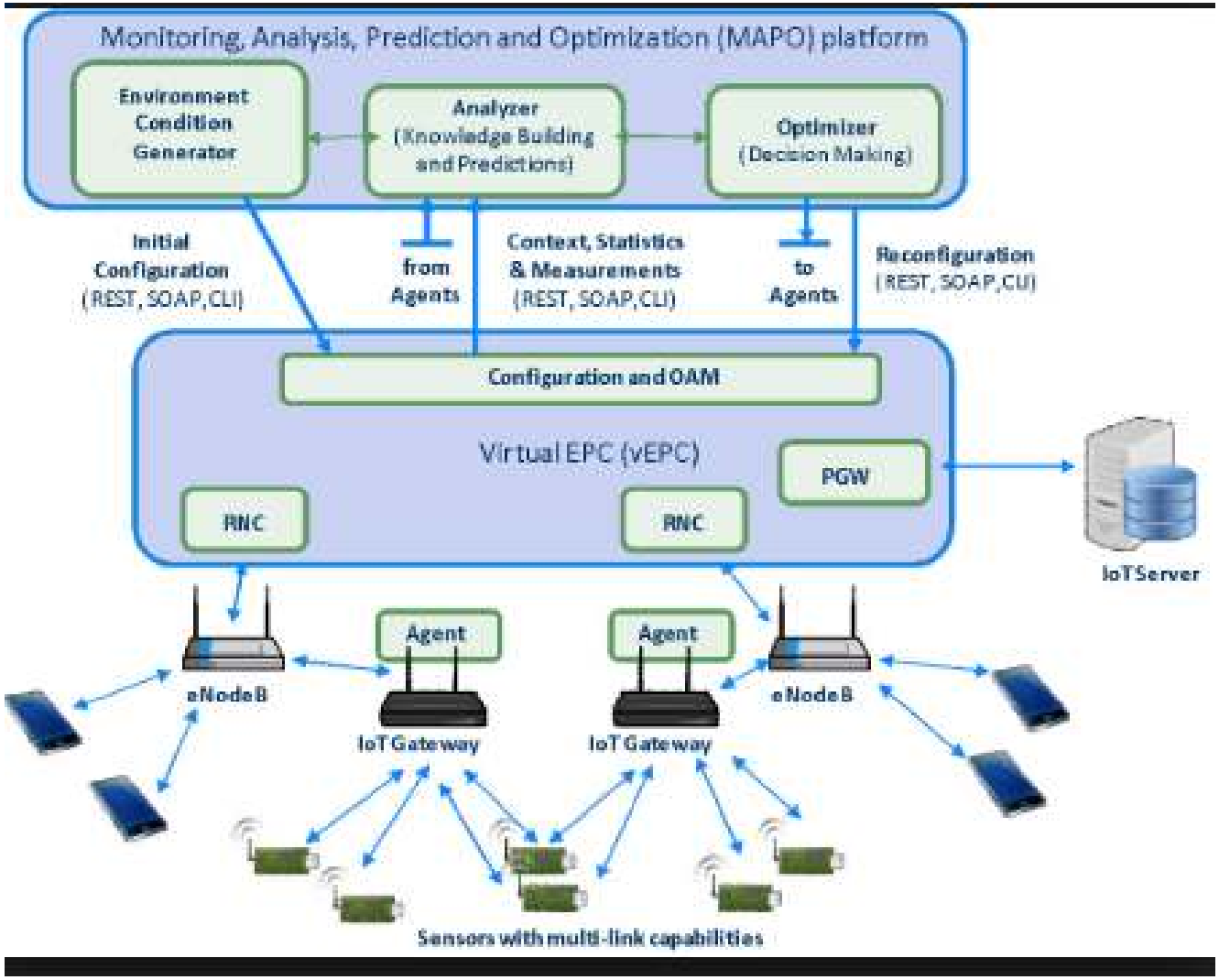
- Concept Phase
  - Item Definition
  - Initiation of safety lifecycle
  - Hazard analysis and risk assessment
  - Functional Safety Concept
- Product Development
  - System Level
  - Hardware
  - Software
- Production and Operation
  - Production
  - Operation
  - Service (Maintenance, Repair)
  - Decommissioning

Introduction to Systems Engineering (SE)

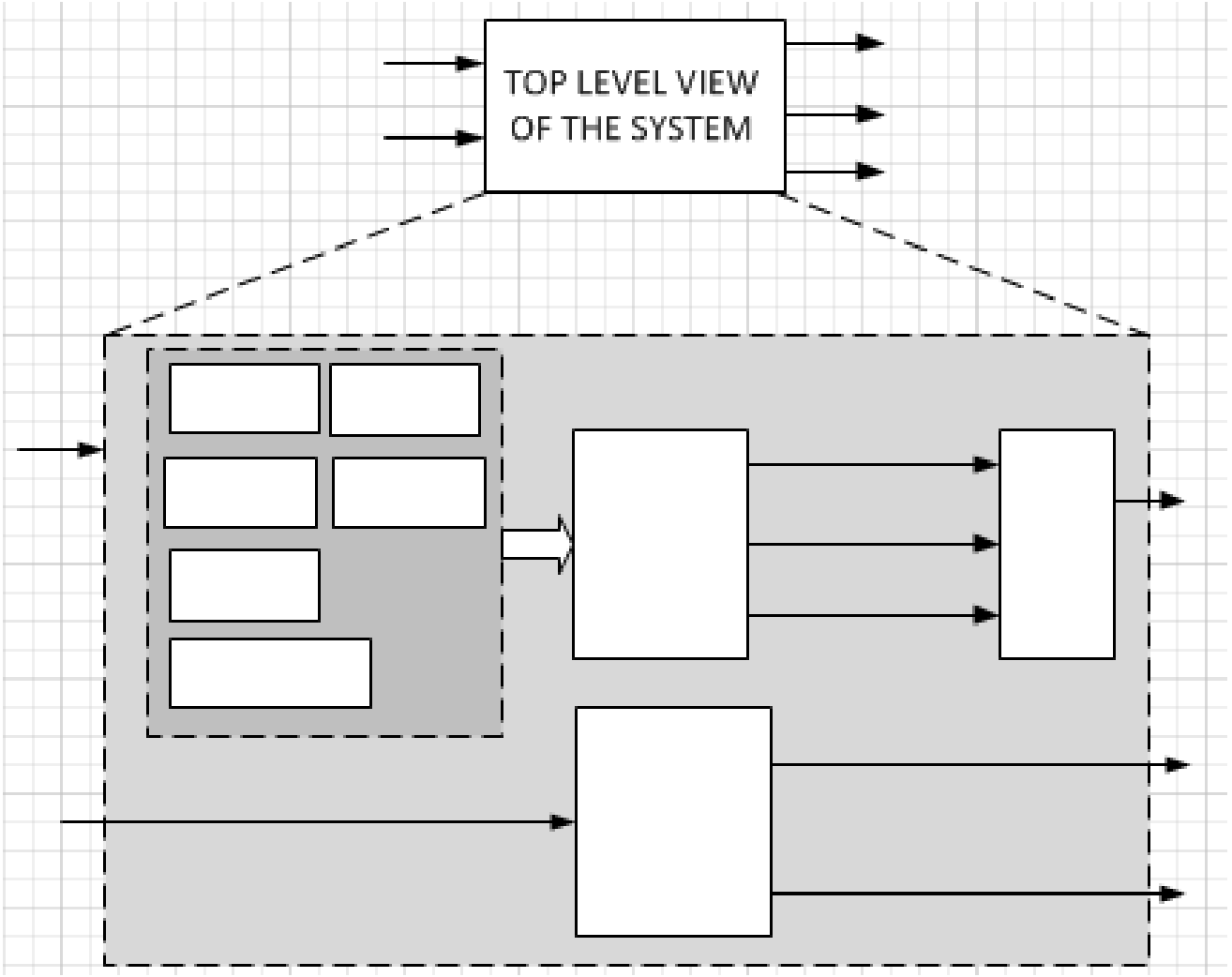
- Although SE can be applied to any endeavor (e.g., physical and social sciences, engineering, and management, etc.) we will apply it to engineering and more specifically, designing an automated vehicle.

- Main principles:
  - Start at highest level
  - Identify main functions using input/output relationships
  - Use functional decomposition and generate a level below.
  - Proceed likewise by generating 4 to 8 levels deep.
  - At each level add sufficient details to clarify composition, relationships, require ments, behaviors, interfaces, parameters, etc.

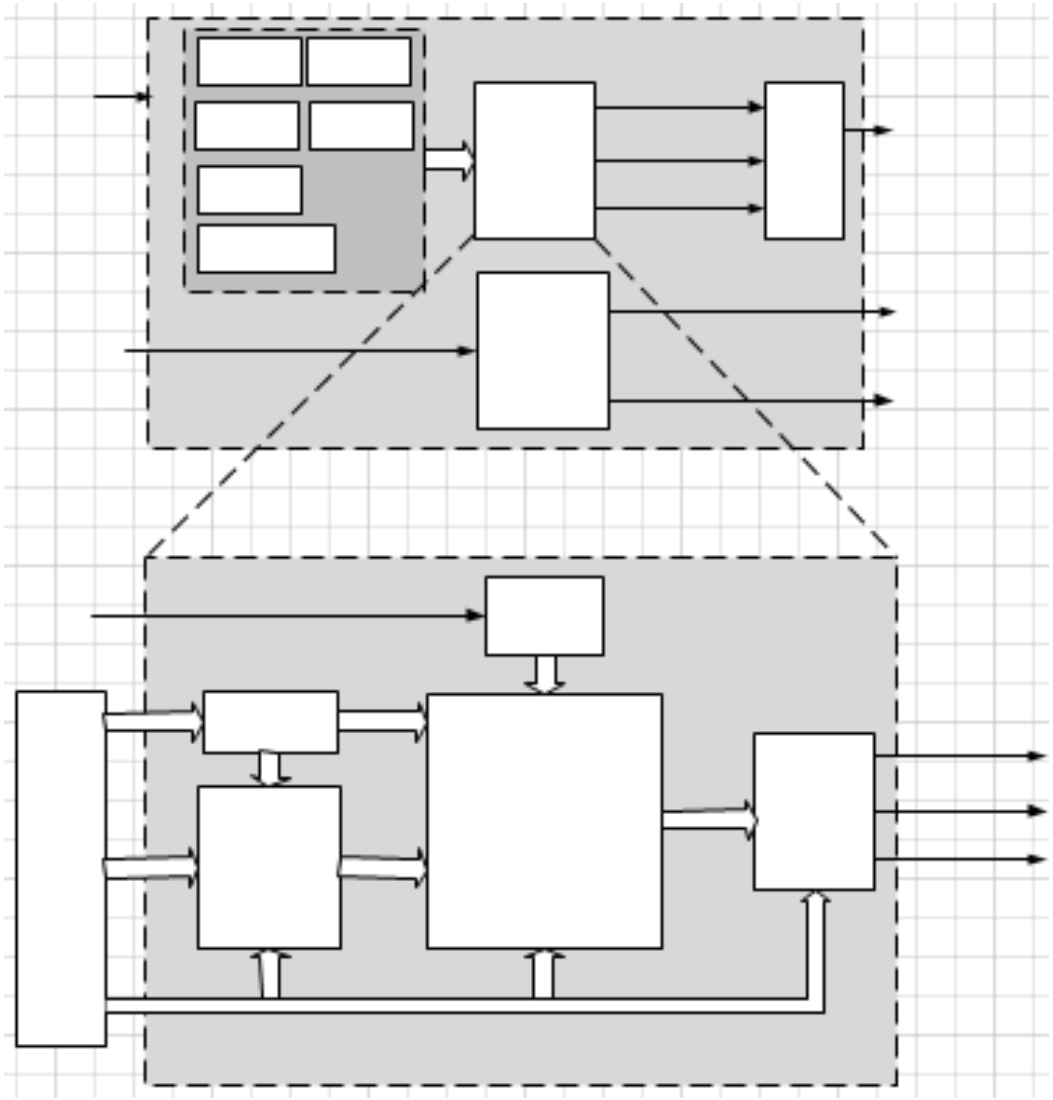
Architecture of complex systems: Hierarchical organization



Introduction to Systems Engineering (SE)



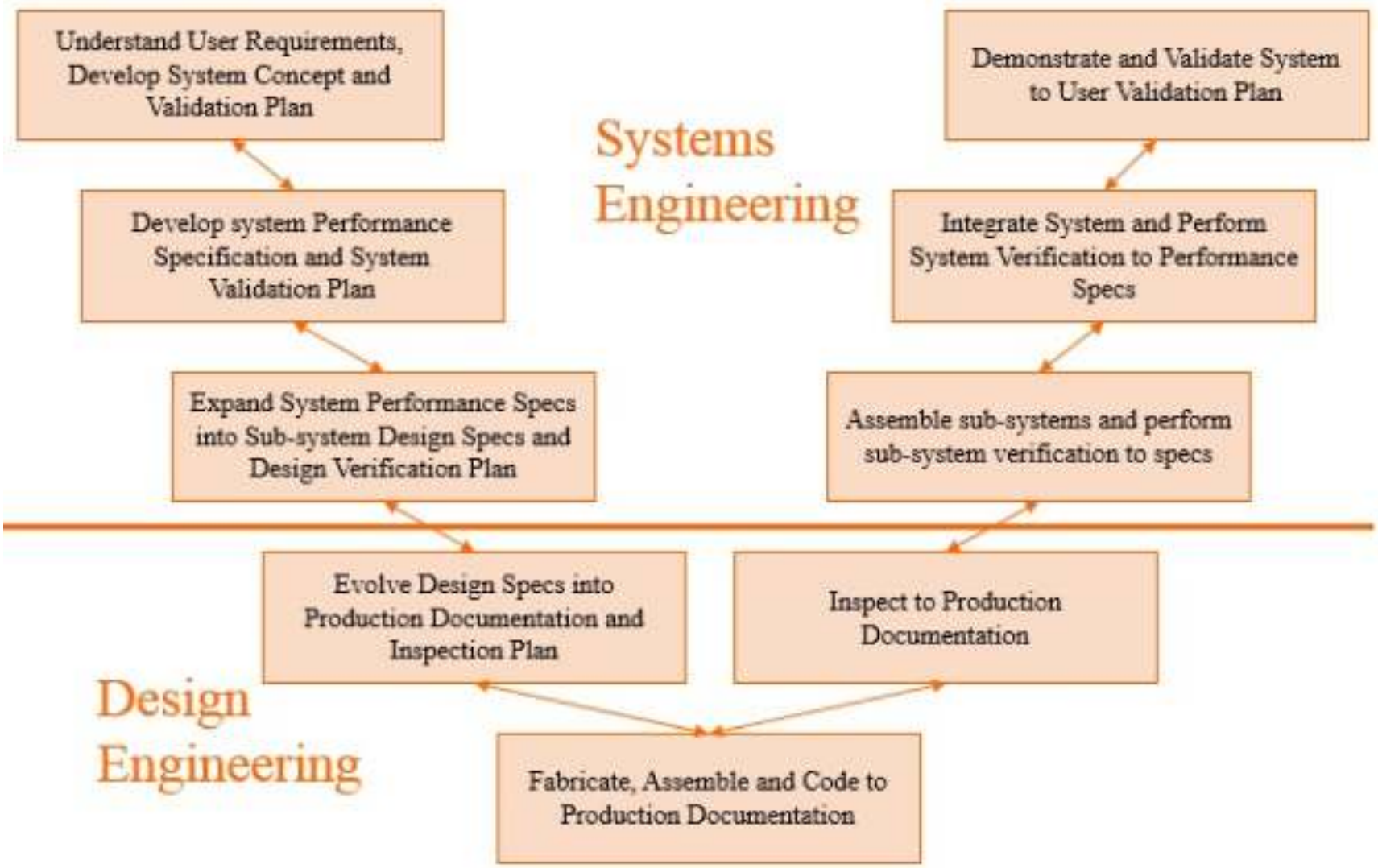
- Introduction to Systems Engineering (SE)



The V model of Systems Engineering

- Forsberg and Mooz were the first to model the SE processes as a V.
- For complex, multidisciplinary systems, the V model provides a nice framework for the key systems engineering activities it takes to realize a successful system.
- On the left side of the V, decomposition and definition activities resolve the system architecture and create the details of the design.
- Integration and verification flow up and to the right as successively higher levels of subsystems are verified, culminating at the system level.
- **Verification** ensures the system *was built right* (meets requirements), whereas **validation** ensures the *right system was built* (meets customers' needs).

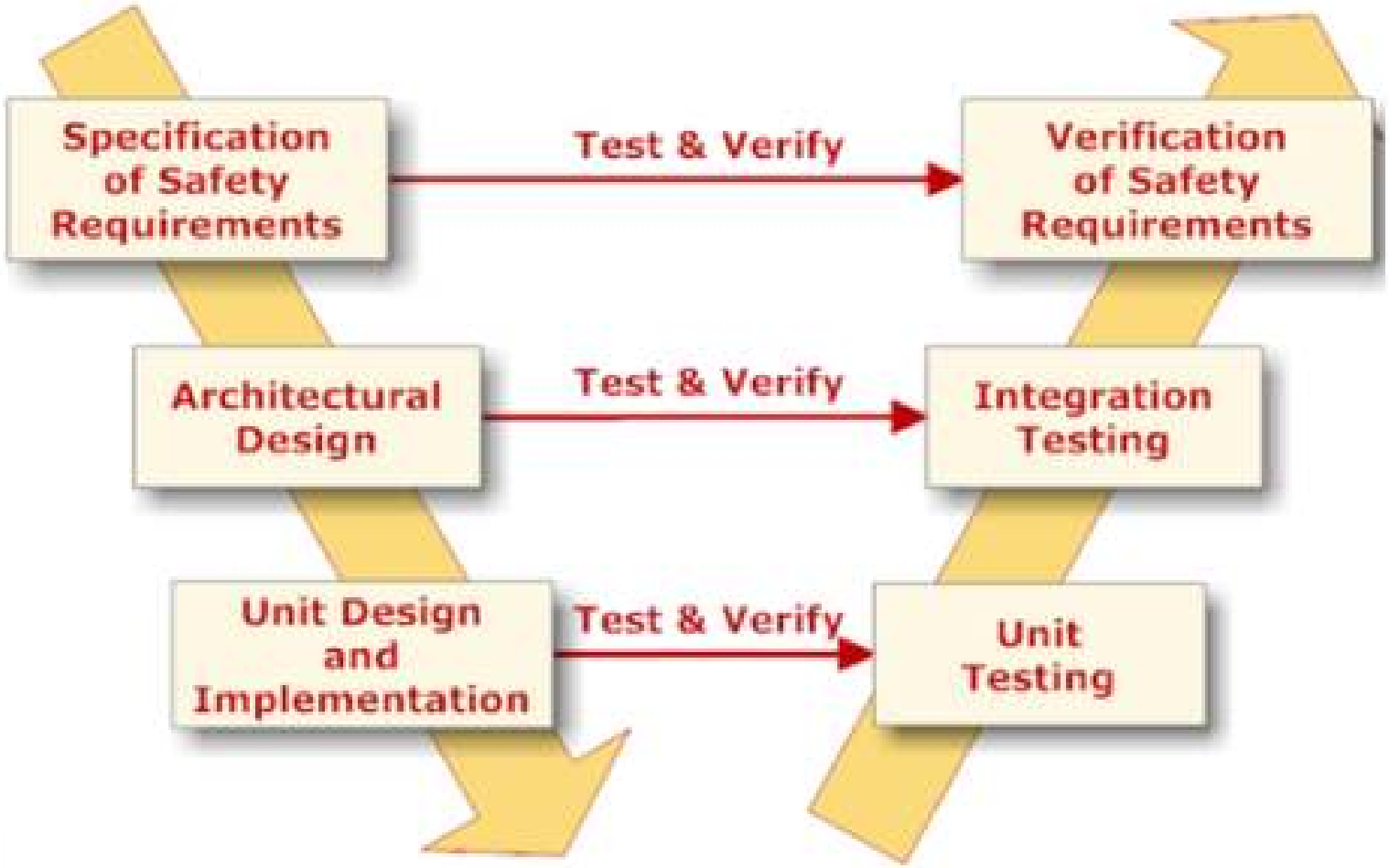
The V model of Systems Engineering



The V model of Systems Engineering

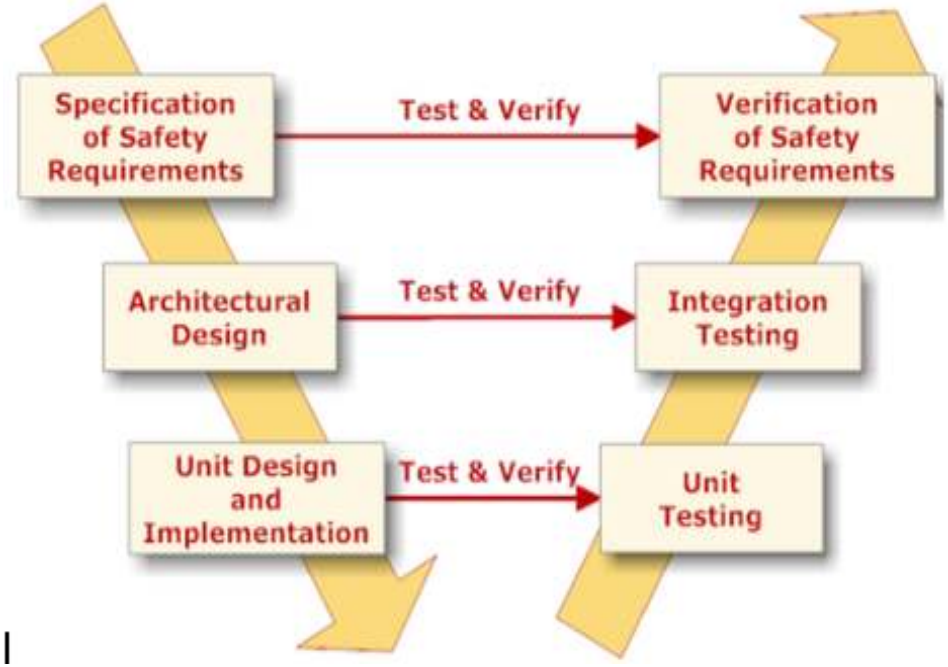
- Ascending the right side of the V is the process of integration and verification.
- At each level, there is a direct correspondence between activities on the left and right sides of the V.
- This is deliberate - the method of verification must be determined as the requirements are developed and documented at each level.
- This minimizes the chances that requirements are specified in a way that cannot be measured or verified.

The V model of Systems Engineering

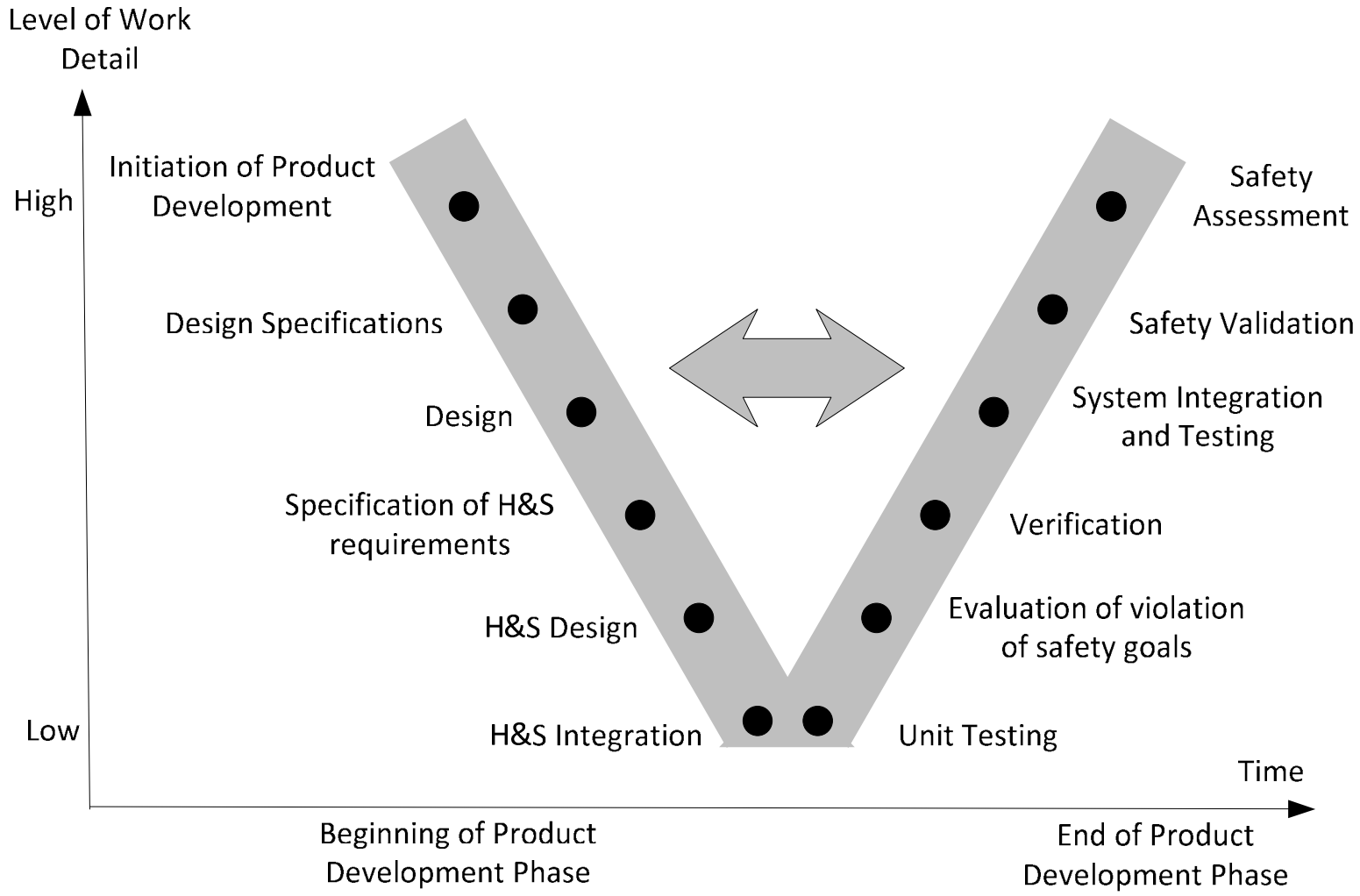


The V model of Systems Engineering

- The safety components of a system constitute yet another complex system and ISO 26262 uses the V model as the underlying framework.
- This decision minimizes unnecessary rework, errors in requirements development and cascading.
- This will force testing and verification to occur at various levels of integration early in the process
- More specifically, ISO 26262 uses the V model at the system level (Section 4), at the hardware level (Section 5) and at the software level (Section 6).



Product Development Phases

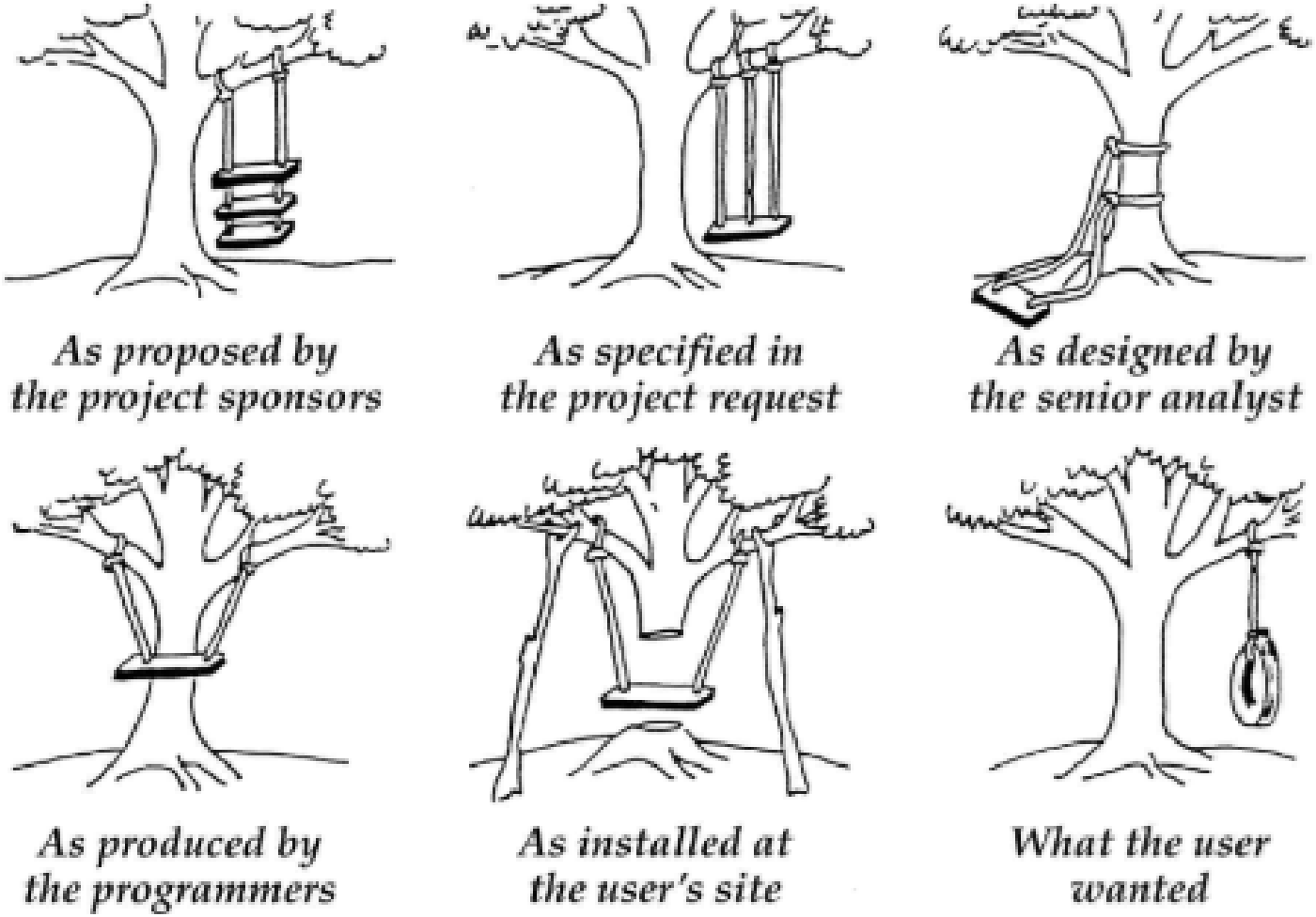


Major Phases: Requirements

- Important phase that takes place at the beginning of a project.
- There are top level requirements
- And requirements at all functional levels
- Some requirements are further defined or extended into "derived requirements"
- This process is repeated appropriately.
- Two types of requirements:
  - Functional
  - Non-Functional
- Requirements need to be carefully selected in order to ensure that they make sense in the context of the final outcome of the project and conveyed to all the team members.
- Missing out on a requirement or misapplying one could spell disaster for a project.

Major Phases: Requirements

"Problem solving is an art form not fully appreciated by some"



Major Phases: Requirements (Functional)

- Any requirement which specifies **what** the system should do.
- Describes a particular behaviour of function of the system when certain conditions are met, for example: "Send email when a new customer signs up" or "Open a new account".
- A functional requirement for an everyday object like a cup would be: "ability to contain tea or coffee without leaking".
- Typical functional requirements include:

Typical functional requirements include:

- Carry people on public streets and roadways.
- Carry people through air space
- External Interfaces
- Business Rules
- Transaction corrections, adjustments and cancellations
- Administrative functions
- Authentication
- Authorization levels
- Audit Tracking

Major Phases: Requirements (Non-Functional)

- Any requirement which specifies **how** the system performs a certain function.
- Describes how a system should behave and what limits there are on its functionality.
- Specify the system's quality attributes or characteristics, for example: "Modified data in a database should be updated for all users accessing it within 2 seconds."
- A non-functional requirement for the cup mentioned previously would be: "contain hot liquid without heating up to more than 45 °C".

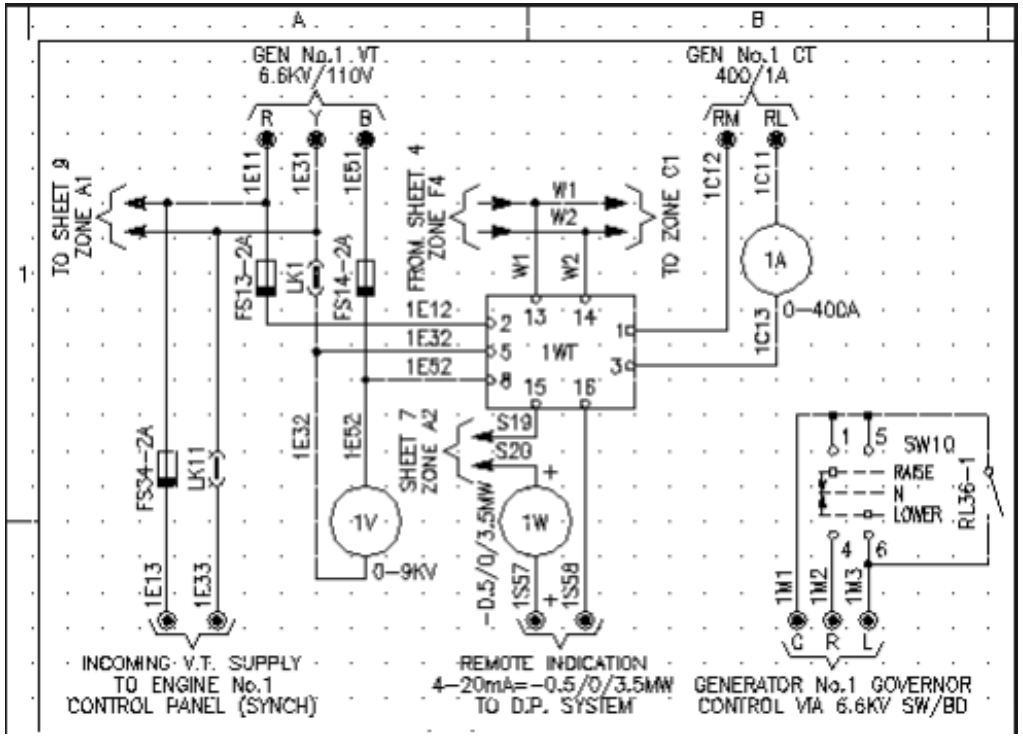
Examples:

- Performance – for example: response time, throughput, utilization, static volumetric
- Capacity
- Availability
- Reliability
- Recoverability
- Maintainability
- Serviceability
- Safety, Security
- Regulatory
- Environmental
- Data Integrity
- Interoperability

Major Phases: Preliminary and Detailed Design

- Intermediate and final designs.
- It includes details and specifics for its implementation.
- With sufficient detail to be implemented or produced by other parties.
- Based on many assumptions, calculations, measurements, simulations, etc.
- Produced using an assortment of tools:
  - Drawing
  - Simulation
  - System engineering
  - Etc.

Example of a detailed design



Major Phases: Construction & Production

- This is the phase where the system is built up using blueprints of the detailed design phase.
- When the construction is done in series, e.g., in a manufacturing plant, then it is termed production.
- Testing is an important activity to be done during construction.
- Types of tests:
  - Unit test
  - System test
  - Interoperability test
  - Stress test
  - Etc.

Major Phases: Utilization

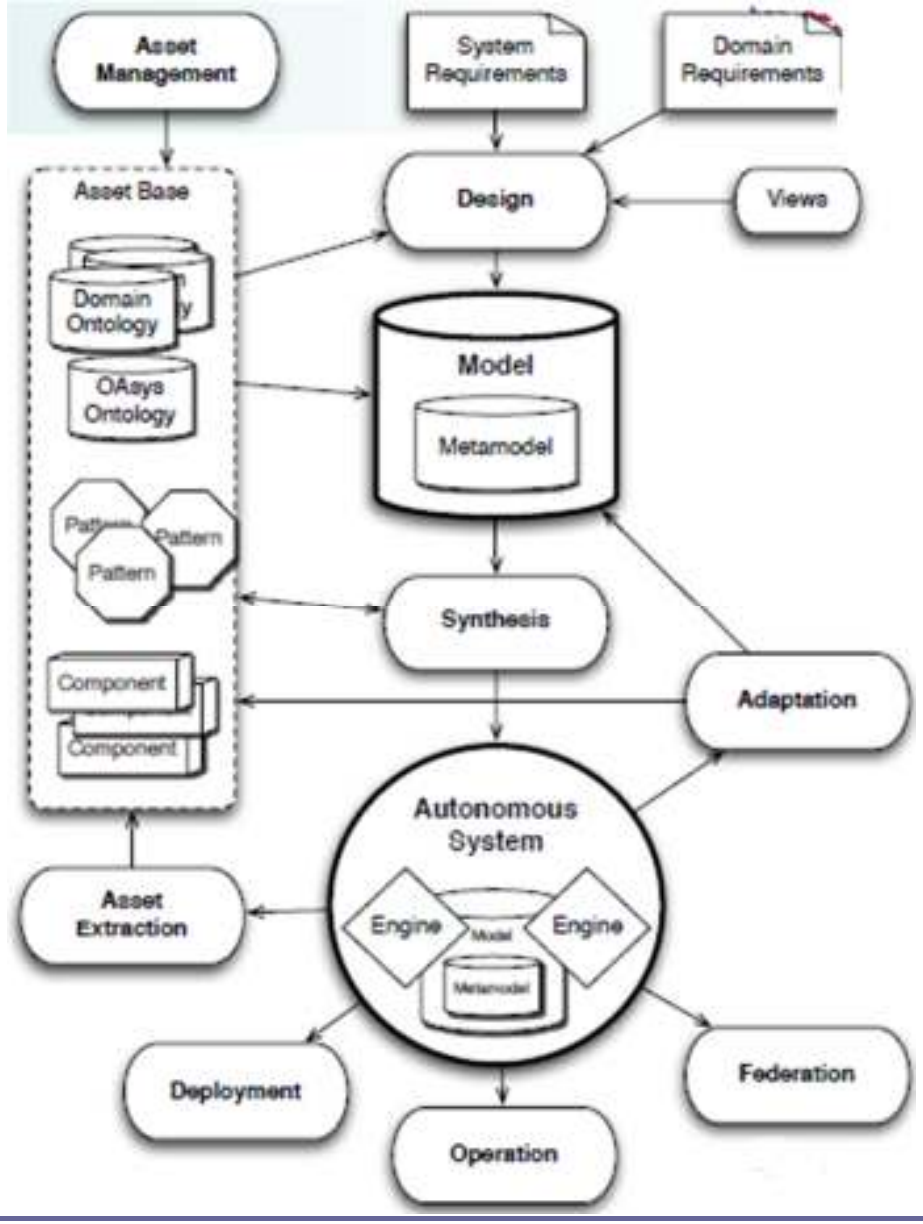
- This is the phase that starts when the system or product is put in service.
- It ends when the system is taken out of service or decommisioned
- Important activities:
  - Operation
  - Maintenance
  - Troubleshooting
  - Repair

Model based systems engineering (MBSE)

- Engineers **use system models** to support their activity.
- Autonomous systems (AS, or ASys) is one example of a complex system.
- ASys shall follow a pervasive model-based approach:
  - – An ASys will use models to perform its activity.
  - – An ASys will be built using models of it.
  - → An ASys can exploit its own very models in controling its behavior.
- Model-based engineering and model-driven behavior can then be merged into a single phenomenon: **model-based autonomy**.

Model based systems engineering (MBSE)

- An architecture centric domain engineering life-cycle
- Pervasively Model-based
- Using:
  - – Patterns
  - – Components
  - – Ontologies



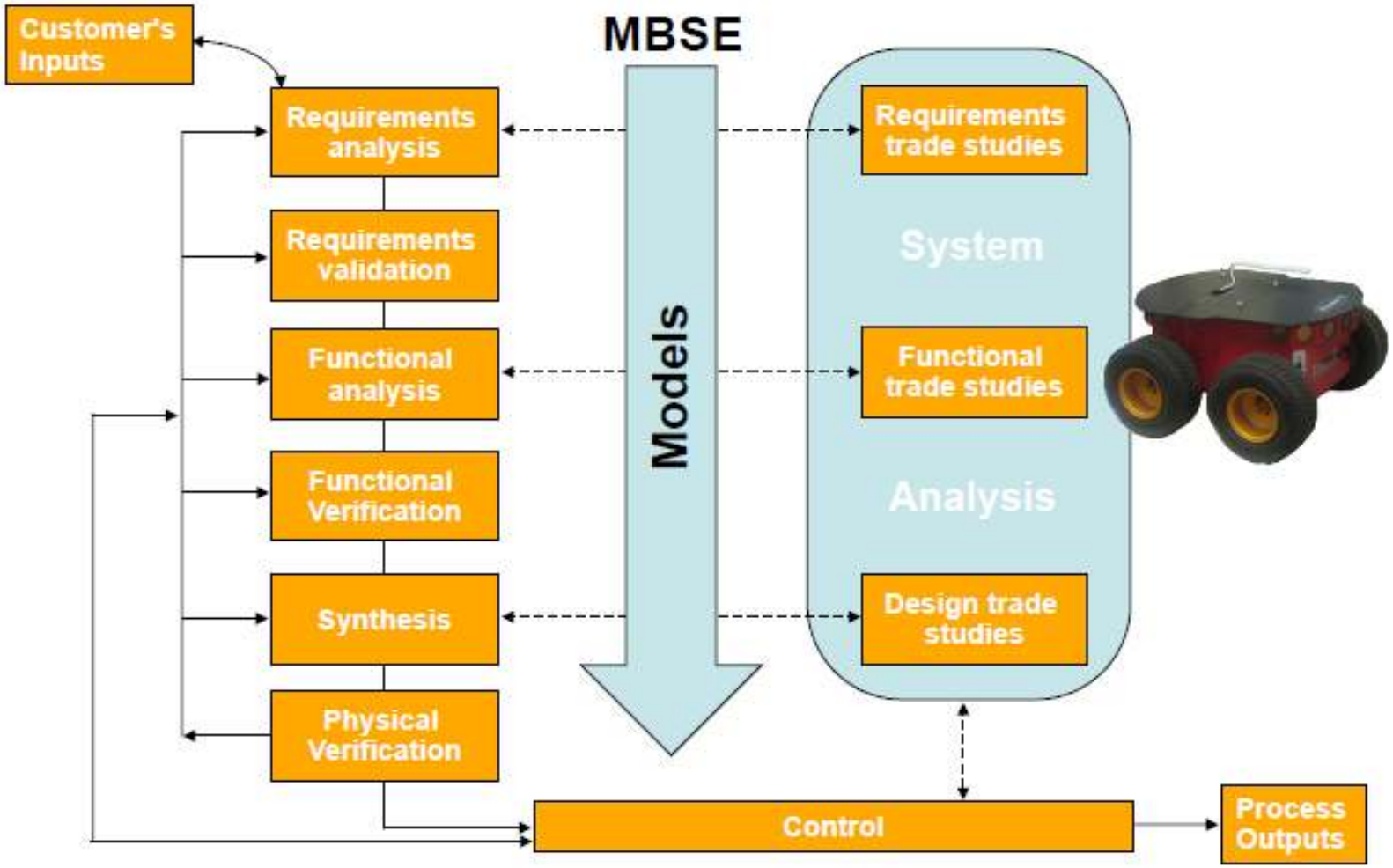
Model based systems engineering (MBSE)

A fundamental question

- What is the knowledge that an agent must have about itself in order to achieve mission-level robustness?
- Knowledge about:
  - – Goals
  - – Resources
  - – Constraints
  - – Performance
  - – Etc
- This is **Systems Engineering Knowledge**
- This is used in **MBSE**

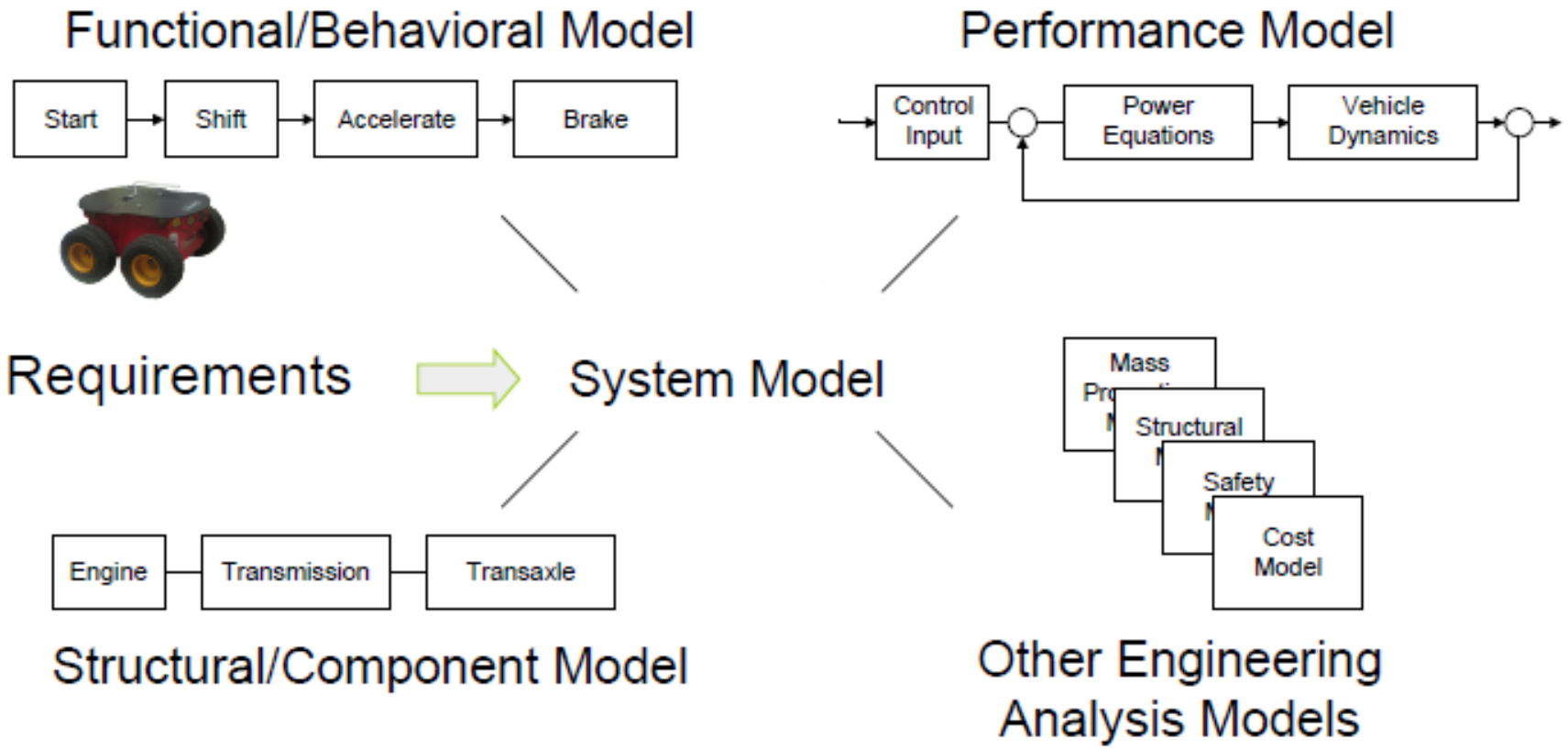
Model based systems engineering (MBSE)

Systems Engineering Process – IEEE 1220



Model based systems engineering (MBSE)

Aspects of system modeling



Integrated System Models Address Multiple Aspects of a System

Model based systems engineering (MBSE): Benefits

- Shared understanding of system requirements and design
  - – Validation of requirements
  - – Common basis for analysis and design
  - – Facilitates identification of risks
- Assists in managing complex system development
  - – Separation of concerns via multiple views of integrated model
  - – Supports traceability through hierarchical system models
  - – Facilitates impact analysis of requirements and design changes
  - – Supports incremental development & evolutionary acquisition
- Improved design quality
  - – Reduced errors and ambiguity
  - – More complete representation
- Supports early and on-going verification & validation to reduce risk
- Provides value through life cycle (e.g., training)
- Enhances knowledge capture

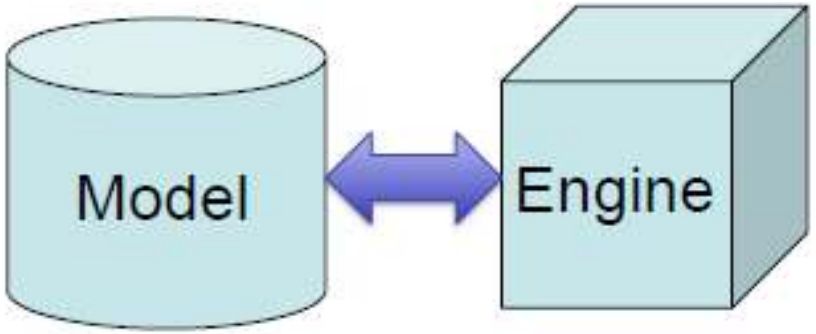
Model based systems engineering (MBSE): What models do we need?

- Two modeling strategies:
  - Use **task-specific models**: Models that can be used for a specific task.
  - Use **universal models**: Models that can be used for anything.

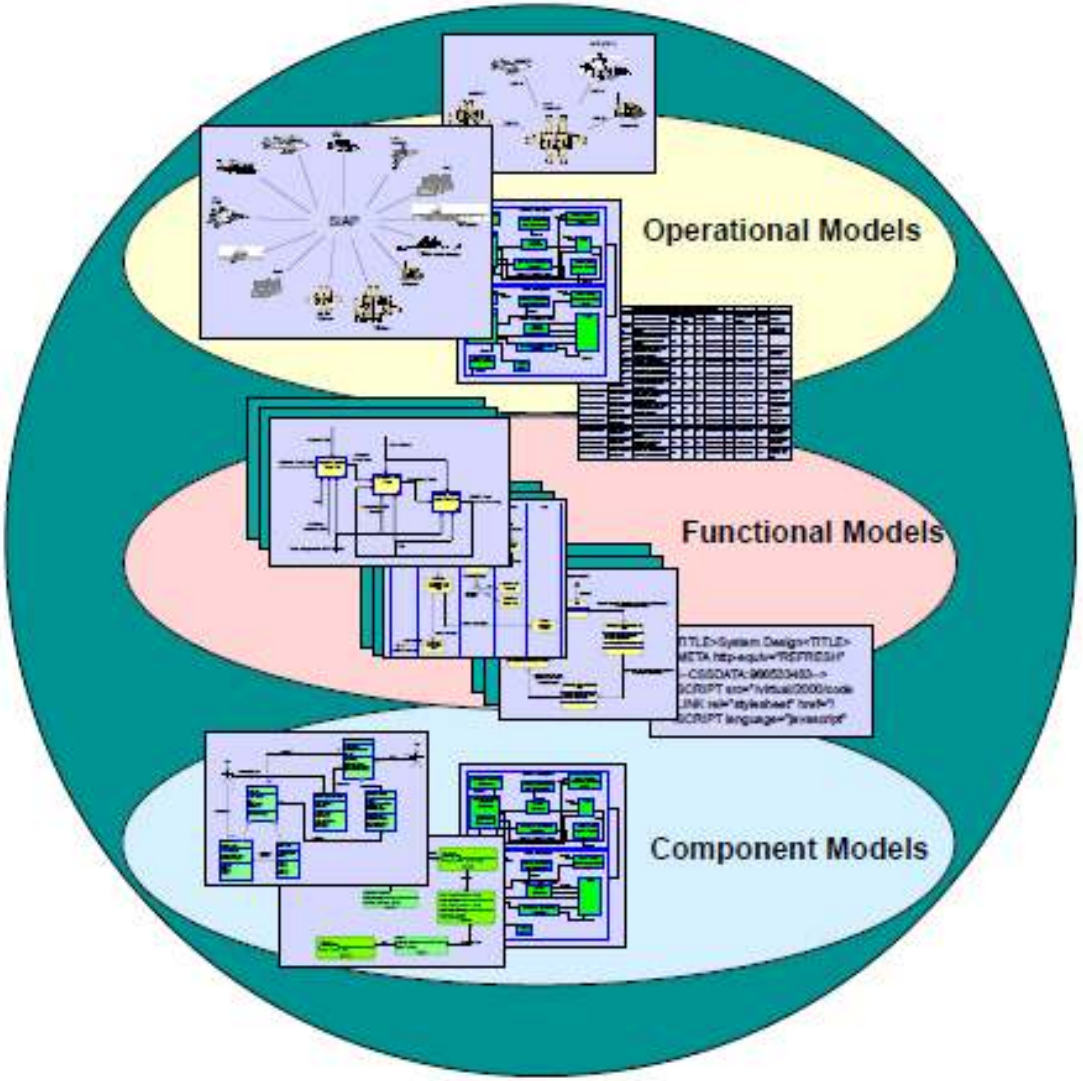
} *A modelling trade-off*

Both are needed in SE

- Note that models are **exercised**:
  - Simulators simulate (using the model)
  - Diagnosers diagnose (using the model)



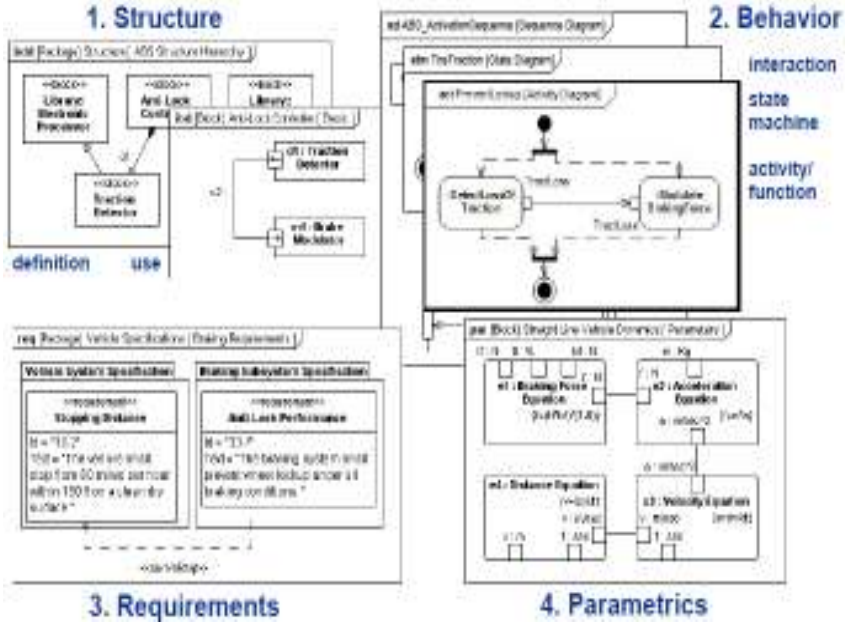
Model based systems engineering (MBSE): Multiple level models



The four pillars of SysML

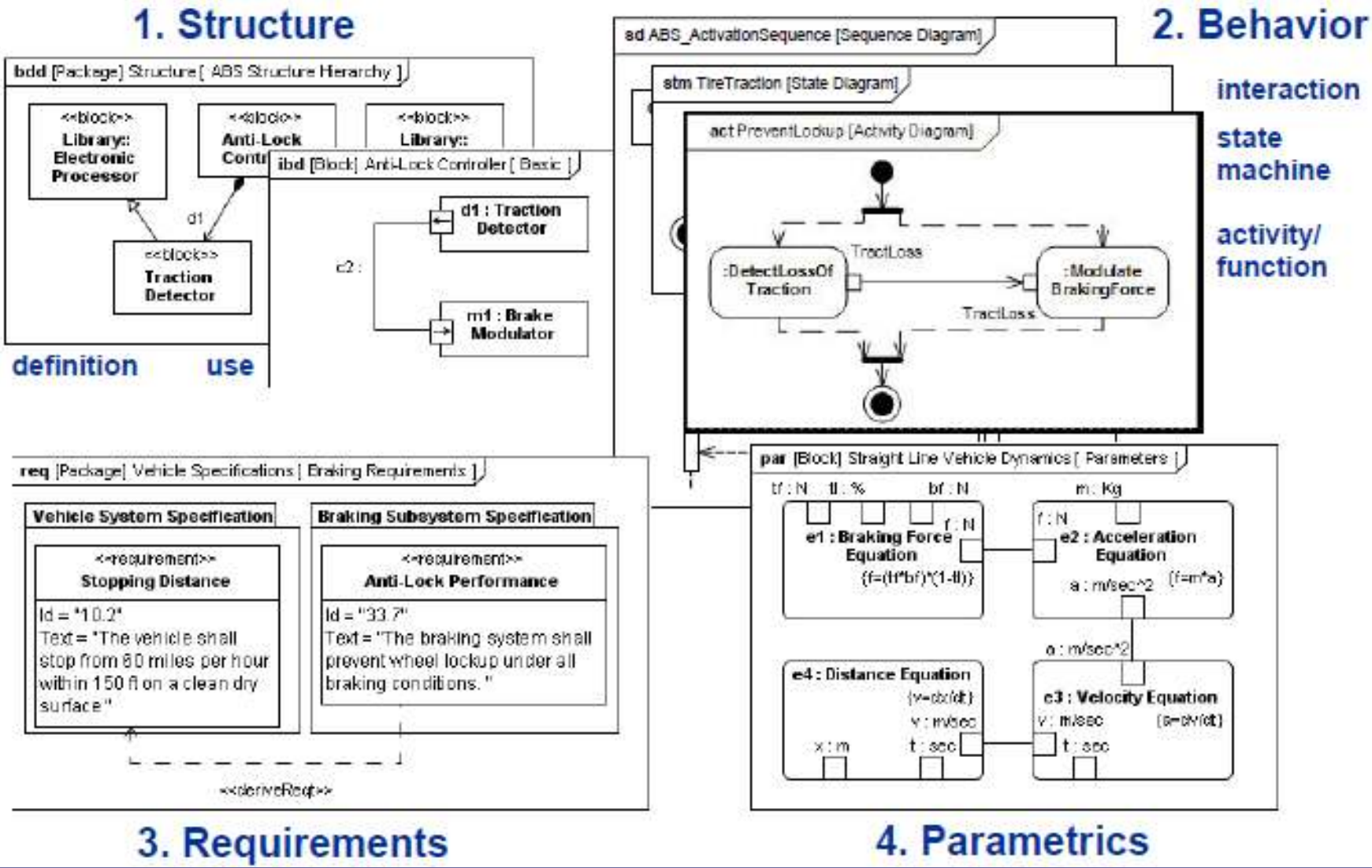
SysML = Systems Modeling Language

- **Structure**
  - What the system is
- **Behavior**
  - How the system behaves
- **Requirements**
  - What is expected from it
- **Parametrics**
  - Quantification beyond qualitativeness



- SysML supports the specification, analysis, design, verification, and validation of a broad range of complex systems.
- These systems may include hardware, software, information, processes, personnel, and facilities.

The four pillars of SysML



The four pillars of SysML
The 9 SysML Diagrams

Structure

- bdd – Block Definition diagram
- ibd – Internal Block diagram
- pkg – Package diagram

Behavior

- sd – Sequence diagram
- stm – State Machine diagram
- act – Activity diagram
- uc – Use Case diagram

Requirements

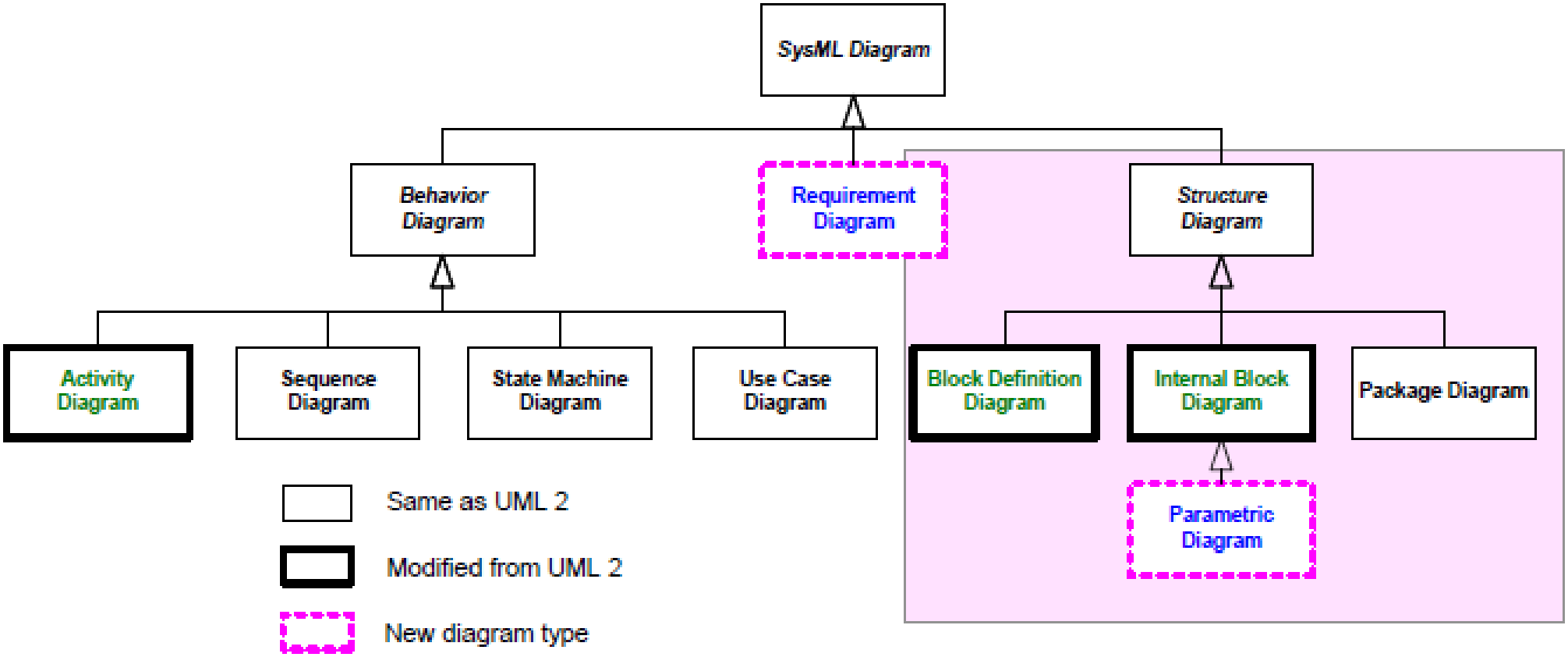
req – Requirement diagram

Parametrics

par – Parametric diagram

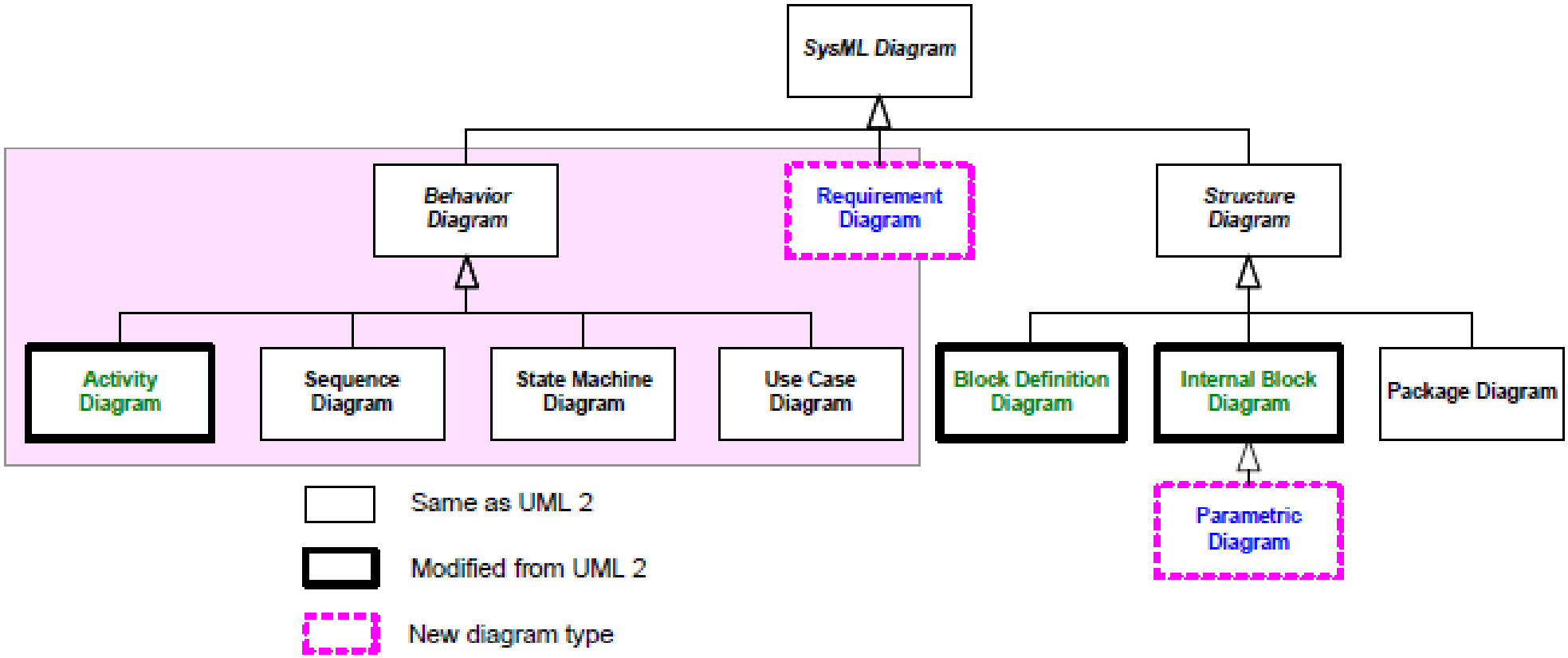
SysML as an example language for MBSE

Structural Diagrams



SysML as an example language for MBSE

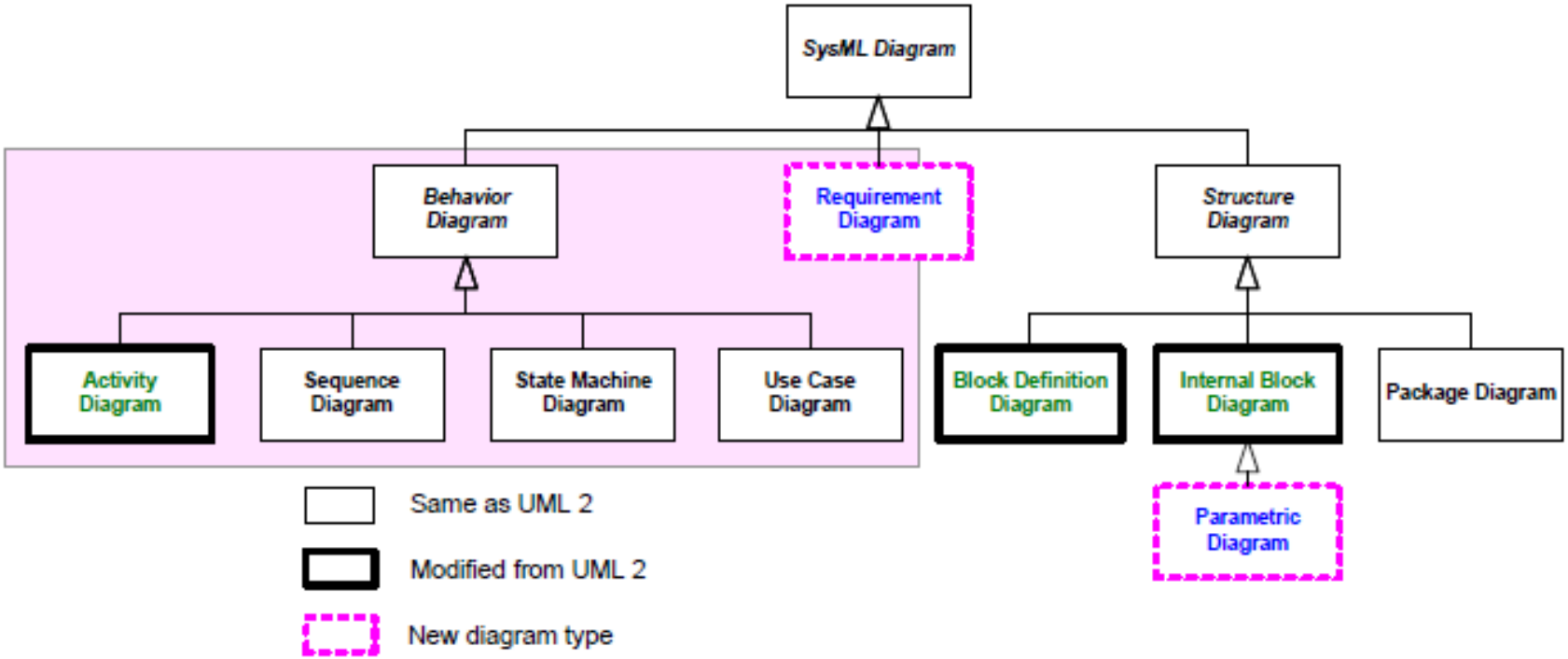
Behavioral Diagrams



SysML: diagram summary

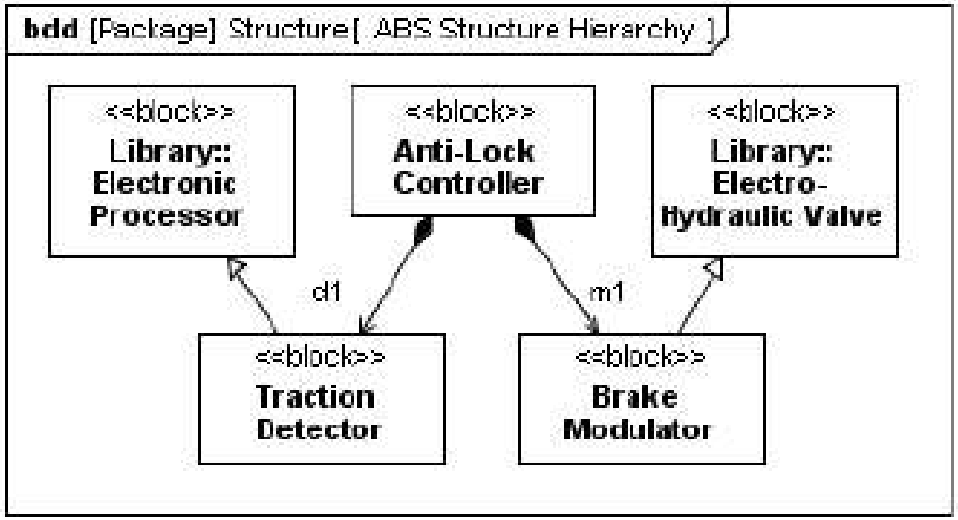
The 9 SysML diagrams

- req – Requirement diagram
- act – Activity diagram
- sd – Sequence diagram
- stm – State Machine diagram
- uc – Use Case diagram
- bdd – Block Definition diagram
- ibd – Internal Block diagram
- par – Parametric diagram
- pkg – Package diagram

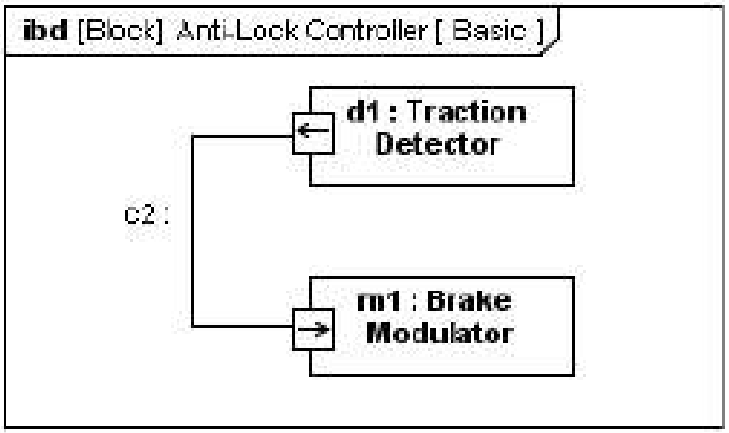


Block Definition vs. Usage

Block Definition Diagram



Internal Block Diagram



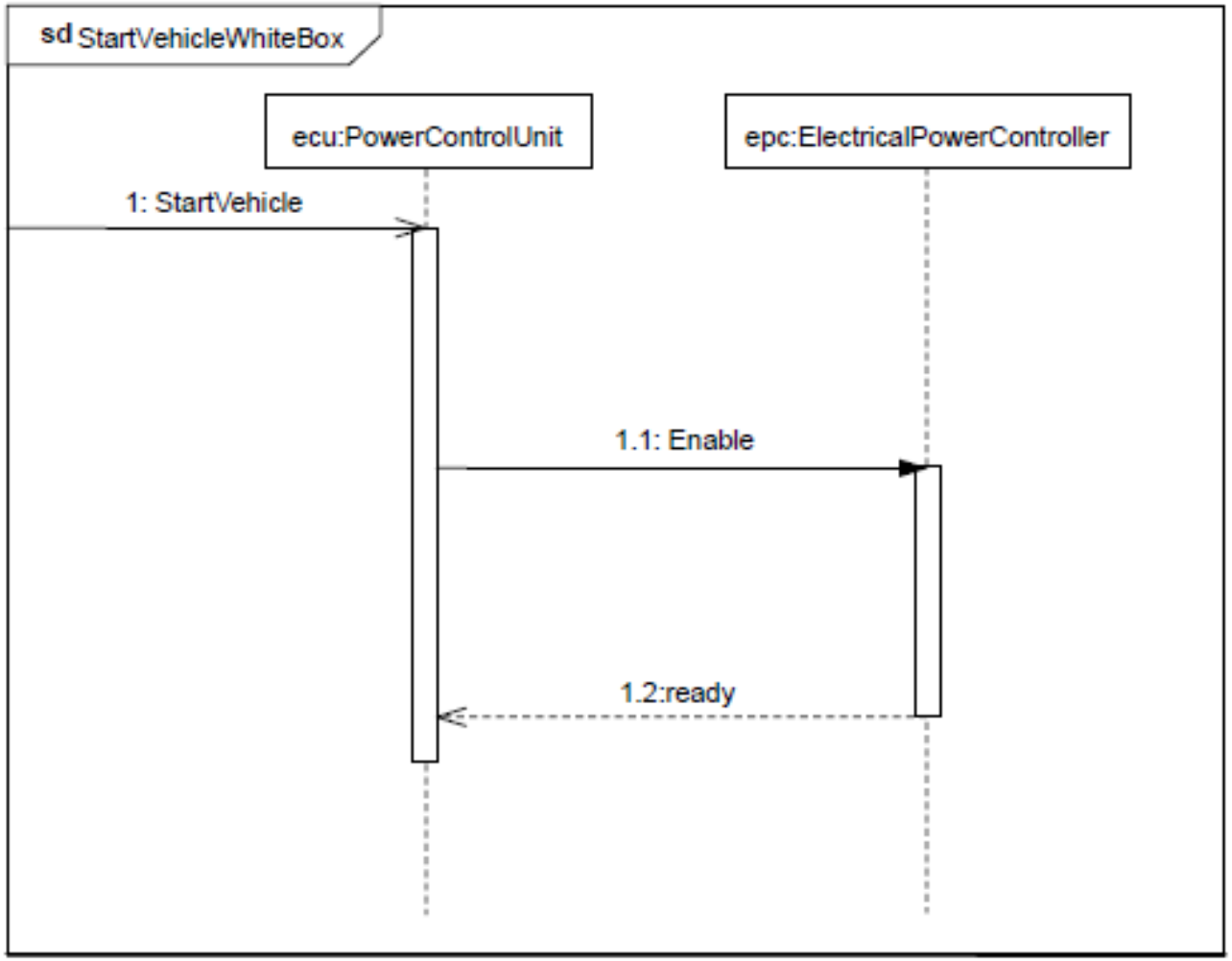
Definition

- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

Usage

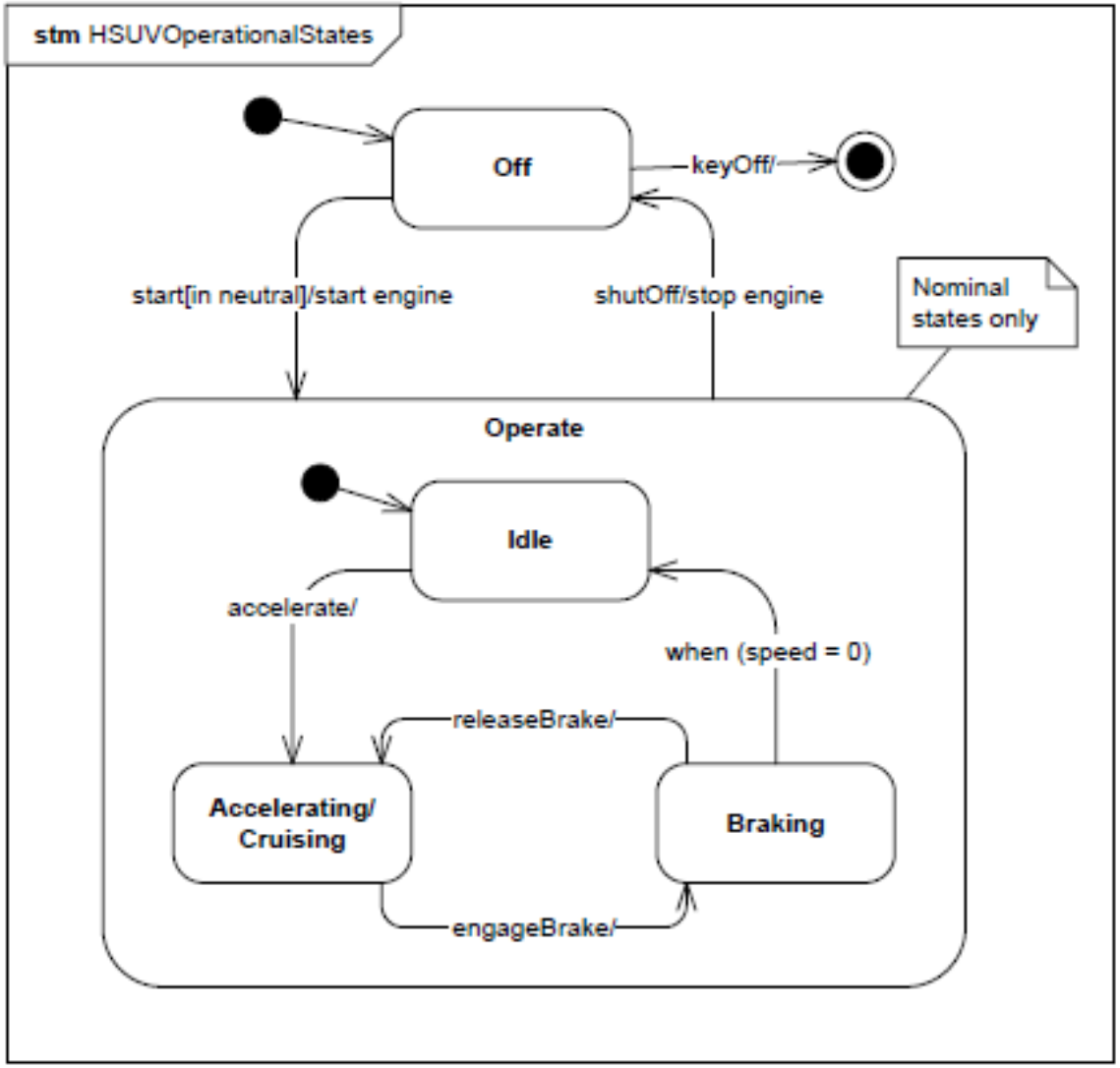
- Part is the usage of a block in the context of a block
- Also known as a role

Sequence diagram



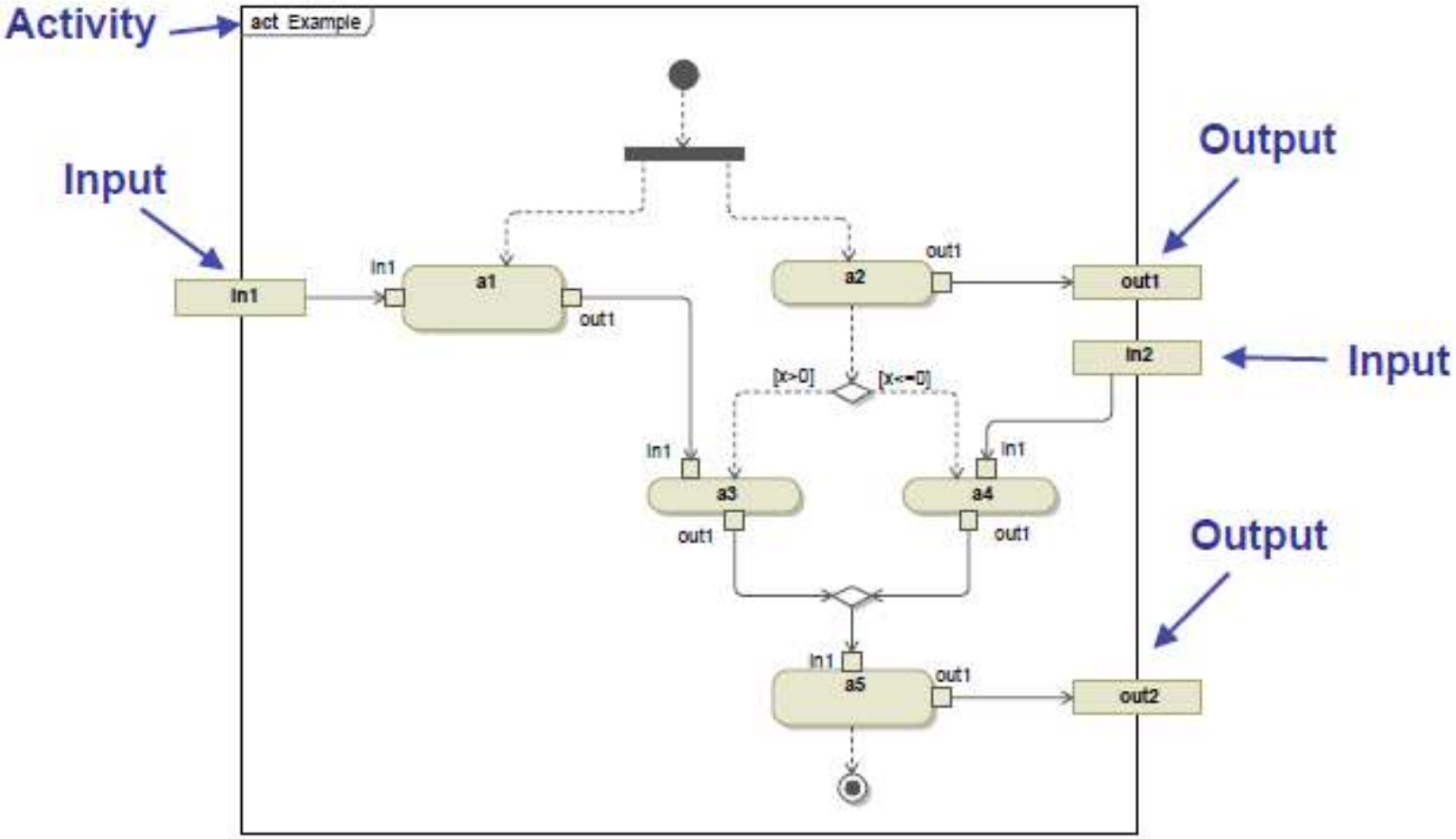
Decomposition of Black Box Into White Box Interaction

State Machine diagram: Operational States (Drive)

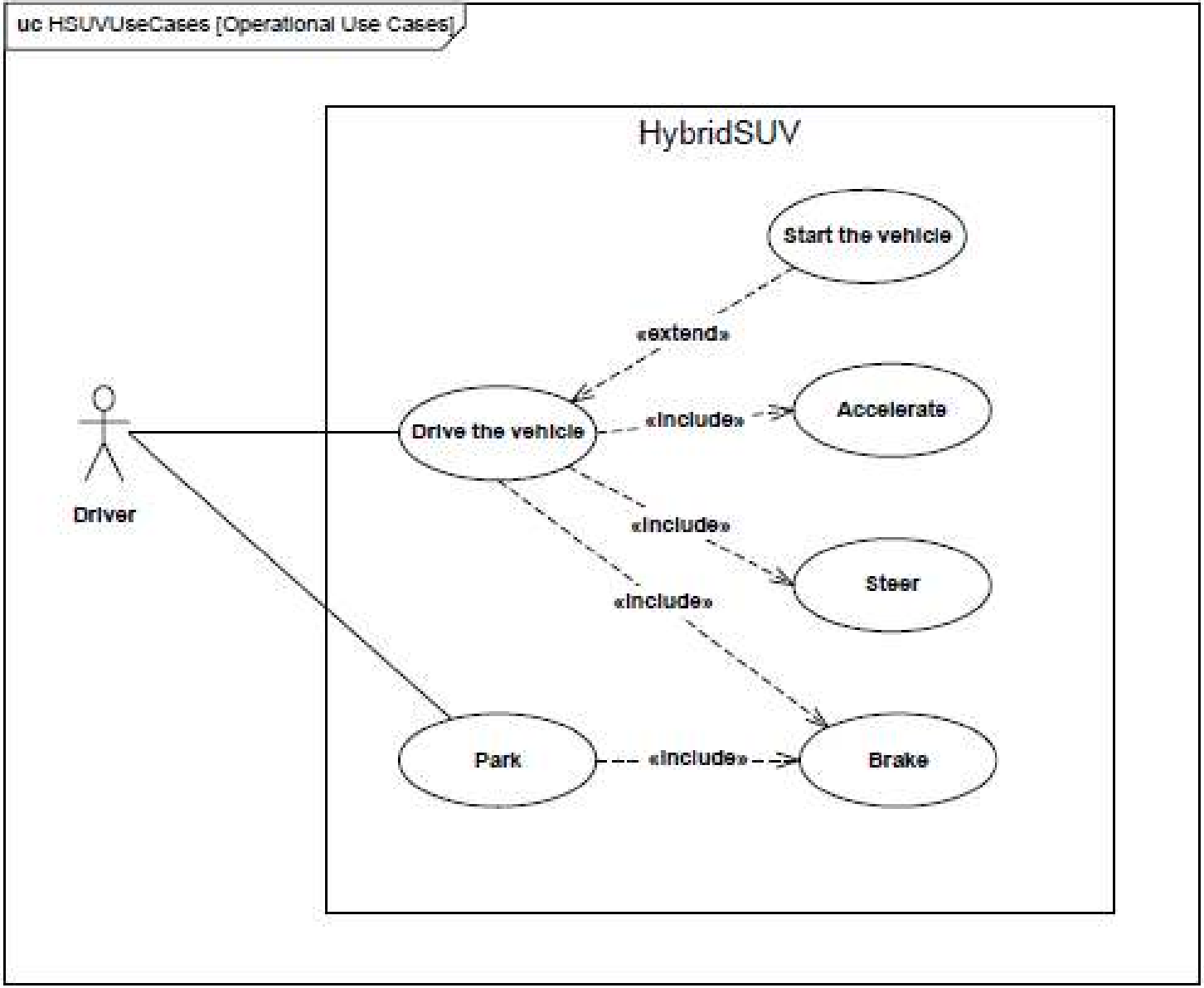


Transition notation:
trigger[guard]/action

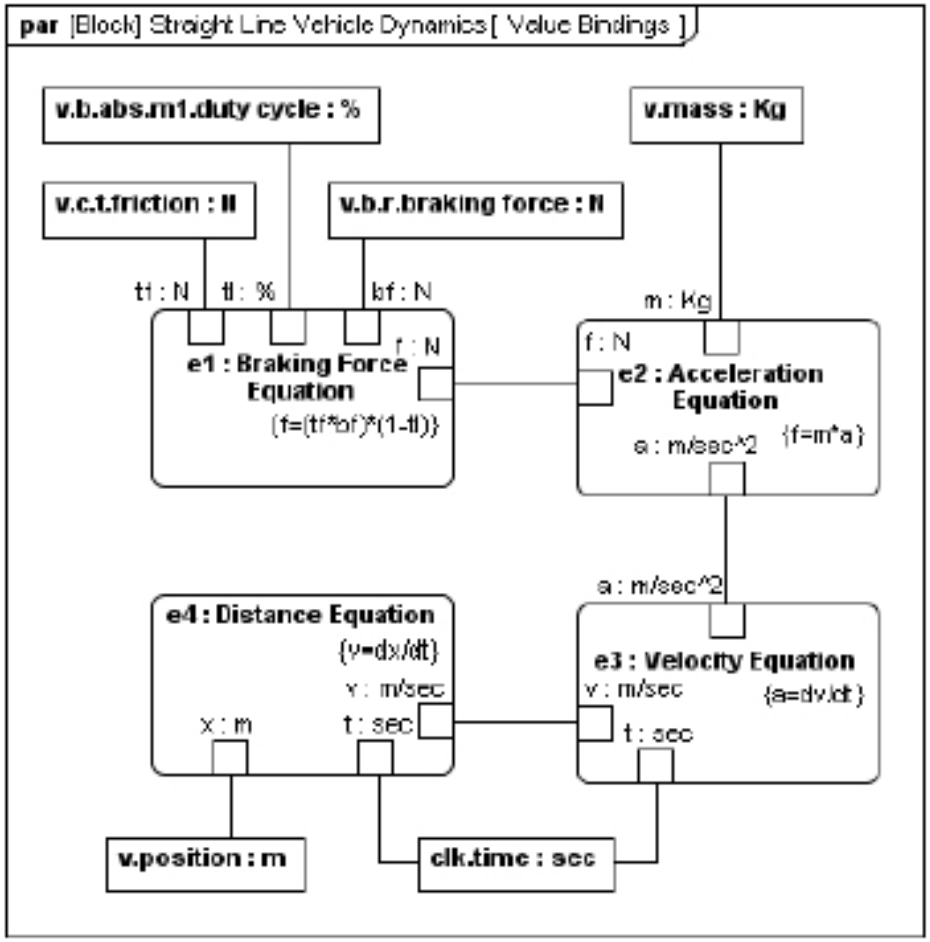
Activity diagram



Use case diagram

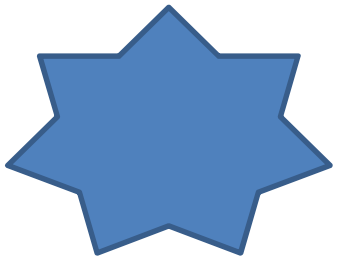


Parametrics



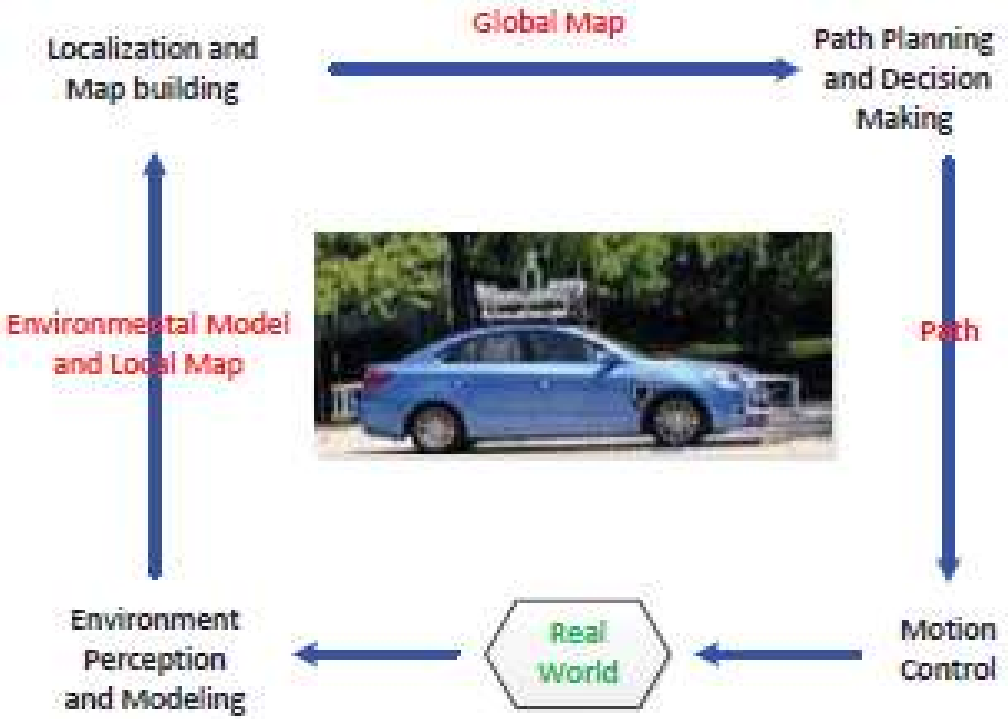
Using the Equations in a Parametric Diagram to Constrain Value Properties

EXTRAS



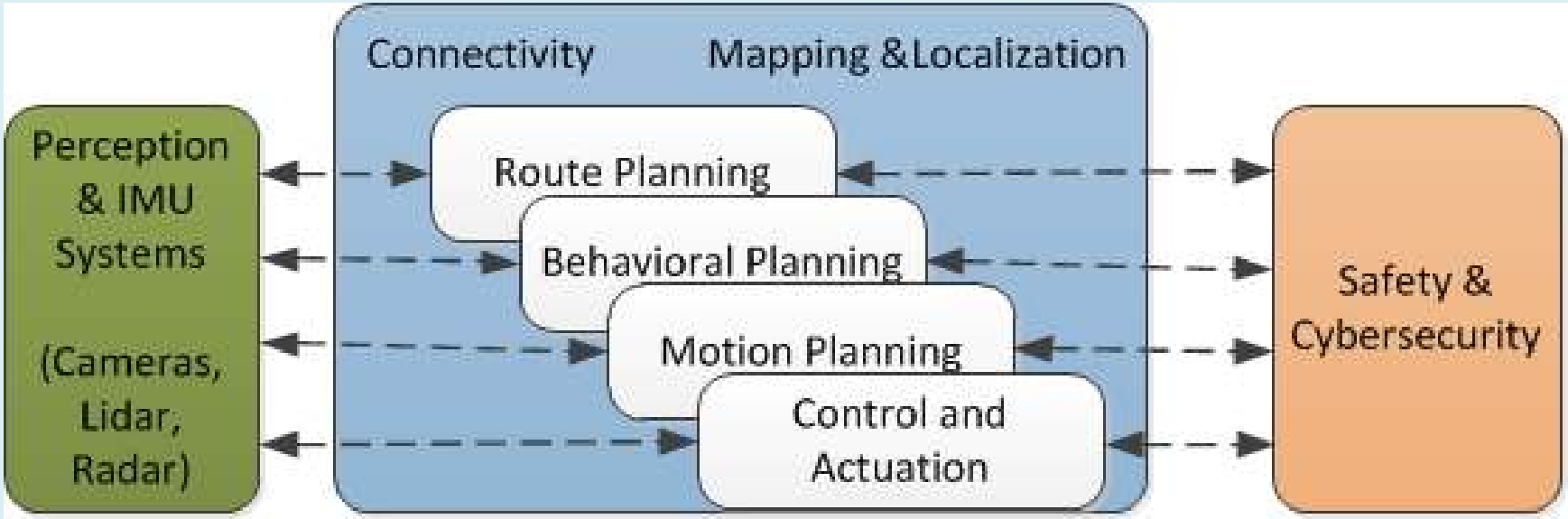
Introduction to Self-Driving Vehicles

- Depiction of the main technologies of autonomous systems
- Perception and modeling
- Localization and map building
- Path planning and decision making
- Motion control
- Many algorithms: e.g., computer vision
- Examples of computer vision tasks:
  - Lane and road detection,
  - Traffic sign recognition
  - Vehicle detection and tracking

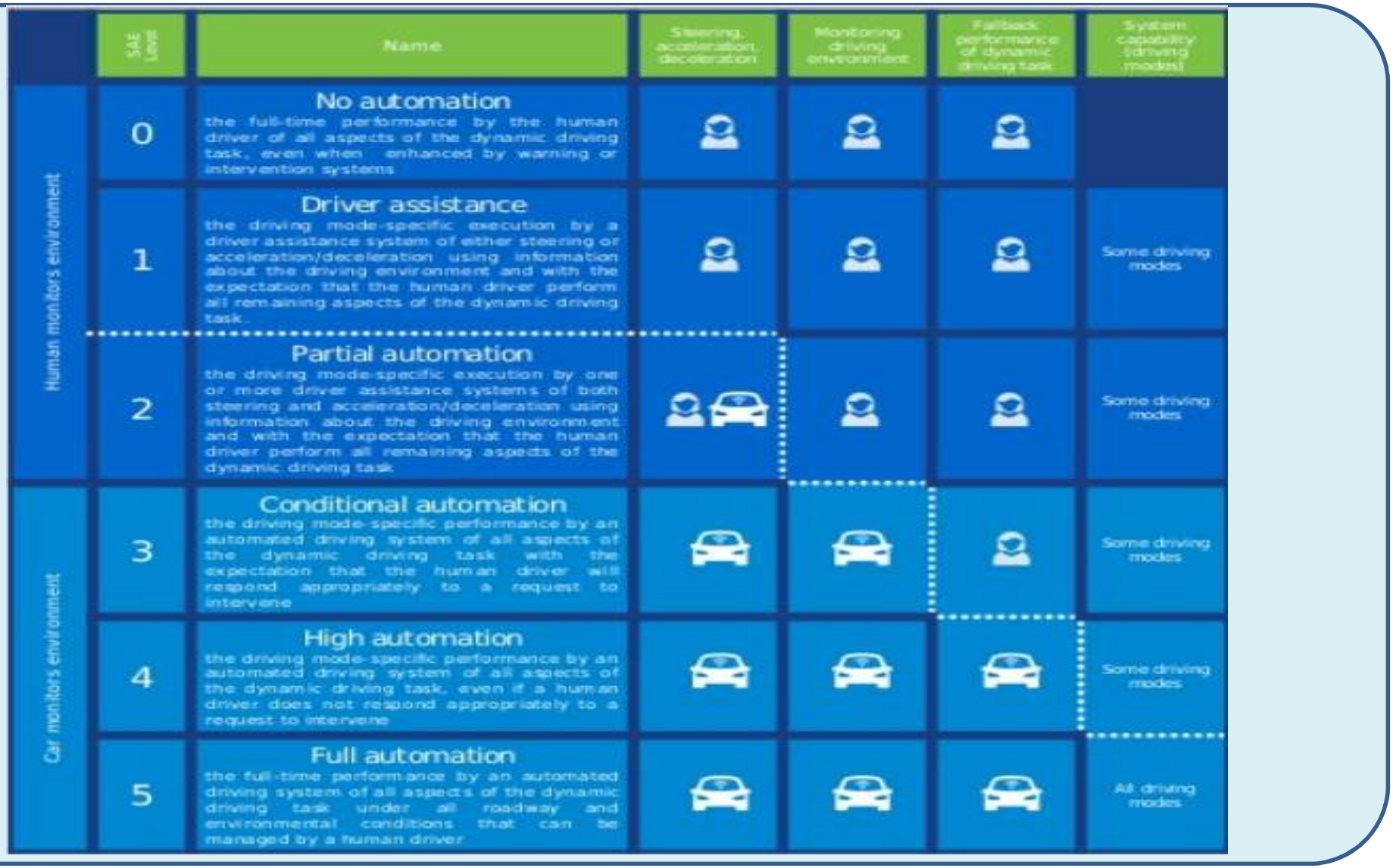


Hierarchical, Modular Sub-systems

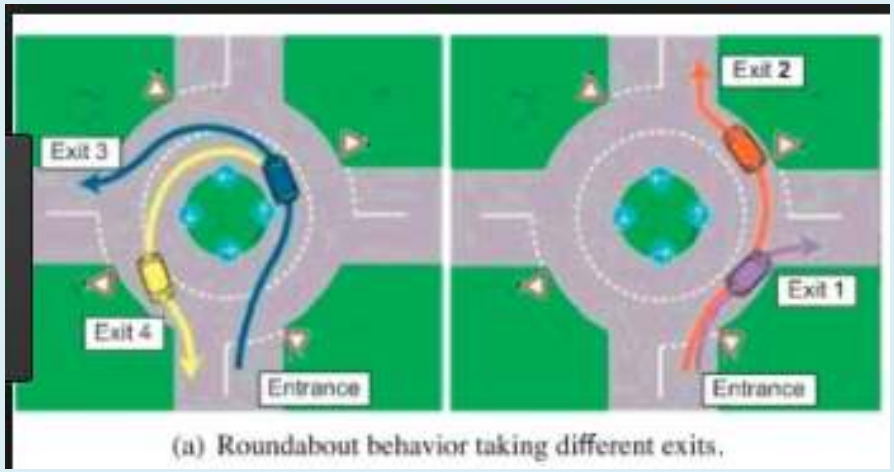
- Connectivity, Mapping, & Localization
- Perception and IMU
- Planning & Control Hierarchy
- Safety & Cybersecurity



Day 1 – Introduction to systems engineering



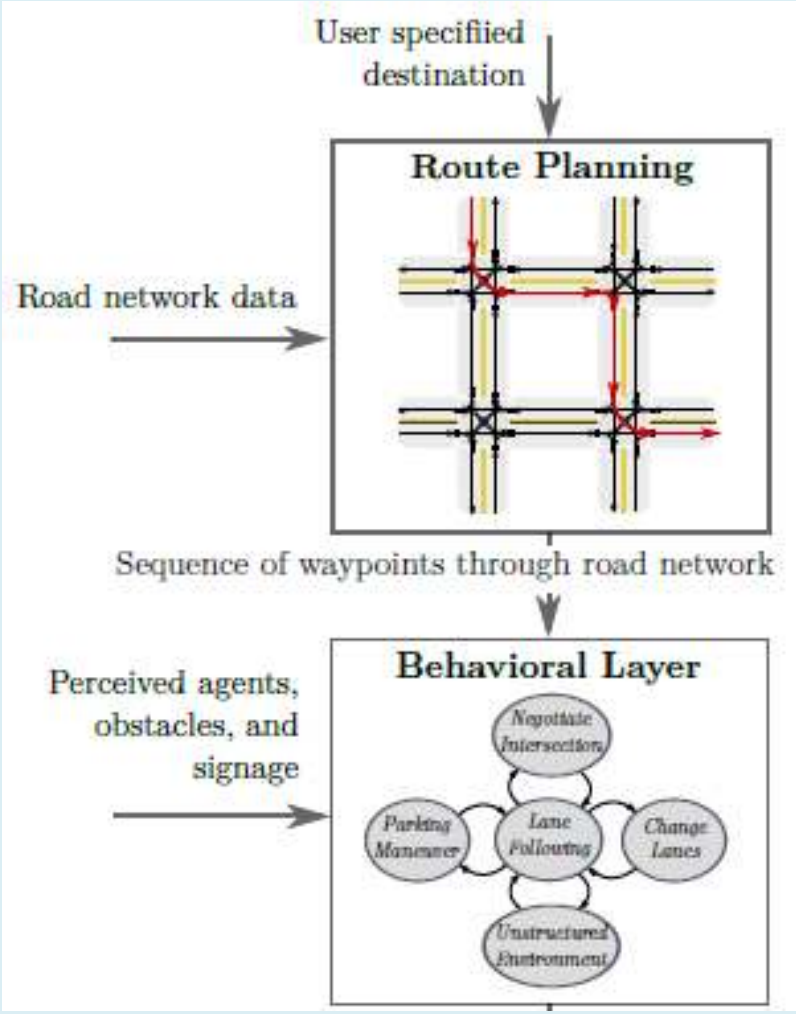
- Planning: How do I get there?
  - Navigation
  - Behavioral
  - Motion



(a) Roundabout behavior taking different exits.

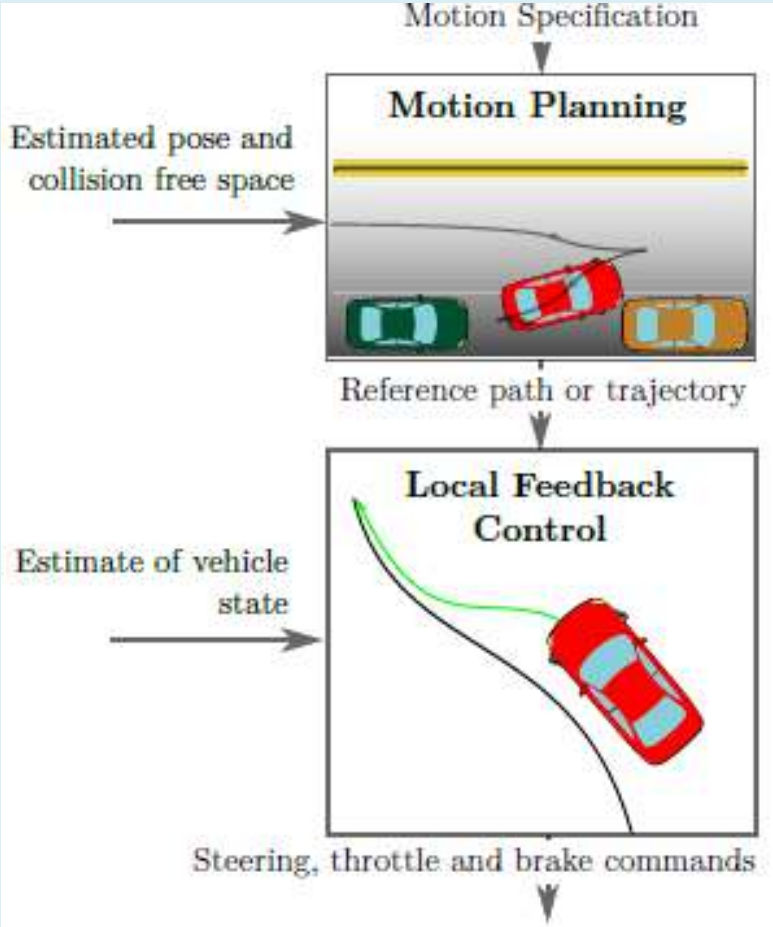
- Control: What do I command to get there?

- Mission planning: Involves several destination or routes.
- Route Planning (Navigation)
  - Behavioral decision making
    - Local motion planning
      - Feedback control
- Destination → route planner
- Output: Route tru the network
- A behavioral layer → motion
- specification (to progress along the selected route and abides by rules of the road.)

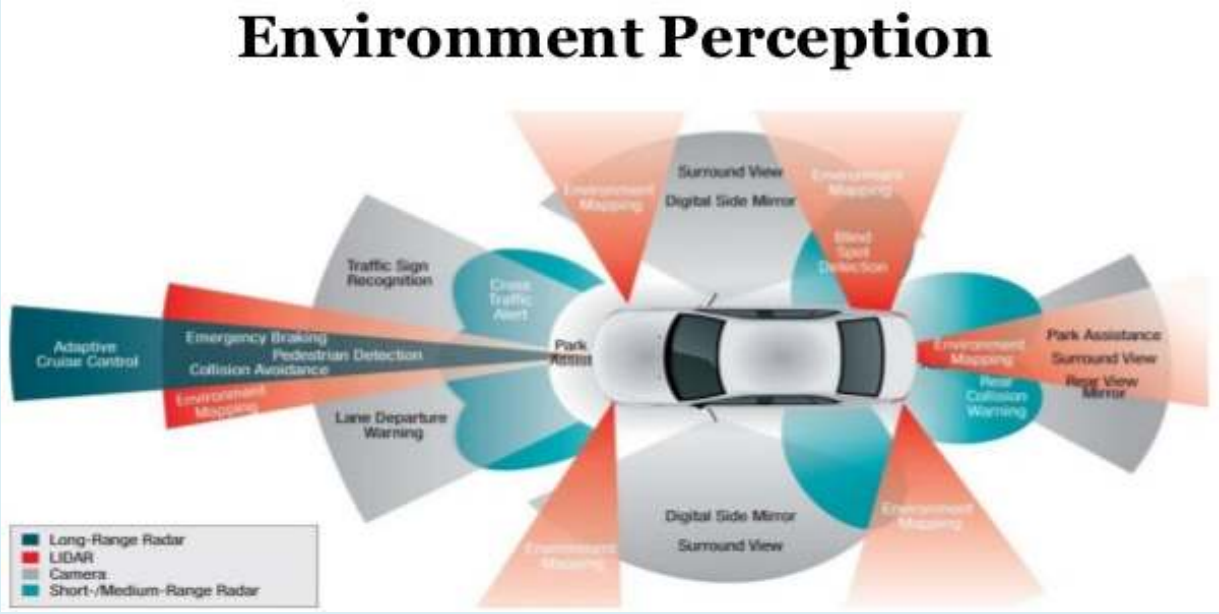


Planning Hierarchy:

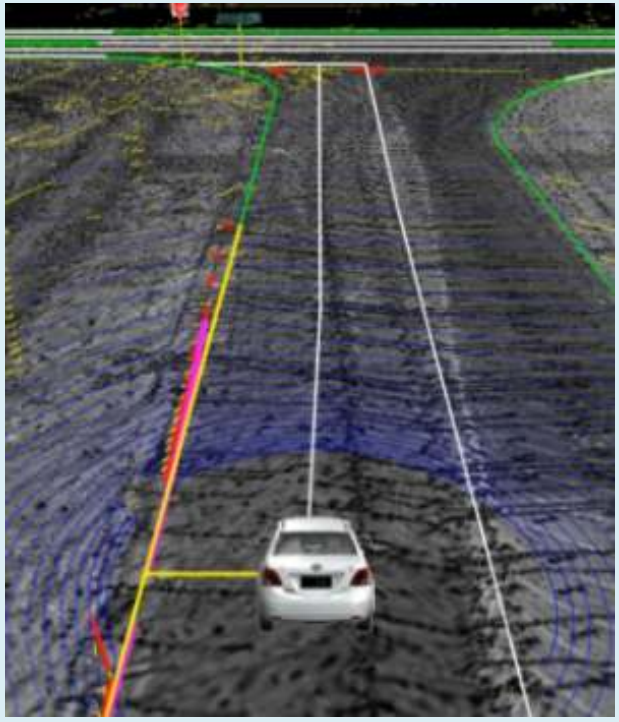
- Motion planner: solves for a feasible motion accomplishing the specification. Outputs: path or trajectory.
- A feedback control system adjusts actuation variables to correct errors in executing the reference path.



- Perception: What do I perceive? (see, hear,…)



- Localization: Where am I?

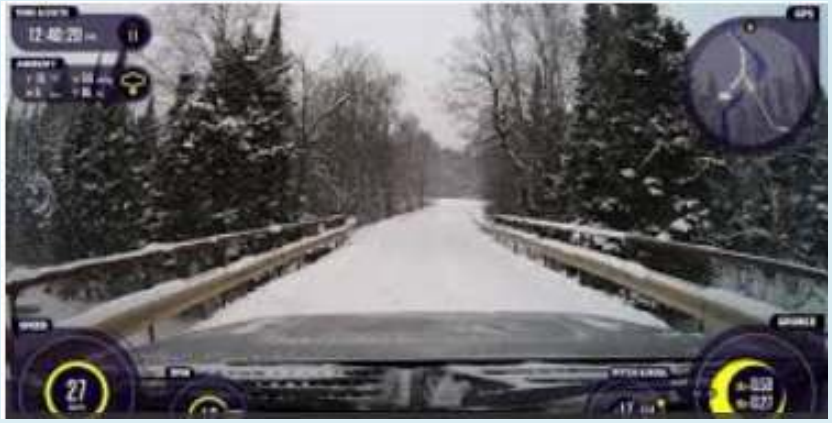


PERCEPTION SENSORS: CAMERAS

- A passive technology, array of detectors that are sensitive to light intensity originating from a field of view (FOV).
- Photo detectors:
  - CCD (charge couple devices)
  - CMOS (complementary metal oxide se miconductors)
- Detectors configured as 2D arrays of pixels (picture elements) where each pixel can detect monochrome (using a gray scale) or multi-chrome light using the basic colors blue, green, and red.

Object Avoidance

- Made of an array of detectors that are sensitive to light originating from a field of view (FOV)
- The array of detectors are made using (charge coupled devices) CCDs or (complementary metal oxide) CMOS technologies
- The detectors are configured as 2D arrays of pixels (picture elements) where each pixel can detect monochrome or multi-chrome light using the basic colors blue, green, and red



PERCEPTION SENSORS: LIDAR

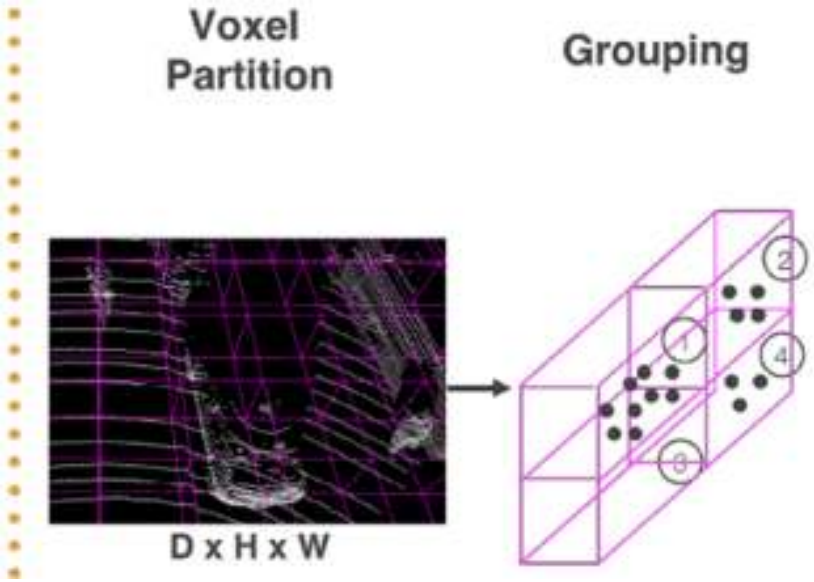
- Active sensor, it emits a laser beam and detects a corresponding beam that is reflected upon hitting an object.
- There are two types of lidar:
  - Flash
  - Scanning
- Flash LIDAR sends out a burst of infrared (IR) light through a laser to a fixed FOV.
- The reflected light from objects in the path is received typically in an array of p-i-n photodiodes.
- The received signal is then analyzed to compute the distance and angular location of the object.

Object Avoidance

- *Mechanical/spinning LIDAR*:
- IR-coherent light is emitted from a laser
- Then collimated and circularized into a beam
- Each beam is matched with a receiver, typically an avalanche photodiode (APD).
- Multiple emitter–detector pairs are mounted on a column that is spun by a motor typically between 10 and 20 Hz
- These systems have good signal to noise ratio and generally provide 360° HFOV.

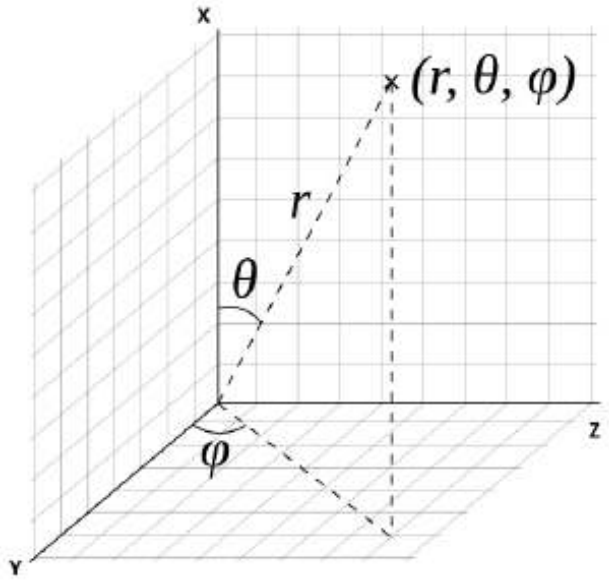
Lidar Point Cloud

- A cloud made of returns of a lidar sensor.
- Returns are modeled as points p_i
- $P_i = [x_i , y_i , z_i , r_i]^T \in R^4$
- (x_i , y_i , z_i) : XYZ coordinates
- (r_i): received reflectance
- Voxel
  - A cube that contains a set of points (similar to a pixel but 3D)



Point Cloud Applications

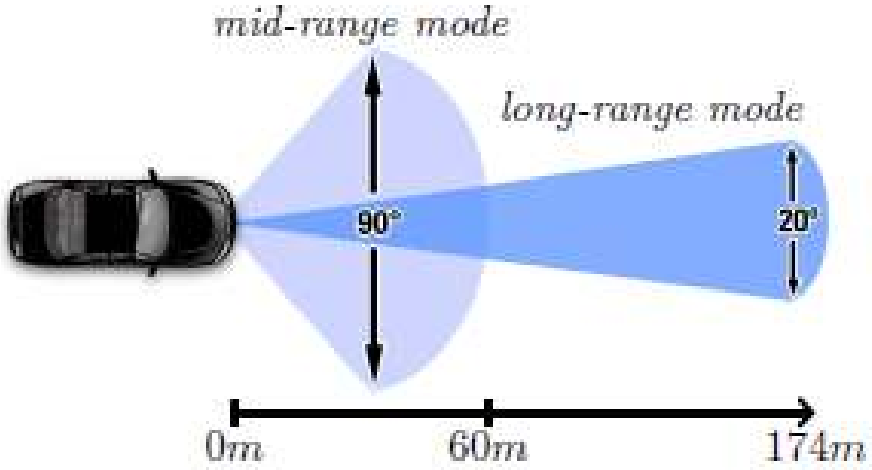
- Object detection
- Sensor fusion
  - With camera
  - With radar
  - With camera and radar



PERCEPTION SENSORS: RADAR

Detects:

- Range
- Bearing (angle)
- Velocity

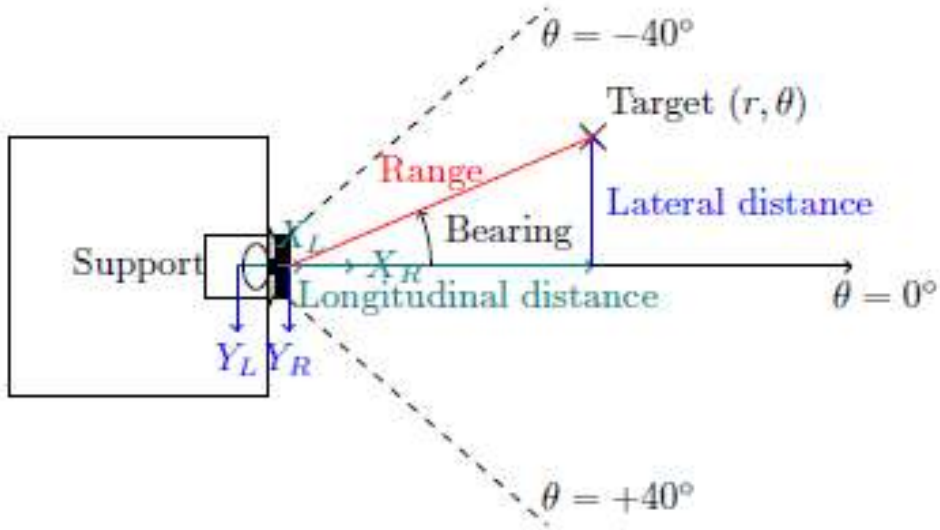


Of a set of targets in the FOV of the Radar.

Mode of operation:

- Short range
- Mid range

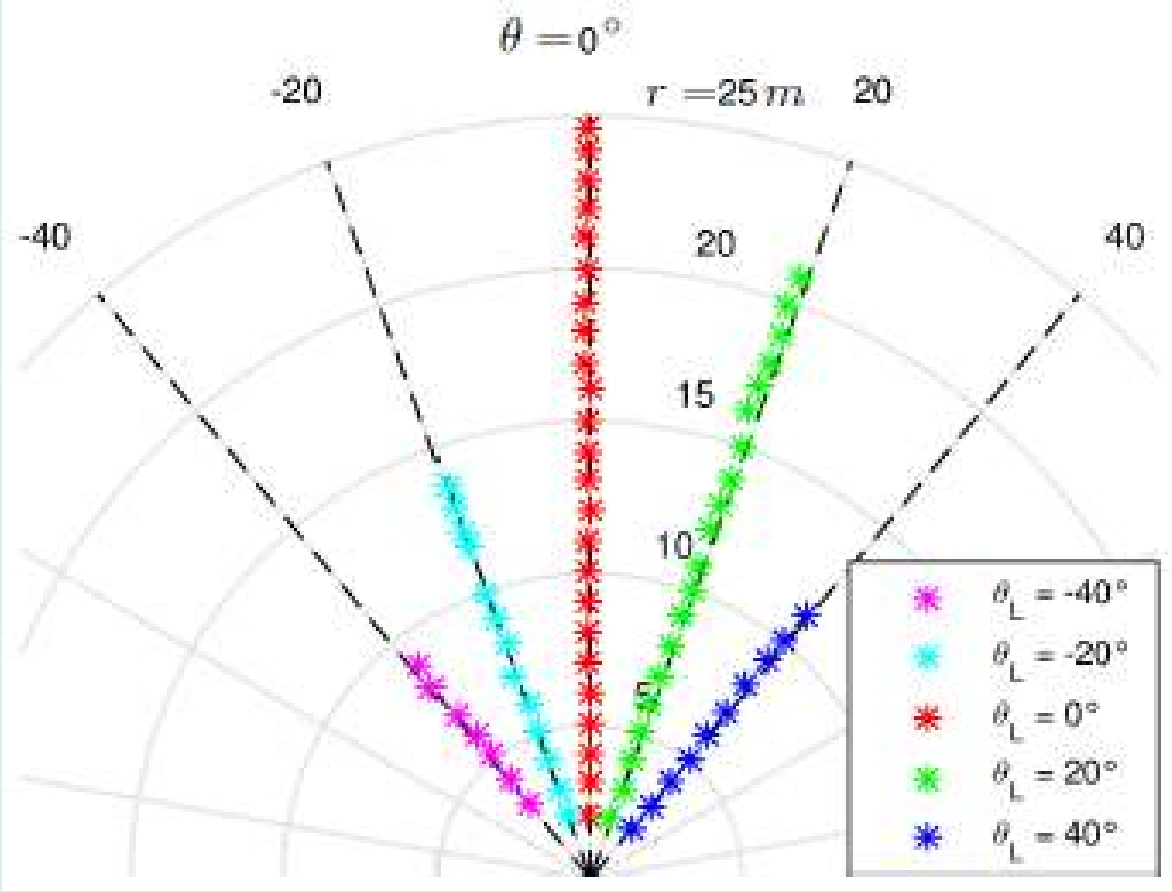
Long range



Radar: FMCW

- An RF transmitter and receiver pair (with own antennas)
- The transmitter sends bursts of sinusoids starting at a frequency f_1 and linearly increasing up to frequency f_2 over a time interval of T seconds.
- If an object is in the FOV of the sensor, it will reflect the signal and it will be detected by the receiver at a frequency f_b called the *beat frequency*

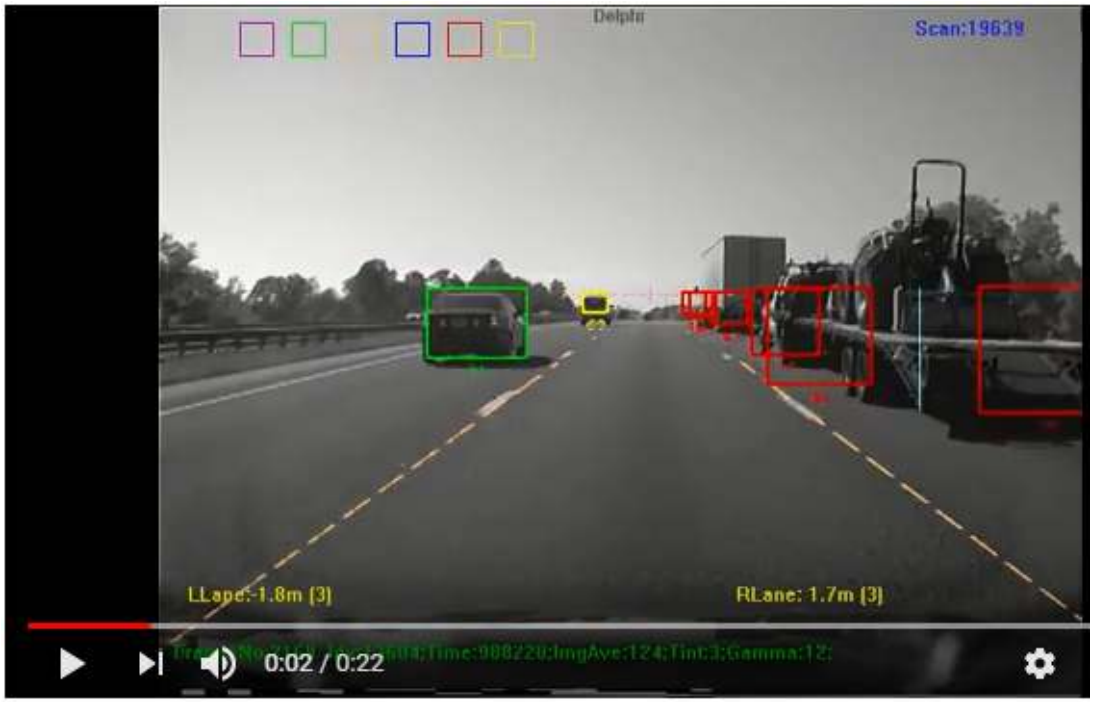
▶ *Radar: Example Test Data*



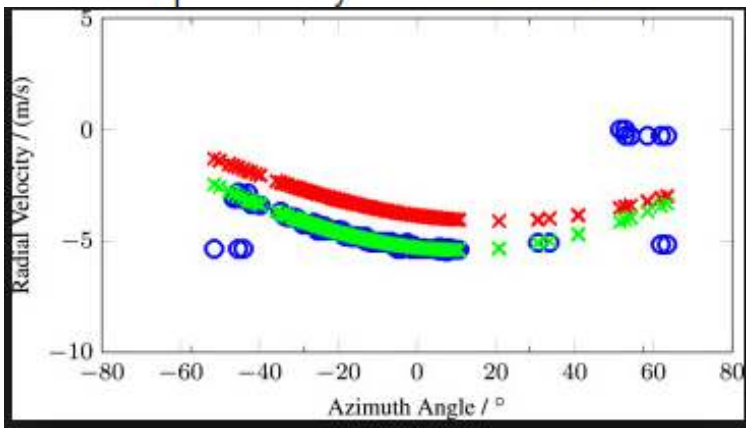
Radar Outputs

- Range (distance)
- Speed (velocity)
- Angle

- Can track multiple objects
- Good a providing ROI (Regions of interest)
- Disadvantages
  - Cannot be used for object identification
  - Resolution is not the best
  - Limited ability to detect static objects
- Needs to be supplemented with other sensors (camera, Lidar)
- Multiple challenges
  - Synchronization
  - Related coordinate systems



Delphi Automotive Radar provided by AutonomouStuff



PERCEPTION SENSORS:
ULTRASOUND

- Active technology using sound waves
- A transmitter sends a pulse
- A receiver receives reflected pulsed that hit objects in the FOV
- Speed of sound = 343 m/s
- At this speed, the applicability of this technology is severily limited.
- Used as a complementary technology for low speed tasks (e.g., parking)
- Usually, a ring of about a dozen sensors are sufficient

Object Avoidance

PERCEPTION SENSORS: PRINCIPLES OF OPERATION

$c = \lambda f$

c: speed of light ($3x10^8$ m/s)

λ: wavelength

f: frequency

- Path Loss:
- Active sensor:
- Passive sensor:
- TOF: Time of flight
- Light detector
- Frequency change (doppler shift)

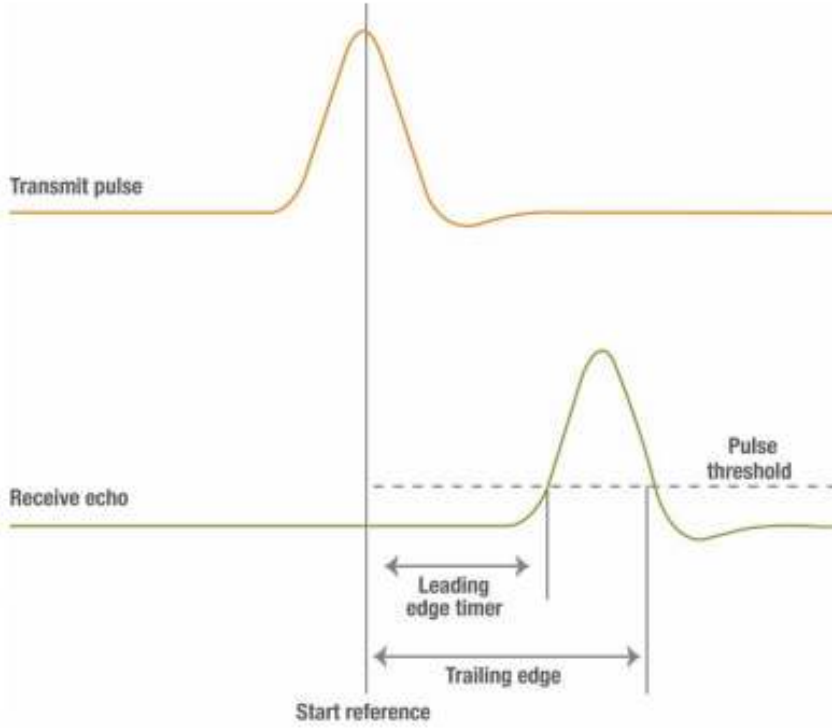
TIME OF FLIGHT

$d = c\ t\ /2$

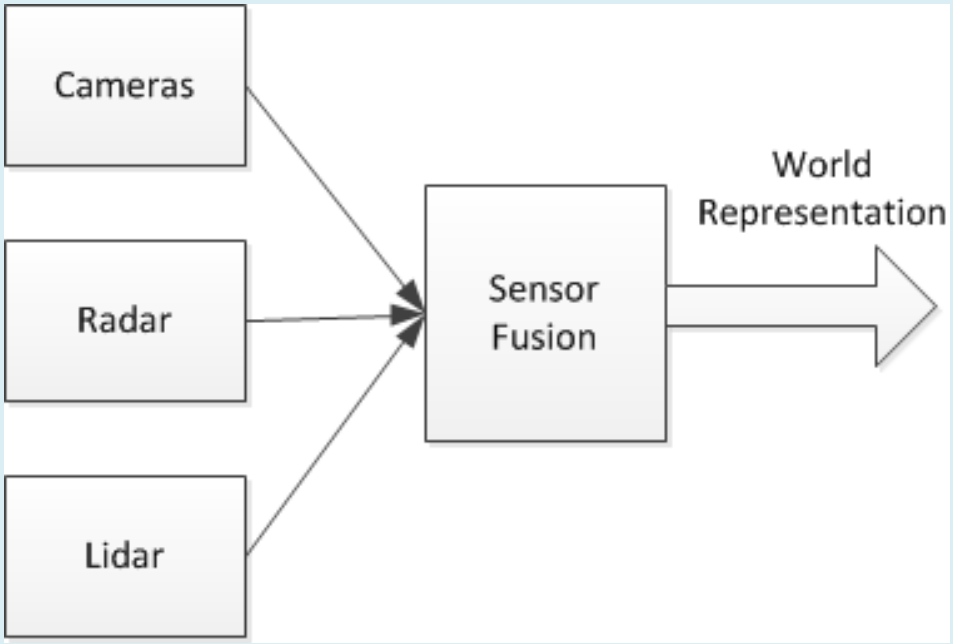
c: 299,792,458 m/s

t: time of flight

d: distance to object (target)



- Sensor Fusion: One method to deal with on board AV sensor limitations.
- Data from many raw sensors are combined to yield more usable information: low level, high level sensor fusion.



Radar & Camera Data Fusion

Region Generated by Radar Data

- This includes position and area
- Will be certified with image data
- STEPS
  - 1. Pre-treatment of radar data (R,θ)
  - 2. Map to image coordinate syst.
  - 3. Find corresponding point O_1 in image.
  - 4. Assume that O_1 is in the center of the target.
- There could be various regions
- Region is defined by their coord.:
  - Lower left: X_l, Y_l, Z_l
  - Lower right: X_r, Y_r, Z_r
- Wm, Hm: Maximum size of vehicle
- α,L: angular and range resolutions

$$\begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} = M_4 M_5 \begin{bmatrix} (R_r - L)\cos(\theta_r + \alpha) \\ (R_r - L)\sin(\theta_r + \alpha) \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} \frac{W_m}{2} \\ \frac{H_m}{2} \\ -L \\ 0 \end{bmatrix}$$

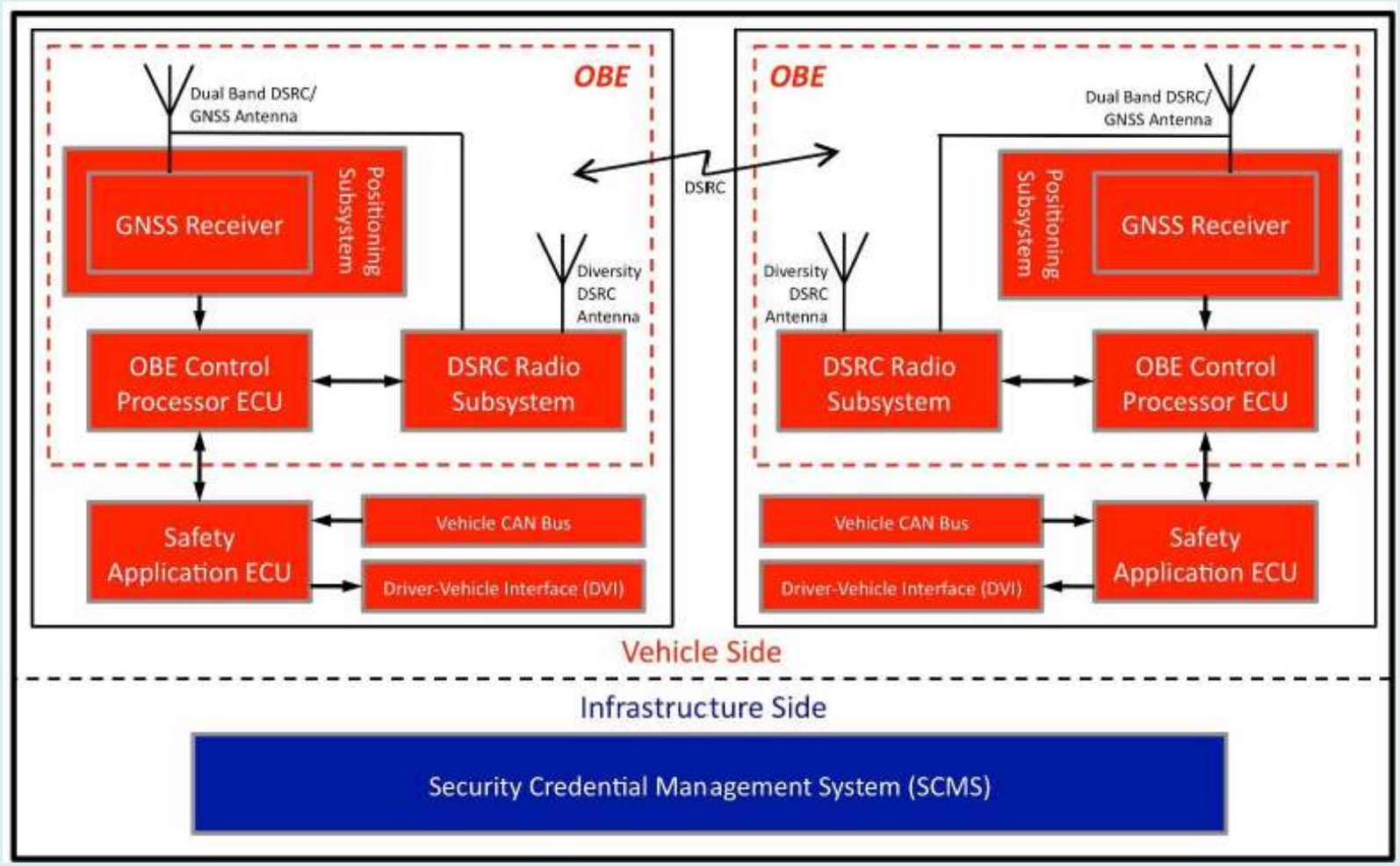
$$\begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix} = M_6 M_7 \begin{bmatrix} (R_r - L)\cos(\theta_r - \alpha) \\ (R_r - L)\sin(\theta_r - \alpha) \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} \frac{W_m}{2} \\ \frac{H_m}{2} \\ -L \\ 0 \end{bmatrix}$$

IMU: INERTIAL MEASUREMENT UNIT

- Supports the localization and planning sub-systems.
- Typically composed of
  - Gyroscopes
  - Accelerometers
  - Magnetometer
- Gyroscopes: angular accelerometers:
  - Yaw, pitch, roll
- Accelerometers: linear accelerometers:
  - x, y, z.
- Magnetometer
  - Enables calculation of the north direction

Object Avoidance

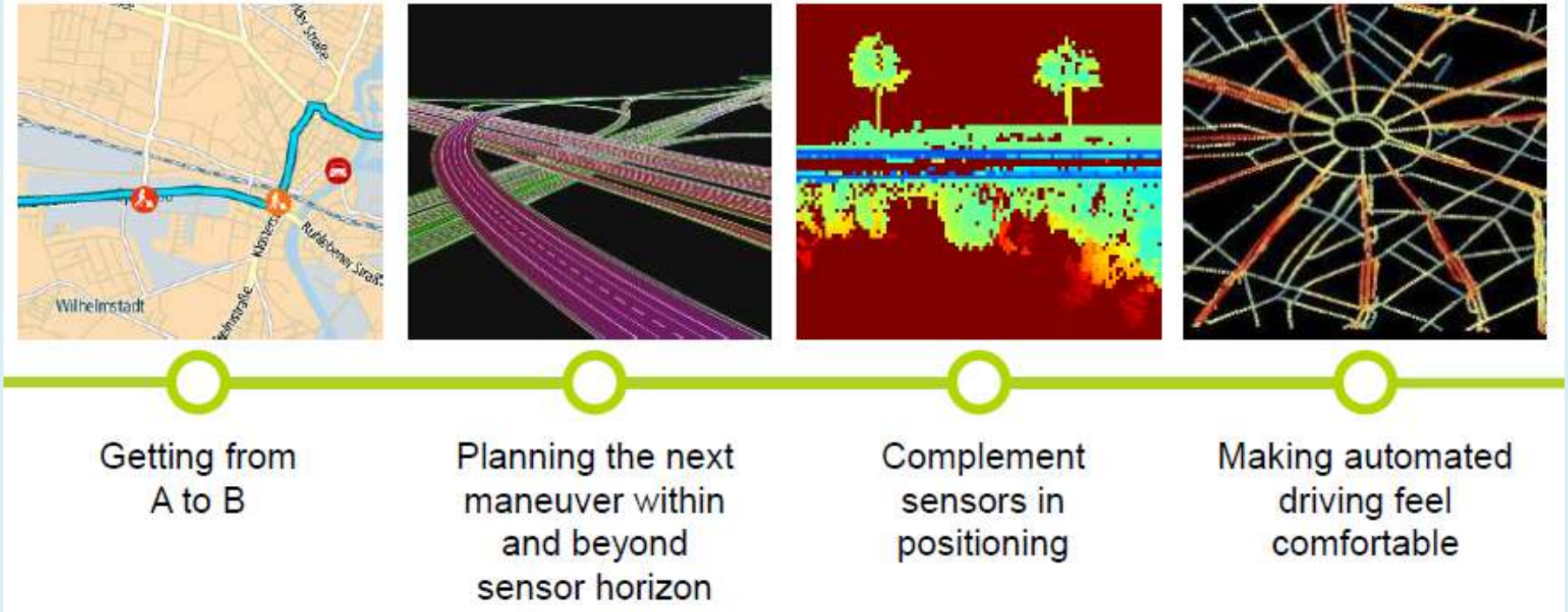
V2X: The SAE 2945 Standard (DSRC): Arquitecture



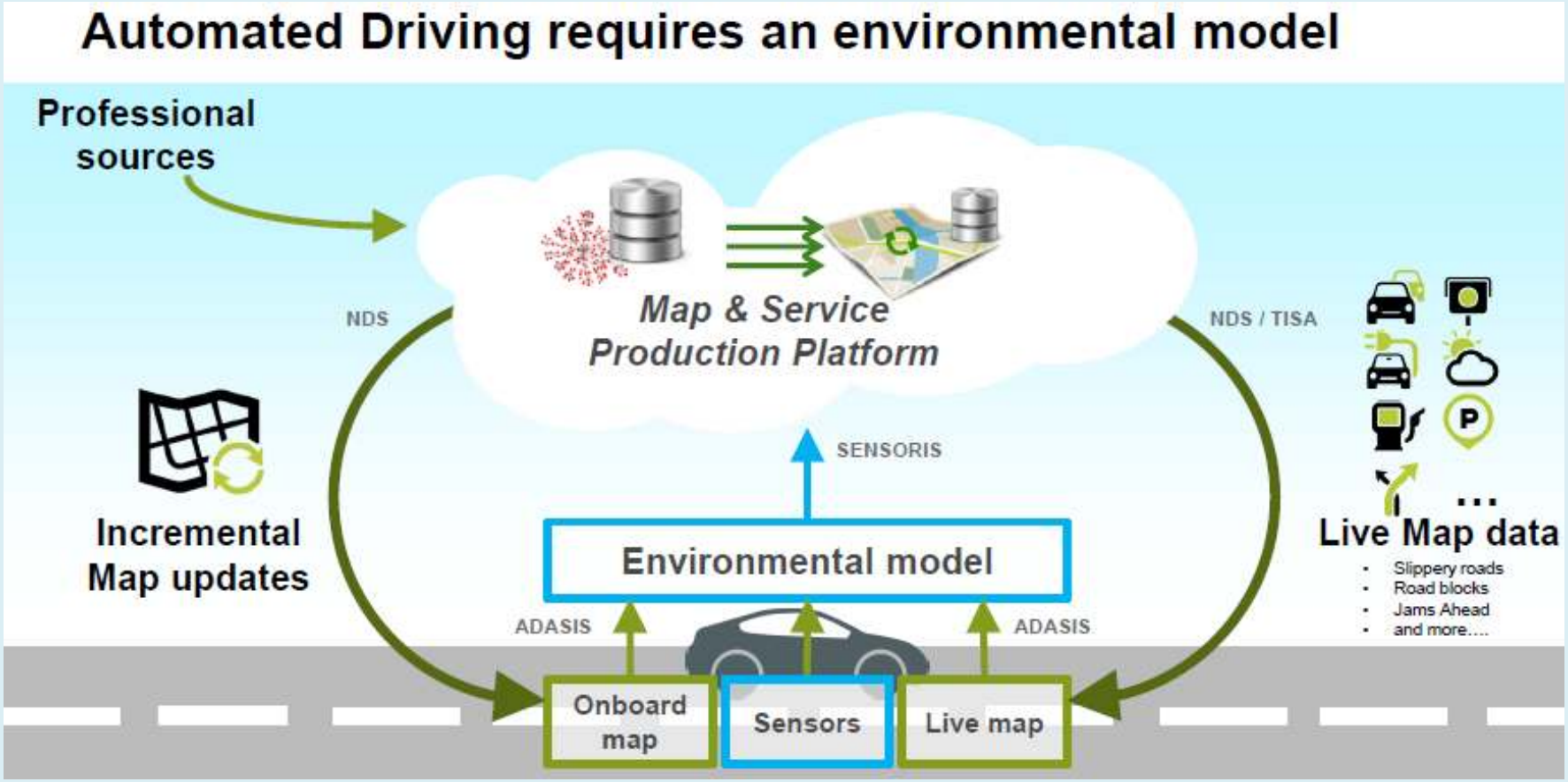
Day 1 – Introduction to systems engineering

- GPS: Global positioning system
- A system that uses up to 32 satellites to provide position information
- It requires direct links with satellites (poor indoor performance)
- Not good accuracy (~ few meters)
- Many company provide embedded chips with GPS receivers & decoders
- By themselves, not sufficient for autonomous driving

The role of a map in automated driving



Source: https://www.adaptive-ip.eu/files/adaptive/content/downloads/moods/Deliverables%20&%20papers/1TomTom_TomWestendorp.pdf



Source: https://www.adaptive-ip.eu/files/adaptive/content/downloads/moods/Deliverables%20&%20papers/1TomTom_TomWestendorp.pdf

Introduction to the Safety of Self-Driving Vehicles

"AV Operational Safety"

- (1) Traditional functional safety as defined by ISO 26262
- (2) SOTIF
  Safety of the intended functionality
- (3) Multi-agent safety
  Driving with others

Introduction

Passive Safety

- Assuming that an accident is effectively inevitable, the aim of passive safety mechanisms is to minimize the severity of that accident.
  - *Examples: S*eatbelts, crumple zones, etc.

Active Safety

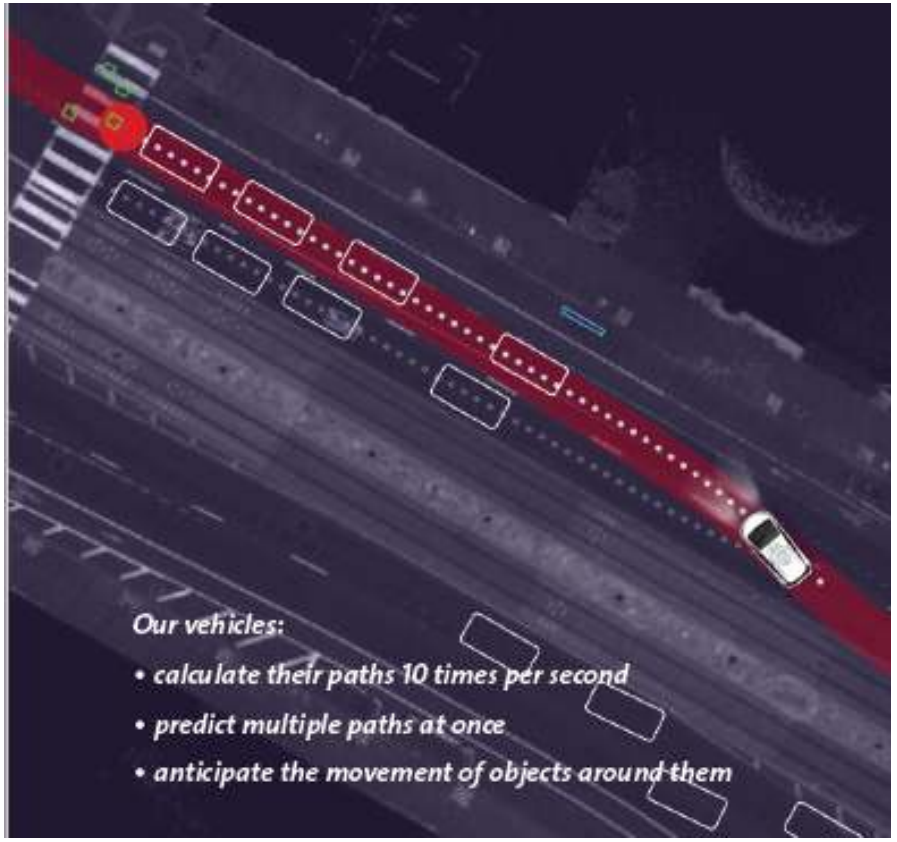
- The systems that are concerned with active safety (based on the knowledge of the current state of the vehicle) -aim to avoid accidents altogether
  - *Examples:* Seatbelt pre-tensioning, airbag deployment, predictive emergency braking, anti-l ock braking systems and traction control

Functional Safety

- This focuses on the correct function of all the electrical and electronic sub-systems
  - *Examples: P*ower supplies, sensors, communication networks, actuators, including active saf ety related systems

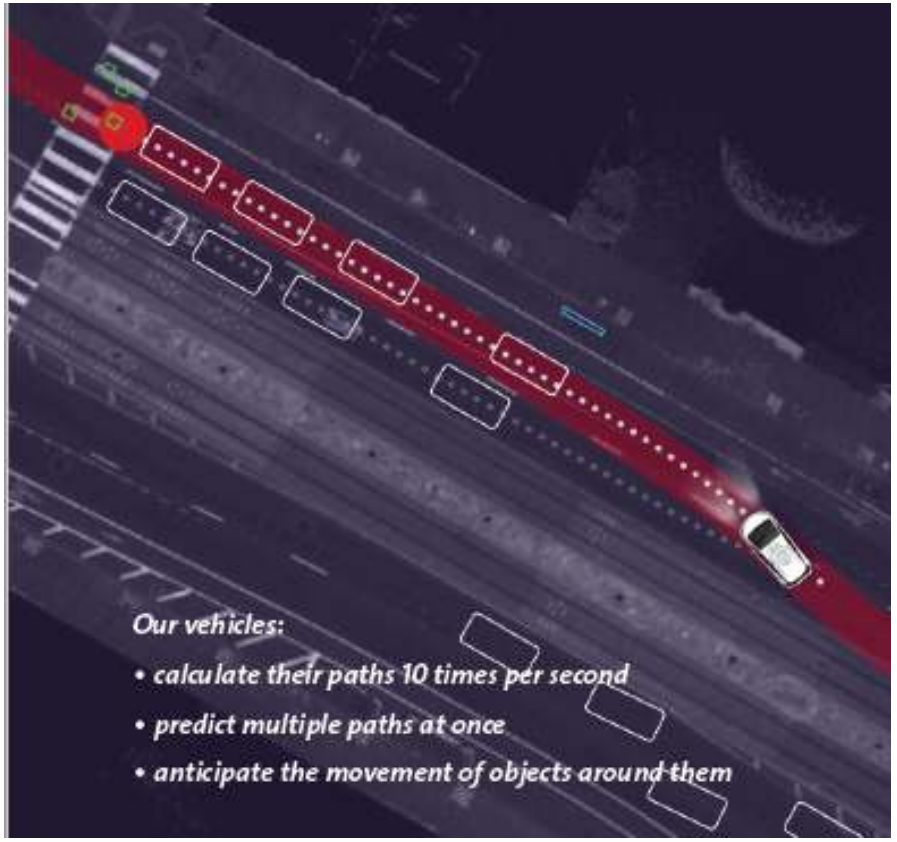
Errors, faults, failures, hazards, accidents, mishap

- Fault
  - abnormal condition that can cause an elem ent or an item to fail
  - Permanent, intermittent, and transient faults are considered.
  - When a subsystem is in an error state it co uld result in a fault for the entire system.
- Errors
  - Discrepancy between a computed, observed or measured value or condition, and the tru e, specified or theoretically correct values
  - An error can arise as a result of unforeseen operating conditions or due to a fault
- Failure
  - Termination of the ability of an element or i tem to perform a function as required



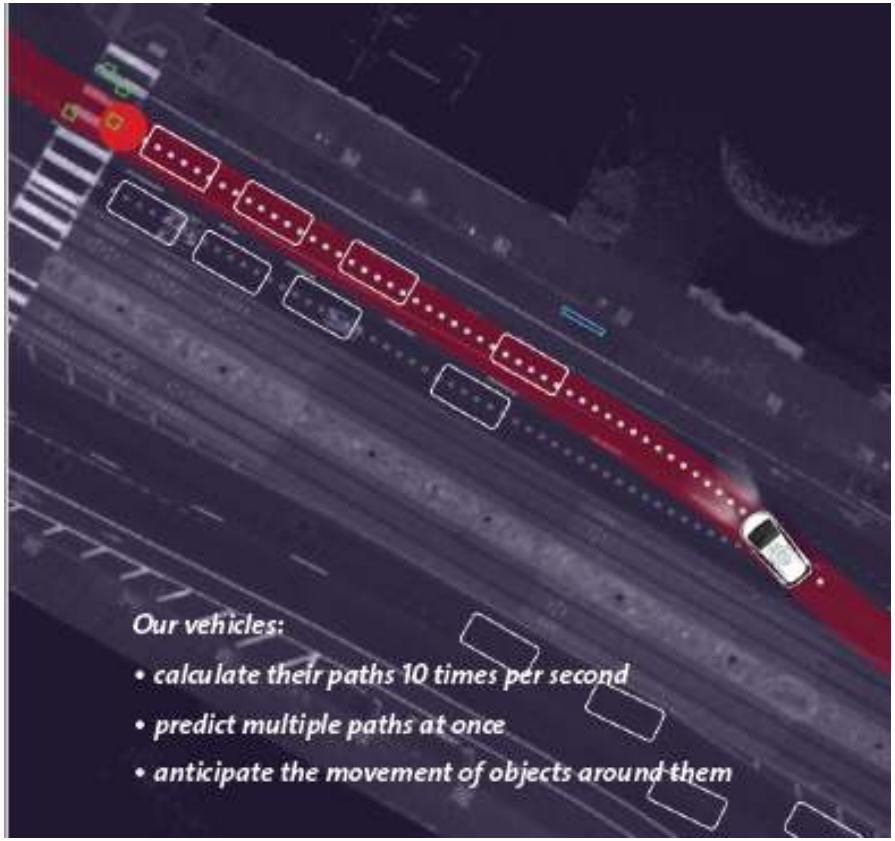
Errors, faults, failures, hazards, accidents, mishap

- hazard
  - potential source of harm caused by malfunc tioning behaviour of the item
- Harm
  - Physical injury or damage to the health of people
- accident
  - An unexpected and undesirable event
- Mishap
  - Unfortunate, unpredictable outcome
- Risk
  - Combination of the probability of occurrenc e of harm and the severity of that harm
- Severity
  - measure of the extent of harm to an individ ual in a specific situation



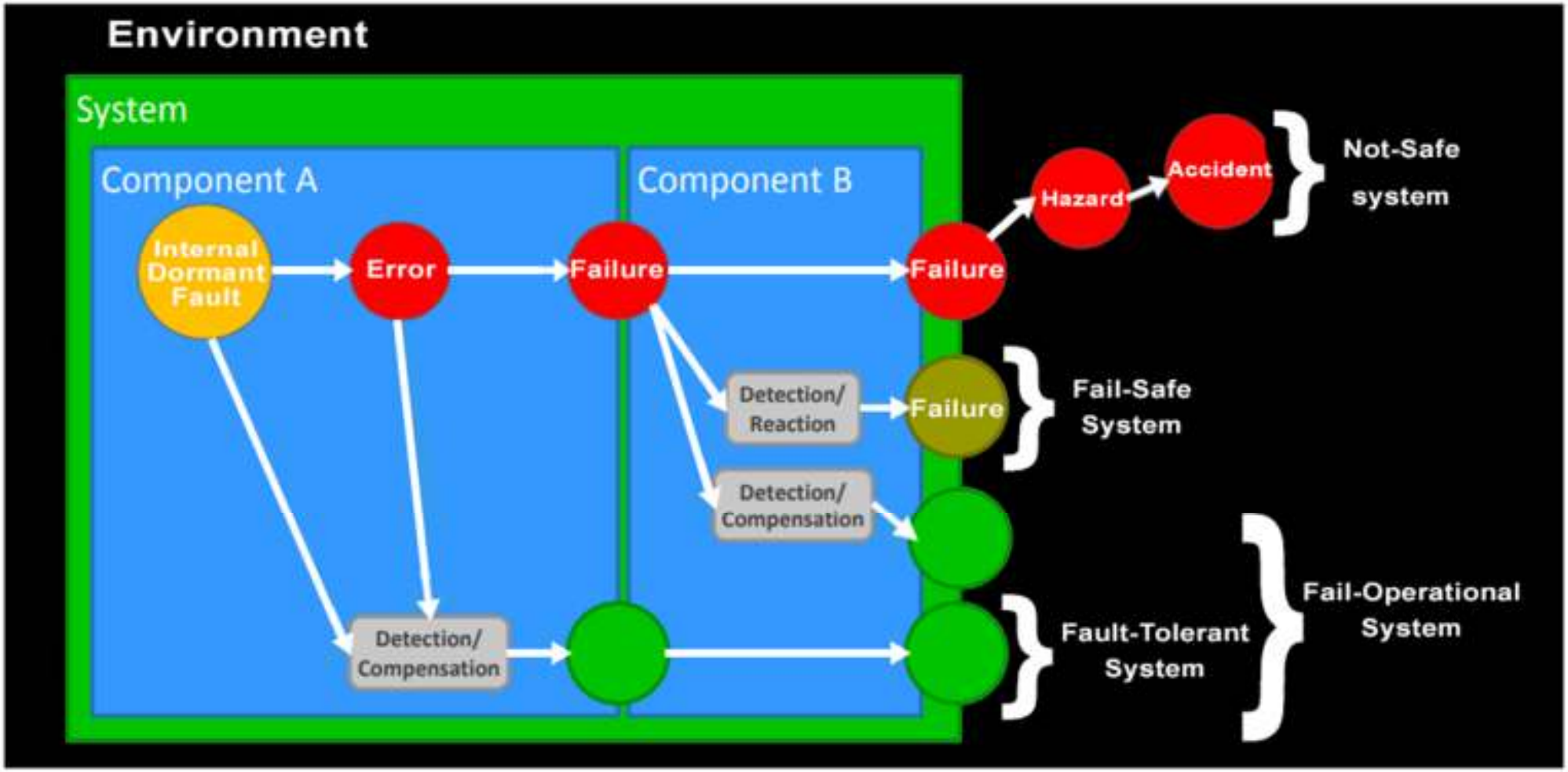
Errors, faults, failures, hazards, accidents, mishap

- Safety
  - Absence of unreasonable risk
- Exposure
  - State of being in an operational situation th at can be hazardous if coincident with the fa ilure mode under analysis
- Controllability
  - avoidance of the specified harm or damage through the timely reactions of the persons i nvolved
- Functional Safety
  - *Absence of unreasonable risk due to hazards caused by malfunctioning behaviour of E/E systems.*



Introduction (cont ...)

- Risk reduction measures and mechanisms.



Safe Autonomous Vehicle platform: Safety-critical components:

Self-Driving Computer

The Cruise AV has two main computer systems operating simultaneously, so if the primary computer has a problem, the secondary system is there to take over.

Signal Communications

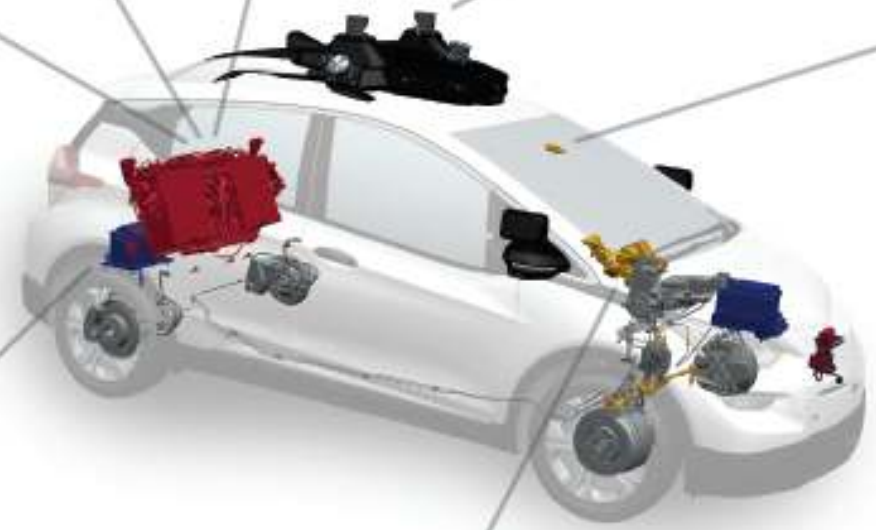
Communications between computers, sensors and actuators have an alternate path if the primary fails.

Perception Sensors

Sensor diversity provides confidence that the self-driving system can detect, track and classify objects around it. Field of view overlaps enable 360-degree vision even if a sensor fails.

Vehicle Localization

The vehicle's location is estimated by many different methods, which means that even if the localization information from one system becomes unavailable, the vehicle can use localization information generated by other sources, such as from LIDAR data or from our inertial tracking system.



Redundant Collision Detection

Our vehicle includes a crash-imminent braking system calibrated to work as a backup to the self-driving system that can apply the brakes to stop the car if necessary.

Electrical Power

We have included redundant electrical power sources and power distribution for all important systems. Main power is provided through the high voltage electric vehicle battery. Should power from that battery fail, backup batteries will power all critical sensors, computers and actuators.

Steering and Braking

On our self-driving vehicles, the steering and braking systems have redundant motor actuators, electrical power and computer electronics so the vehicle can respond safely and keep performing during a failure.

Integrated Vehicle Health Monitor

Keeps track of diagnostics for all self-driving systems in the vehicle and determines operating state of the vehicle.

System Robustness

All critical systems have been designed, tested and validated through intrusive testing, test track durability testing and extensive on-road mileage accumulation.

Safety function

- Any specific functionality specifically designed to mitigate risks.
- **Safety goal**
- Top-level safety requirement as a result of the hazard analysis and risk assessment.
- A safety goal is specific to one or several hazards and associated risks.

Example.

- Let's assume a specific hazard: "*A lidar based pedestrian detection system gives false negatives*"
- The corresponding safety goal can be: "*The autonomous vehicle shall not hit pedestrians.*"

Development of Safety Goals

- Safety Goal: top-level safety requirement as a result of the hazard analysis and risk assessment
- A safety goal is to be determined for each hazardous event evaluated in the hazard analysis
- ASIL determined for the hazardous event is to be assigned to the corresponding safety goal.
- Potential hazard may have more than one safety goal
- If similar safety goals are determined, they can be combined into one safety goal that will be assigned the highest ASIL of the similar goals

Risk reduction

Any design, method, techniques, measure, mechanism or process used to reduce risk to an acceptable level.

Safety measure & mechanisms

Activity or technical solution to avoid or control systematic failures and to detect random hardware failures or control random hardware failures, or mitigate their harmful effects

Note: Safety measures include safety mechanisms.

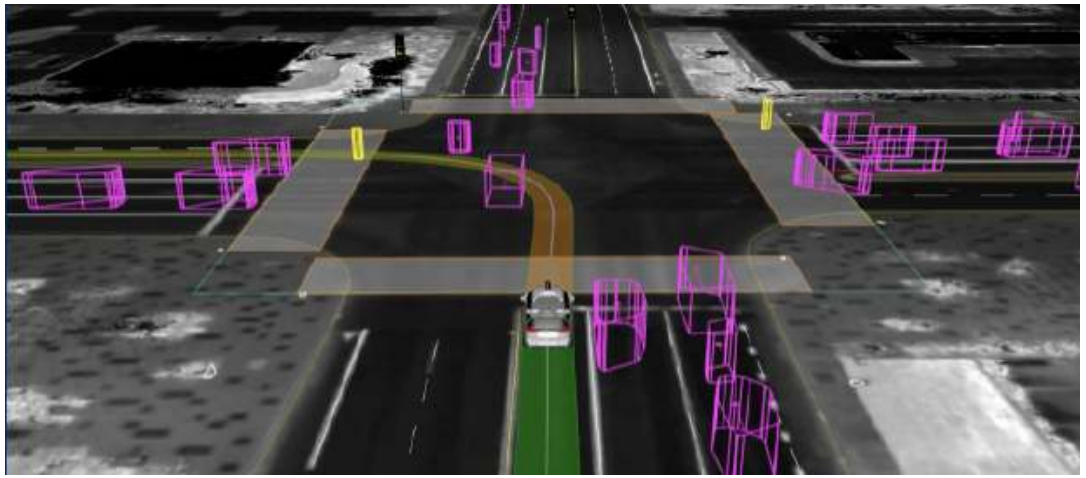
EXAMPLE: FMEA and software without the use of global variables.

Residual risk

Risk remaining after the deployment of safety measures

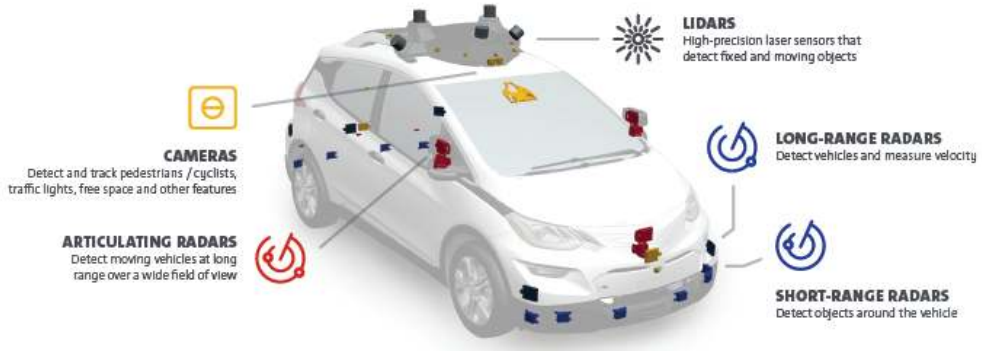
Risk reduction mechanisms

- Adopt a safety culture
- Adhere to appropriate standards
- Use safety architectures
- **Systematic failures**
  - Design and code reviews
  - Use of simulation (e.g., algorithms)
  - Testing
    - Unit test
    - System test
- **Random failures**
  - Fuse sensor data
  - Include redundant HW & SW components
  - Use certified components, particularly comp uting
  - Use of checksums
  - Use of diagnostics
  - Perform plausability checks



Development of the Functional Safety Concept (FSC)

- FSC: Specification of the functional safety requirements with associated information, their allocation to architectural elements, and their interaction necessary to achieve the safety goals
- E.g., (GM).
- *The first key safety performance threshold:*
  - Our vehicle can keep operating properly even if there is a failure in a critical system (fail-operational functionality).
- The second key safety performance threshold:
  - We evaluate the operations of all critical self-driving functions (qualitative & quantitative)

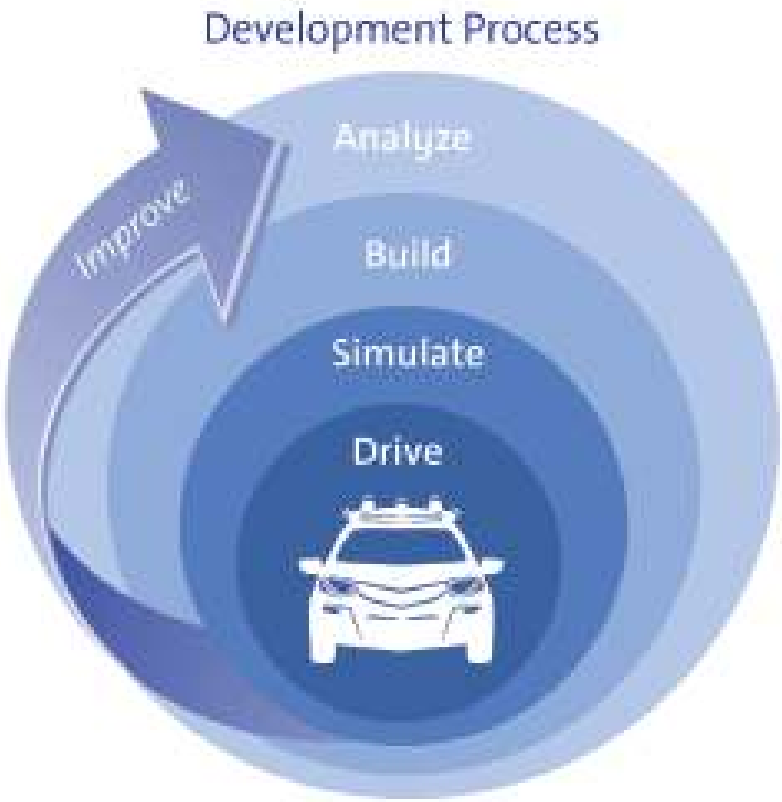


Development of the Functional Safety Concept (FSC)

- ISO 26262 phases: concept, product development, production, operation, service, and decommissioning.
- FSC is the end work item of the concept phase and beginning work item of the product development phase
- Main components of FSC:
  - Top level system design
  - Hazard risk analysis
  - For each hazard:
    - Hazard risk assessment
    - ASIL determination
  - Risk mitigation measures
    - Safety architectures
- The next phase (product development) takes the FSC design specifications to develop system, hardware, and software sub-systems with the required level of risk reduction

Development of the Functional Safety Concept (FSC)

- AV top level design
  - Functionality
  - Perception system
  - Computing platform
  - AV platform
- Failure identification
  - Random
  - Systematic
- AV safety critical subsystems
- List of errors, faults, failures
- AV safety goals
- Preliminary hazard analysis (PHA)
- Assignment of ASILs
- Risk mitigation and risk reduction
  - Systematic failures
  - Random failures



Preliminary hazard analysis (PHA)

- An analysis of all potential hazards including
  - An identifier
  - A description
  - Inputs (causes) of the hazard
  - Resulting state (if known)
  - Worst case harm produced
  - ASIL rationale
  - Associated safety goal

- Example
  - Hazard identifier: PER_028
  - Hazard description: Camera system cannot detect distance to pedestrian.
  - Worst case harm produced by the hazard: AV hits pedestrian.
  - ASIL rationale: Very high
  - Associated safety goal: AV shall not hit pedestrians.

- NOTE: A safety goal ~ Safety Instrumented Function

Introduction to Multi-agent safety: Sharing the road with others

- AV safety taking into accounts multiple agents "multi-agent safety" or "behavioral safety"
- In the context of multi-agent safety, can we **guarantee safety**? If so, under what conditions?
- Ultimate goal: guarantee zero accidents, is this possible?
- Impossible since multiple agents are typically involved in an accident and one can envision situations where an accident occurs solely due to the blame of other agents.
- Model based approach is important that complements "Functional Safety" approach which is probabilistic based.
- Model "Responsibility Sensitive Safety" (RSS)
  - Formalizes the notion of "accident blame"
  - Interpretable and explainable
  - Incorporates a sense of "responsibility" into the actions of a robotic agent.

Multi-agent safety: Sharing the road with others

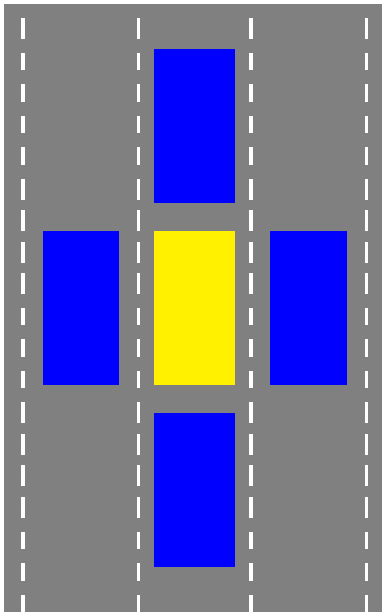
- Agents play a **non-symmetrical role** in an accident where typically only one of the agents is responsible for the accident and therefore is to be blamed for it.
- ***Cautious driving***: under limited sensing conditions, not all agents are always visible (due to occlusions of kids behind a parking vehicle, for example).

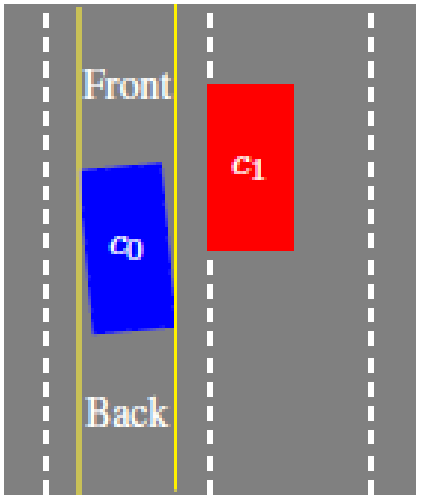
Is it possible to achieve absolute safety?

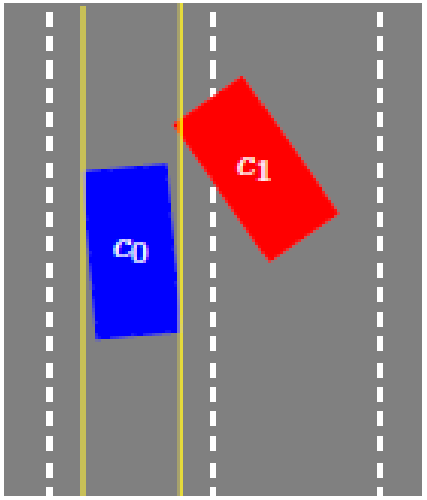
- An action taken by a car c is absolutely safe if no accident can follow it at some future time.
- It is impossible to achieve absolute safety, by observing simple driving scenarios.
- What kind of actions or behaviors (e.g., staying in its own lane) can be considered safe?

Vehicle Dynamics Considerations

- Vehicle safety in a multi-agent environment has to take into account the dynamics of:
  - Vehicles (ego and other)
  - Pedestrians
  - Other objects







Multi-agent Safety: General Considerations

- Safety in the context of behavioral planning
- Complete avoidance of every accident scenario is impossible
- There is a lack of safety guarantees: how do we provide them?
- Can we define "cautious driving"?
- We need a formalized model for fault
- We need safety test & validation processes
  - Data-intensive?
  - Model based?
  - Fault Injection?
- Is it appropriate to assign "responsibility" or "accountability" for accidents?
- Related terms:
  - Fault
  - Blame
  - Guilt

Accidents: Fault, Blame, Guilt

- **Fault:** suggests a failure or deficiency on the part of the responsible party.
- Example: It's my own fault that I wasn't prepared for the exam.
- **Blame:** stresses the assignment of accountability and often connotes censure or criticism.
- Example: The police laid the blame for the accident on the driver.
- **Guilt:** applies to willful wrongdoing and stresses moral or legal transgression.
- Example: The prosecution had evidence of the defendant's guilt.

Responsibility Sensitive Safety (RSS)

- Normalizes the notion of "accident blame"
- It is interpretable and explainable, and incorporates a sense of "responsibility" into the actions of a robotic agent.
- Agents play a non-symmetrical role in an accident
- Typically only one of the agents is responsible for the accident and therefore is to be blamed for it.
- The RSS model includes "cautious driving" under limited sensing conditions where not all agents are always visible (due to occlusions of kids behind a parking vehicle, for example).
- Is it possible to construct a set of local constraints on the short-term that guarantees Safety in the long term?

Responsibility Sensitive Safety (RSS): Definitions

- **Blame Time:**
- The Blame Time of an accident is the earliest time preceding the accident in which:
  - there was an intersection between one of the cars and the other's corridor, and
  - the longitudinal distance was not safe.
- **Exposure Time:**
- The Exposure Time of an object is the first time in which we see it.
- **Blame due to Unreasonable Speed**
  - Assume that at the exposure time or after it, car c1 was driving at speed v > vlimit, and c0 wasn't doing so. Then, the blame is only on c1. We say that c1 is blamed due to unreasonable speed.

Responsibility Sensitive Safety (RSS): Definitions

- **Accident-with-Pedestrian Blame**
- The Accident-with-Pedestrian Blame is always on the car, unless
- one of the following three holds:
  - the car hits the pedestrian with the car's side, and the lateral velocity of the car is smaller than , w.r.t. the direction of the hit.
  - the pedestrian's velocity at the exposure time or later was larger than vlimit.
  - the car is at complete stop.
- **Default Emergency Policy (DEP)**
- The Default Emergency Policy (DEP) is to apply maximum braking power,
- and maximum heading change towards 0 heading w.r.t. the lane.

Responsibility Sensitive Safety (RSS): Definitions

- **Safe State**
- A state *s* is safe if performing DEP starting from it will not lead to an accident of our blame.

- **Cautious command**
- Suppose we are currently at state s0. A command *a* is cautious if the next state, s1, will be safe with respect to a set *A* of possible commands that other vehicles might perform now.

- **Sensing System**
- Let S denotes the domain of sensing state and let X be the domain of raw sensor and mapping data. A sensing system is a function X -> S.

Guaranteeing Multi-agent safety: Calculation of the minimal safe longitudinal distance

Definition. **Safe longitudinal distance**:

- A longitudinal distance between a car c_r and another car c_f that is in c_r's frontal corridor is safe w.r.t. a response time ρ if for any braking command a, $|a| < a_{max,brake}$, performed by c_f , if c_r will apply its maximal brake from time ρ until a full stop then it won't collide with c_f .
- Main parameters
- $a_{max,brake}$, $a_{max,accel}$: Maximum braking and acceleration commands.
- v_r; v_f : Longitudinal velocities of the rear and front cars
- l_f ; l_r : Lengths of the cars
- $v_{\rho,max} = v_r + \rho \times a_{max,accel}$
- ρ : Response time

Guaranteeing Multi-agent safety: Safe cut-in of the ego vehicle

What are the checks (conditions, requirements) so that there can be a safe cut-in of the ego vehicle?

Main parameters

- $\hat{c}$, c : Ego and another vehicle
- $[0, t_{brake}]$: time interval in consideration
- $\hat{c}_{diag}$: Length of a diagonal of a minimal rectangle bounding $\hat{c}$.
- $c_{length}(t)$: Longitudinal "span" of c in time t.
- $L(t) = (\hat{c}_{diag} + c_{length}(t))/2$
- $c_{width}(t)$: Lateral "span" of c in time t.
- $W(t) = (\hat{c}_{diag} + c_{width}(t))/2$

Safety of the Intended Functionality (SOTIF)

- The safety of self-driving vehicles is different from safety in other industries such as:
  - Avionics
  - Process Industries (Chemical, Oil & Gas, Petro-chemical, etc.)
  - Automotive

SELF-DRIVING VEHICLE SAFETY ATTRIBUTES

- Performance degradation
- Focus on software
- Non-deterministic perception system
- Perception system complexity
- Overall system complexity

Safety of the Intended Functionality (SOTIF)

PERFORMANCE DEGRADATION

- Traditional safety is based on faults and failures of mostly hardware components (*reliability approach to safety*)
- In contrast, accidents involving self-driving vehicles might happen even if no hardware device fails, but rather a performance degradation of some of its functions or intended functionality occurs, (SOTIF)
- Some failures are due to performance degradation of self-driving vehicle components, typically involving higher levels of processing or higher levels of automation, e.g., service failures.
- Examples of failures:
  - missed detections (i.e., false negatives)
  - spurious detections (i.e., false positives).

Safety of the Intended Functionality (SOTIF)

SELF-DRIVING VEHICLE SAFETY ATTRIBUTES

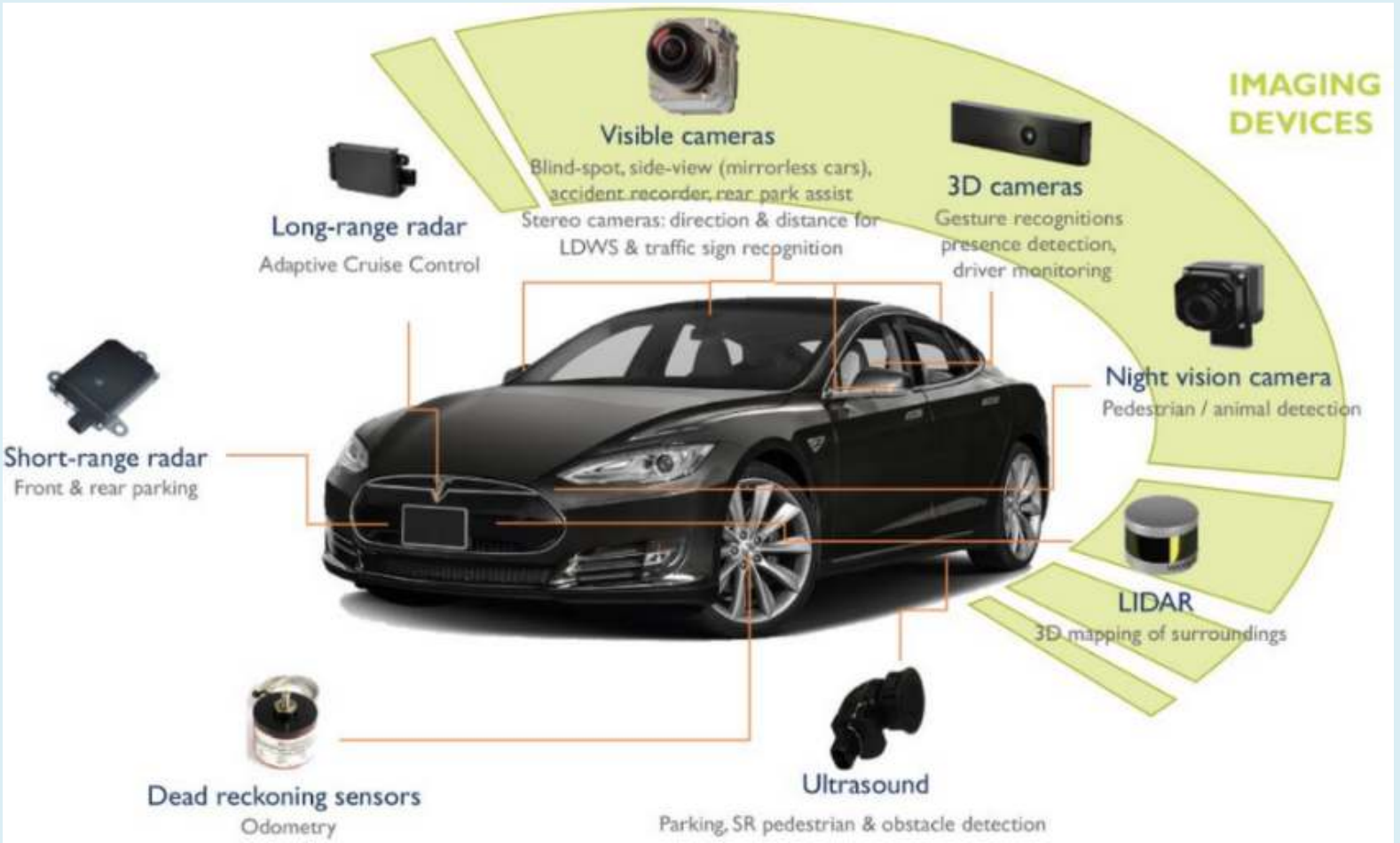
- Non-deterministic perception system
  - One cannot predict errors in the perception system
- Perception system complexity
  - The nature of data (e.g., 3D point cloud)
  - Huge size of data produced by sensors
- Overall system complexity
  - Computing
  - Communications
  - Localization and mapping
  - Control and actuation

Safety of the Intended Functionality (SOTIF)

- Related to the performance degradation attribute
- *Intended functionality:* "the behavior specified for an item, component,
- or service excluding safety mechanisms." (ISO 26262)
- **Examples of intended functionalities:**
  - vehicle detection
  - lane detection
  - road detection
  - pedestrian detection

- **SOTIF.** It involves accidents that can happen:
  - Even if all of the hardware components (i.e., sensors, computing units, comm unication networks) of the perception system are fault free.
  - Due to errors in the sensor processing and perception functions (e.g., vehicle detection).

Self-Driving Vehicles: The perception system



Credit: EE Times.

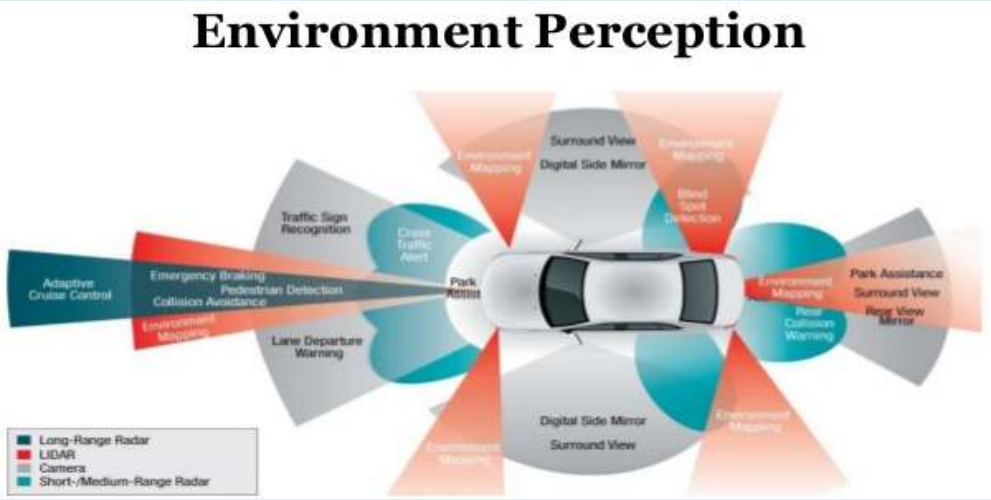
ROLE OF PERCEPTION IN AUTONOMOUS VEHICLES

- Sensory role
  - Image
  - Lidar
  - Radar
  - Ultrasonic
- Communication role
- Localization support role
- Planning role
- Control support role
- Safety role
- Security role

Object Avoidance

Perception system

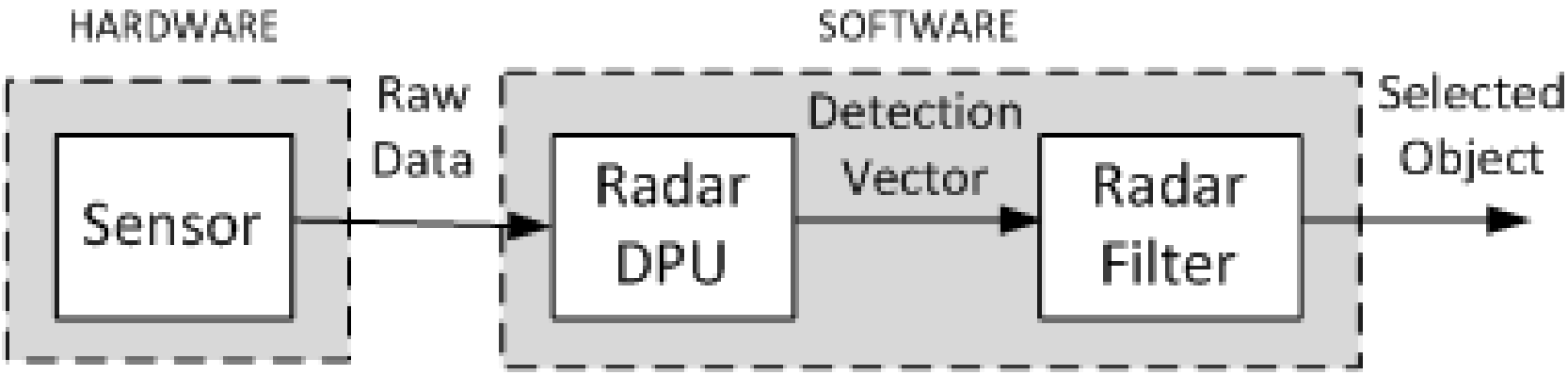
- Made up of sensors for autonomous driving and navigation
- Main sensors:
  - Radar (active): 77 – 81 GHz (long range)
  - Lidar (active): light generated by a laser.
  - Ultrasonic (active): not accurate enough.
  - Cameras (passive)



Self-Driving Vehicles: The radar sub-system

A radar detects range, velocity, and angle of several moving targets near the ego vehicle.

- The sensor unit emits raw Radar sensor data at a certain rate through a communication network that is typically CAN, CAN-FD, or Automotive Ethernet.
- The raw data is processed in software by the Radar DPU and Radar Filter units to produce a detection vector and selected object respectively.
- Detection Vector = Set of {position, velocity, angle}



Self-Driving Vehicles: The radar sub-system

- Data from Wireshark

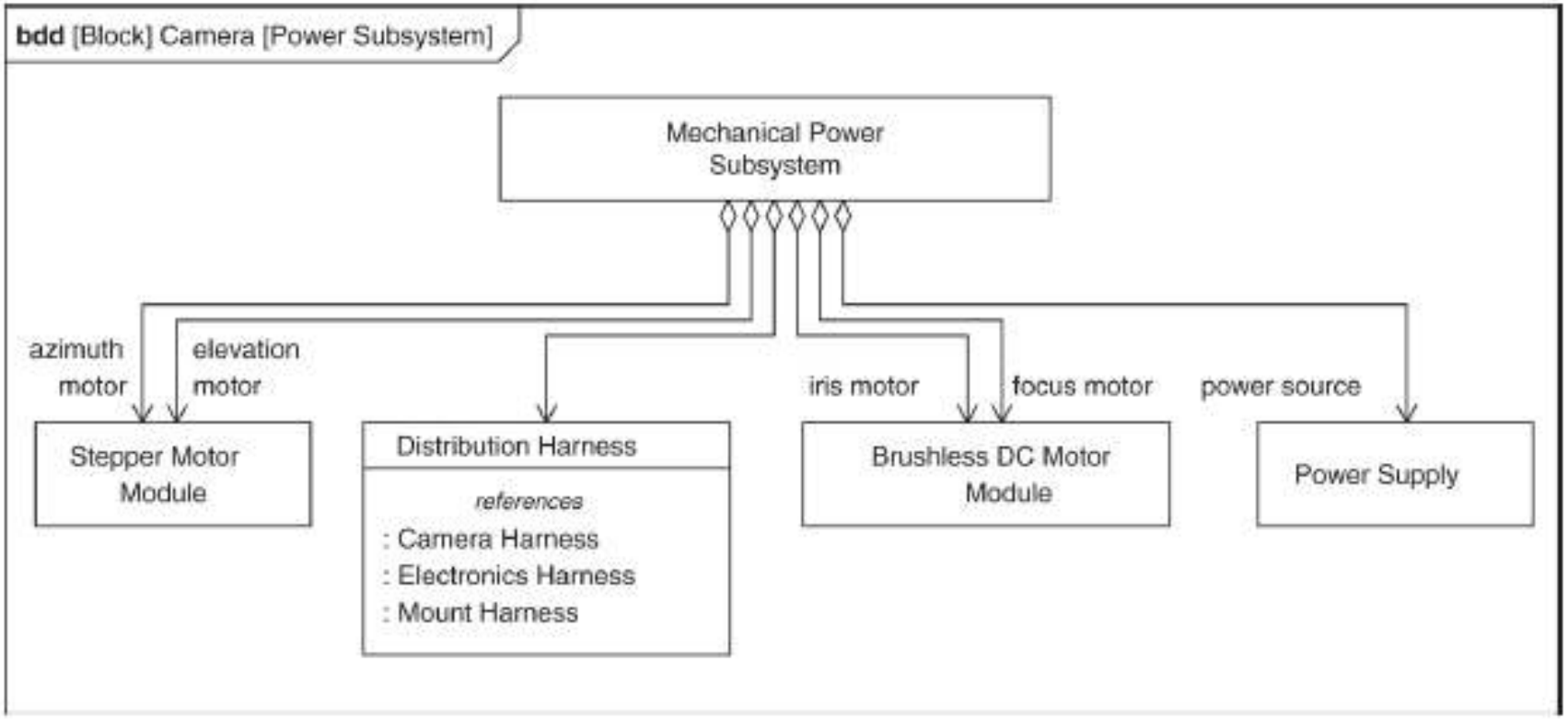
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
2	0.000224346	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
3	0.001273293	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
4	0.001501477	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
5	0.001681543	192.168.1.2	225.0.0.1	UDP	986	31122 → 31122 Len=944
6	0.009811230	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
7	0.010047936	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
8	0.011085131	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
9	0.011312675	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
10	0.011493180	192.168.1.2	225.0.0.1	UDP	986	31122 → 31122 Len=944
11	0.020366493	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
12	0.020570858	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
13	0.029852061	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
14	0.030086886	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
15	0.039940599	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
16	0.040232291	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
17	0.041403669	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
18	0.041837422	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
19	0.042016543	192.168.1.2	225.0.0.1	UDP	986	31122 → 31122 Len=944
20	0.049841389	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
21	0.050088067	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
22	0.051150971	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112
23	0.051380226	192.168.1.2	225.0.0.1	UDP	1154	31122 → 31122 Len=1112

```
> Frame 1: 1154 bytes on wire (9232 bits), 1154 bytes captured (9232 bits) on interface 0
> Ethernet II, Src: MS-NLB-PhysServer-08_02:03:00:00 (02:08:02:03:00:00), Dst: IPv4mcast_01 (01:00:5e:00:00:01)
> Internet Protocol Version 4, Src: 192.168.1.2, Dst: 225.0.0.1
> User Datagram Protocol, Src Port: 31122, Dst Port: 31122
> Data (1112 bytes)
```

```
0000  01 00 5e 00 00 01 02 08  02 03 00 00 08 00 45 00   ··^····· ······E·
0010  04 74 a9 b9 00 00 ff 11  6b 13 c0 a8 01 02 e1 00   ·t····· k·······
0020  00 01 79 92 79 92 04 60  fc 7b 00 dc 00 01 00 00   ··y·y··` ·{····
0030  04 50 00 00 00 00 01 01  01 00 5b ed 04 50 bc 01   ·P····· ··[··P··
0040  00 04 68 d0 e3 c0 31 38  bc 0a 6a e6 00 00 dd fc   ··h···18 ··j·····
0050  00 00 dd fc 00 21 42 37  05 21 03 12 00 00 05 8f   ·····!B7 ·!······
0060  fc 7b 00 c3 e9 80 e5 67  e5 19 08 46 00 22 00 04   ·{·····g ···F·"··
0070  00 04 00 04 20 77 06 18  00 00 08 7a ff 57 03 3d   ···· w·· ···z·W·=
```

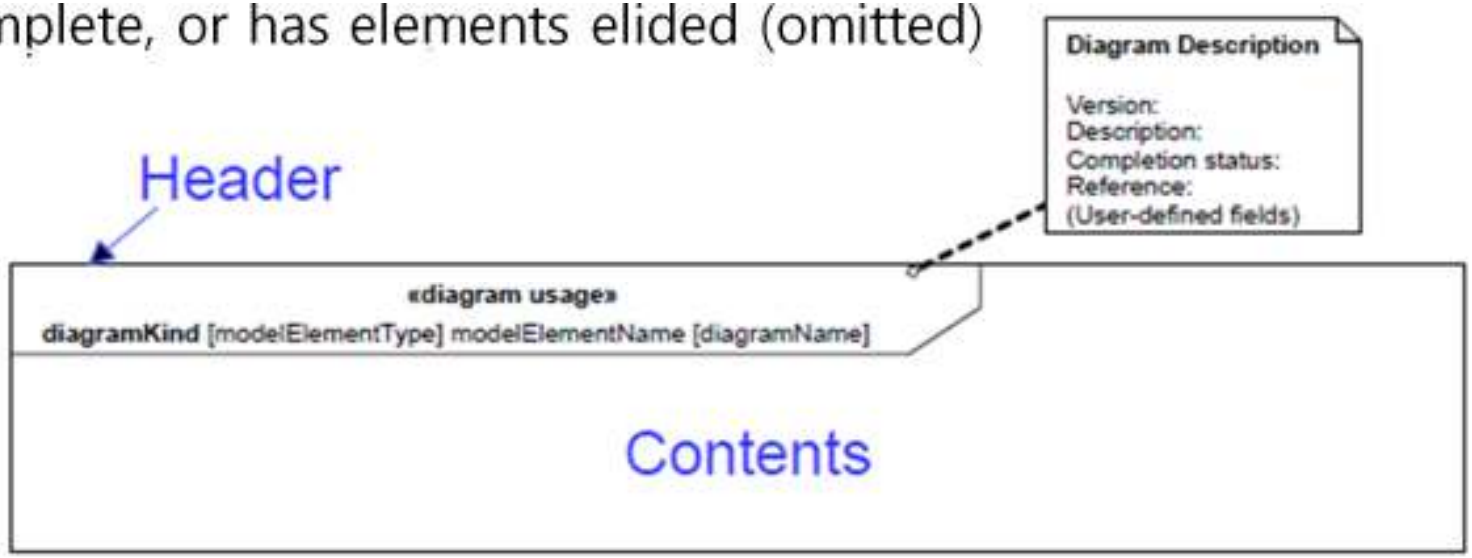
SysML: Example of a bdd diagram



This diagram says that the **Mechanical Power Subsystem** of a **Camera** is composed of several modules: **Stepper Motor**, **Distribution Harness**, **Brushless DC Motor** and **Power Supply**.

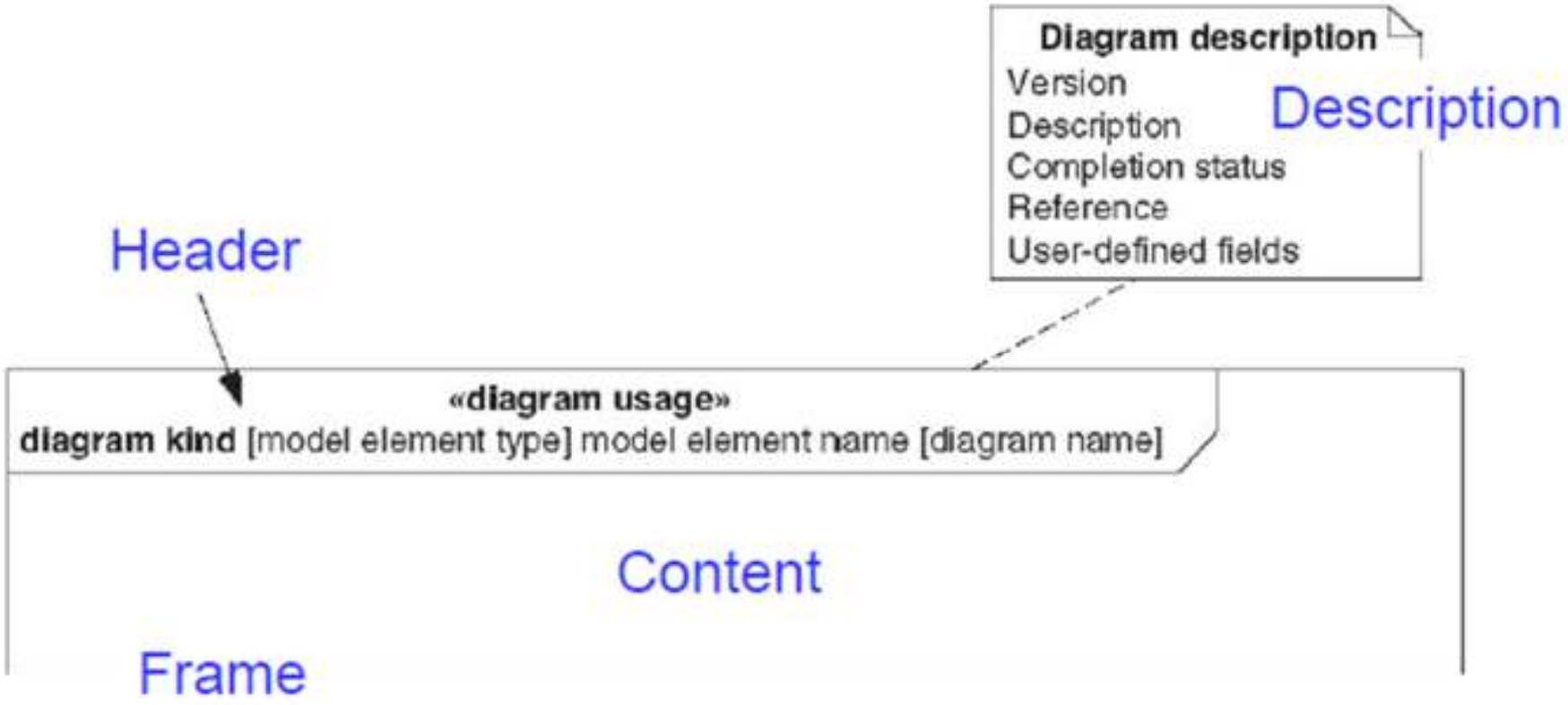
SysML diagram structure

- Each SysML diagram focuses into a single model element
- Note that the model is hidden and we use diagrams as views
- Diagram context and ID is indicated in the header:
  - – Diagram kind (act, bdd, ibd, sd, etc.)
  - – Model element type (package, block, activity, etc.)
  - – Model element name
  - – Diagram name or view name
- A separate diagram may be used to indicate if the diagram is complete, or has elements elided (omitted)



SysML diagram components

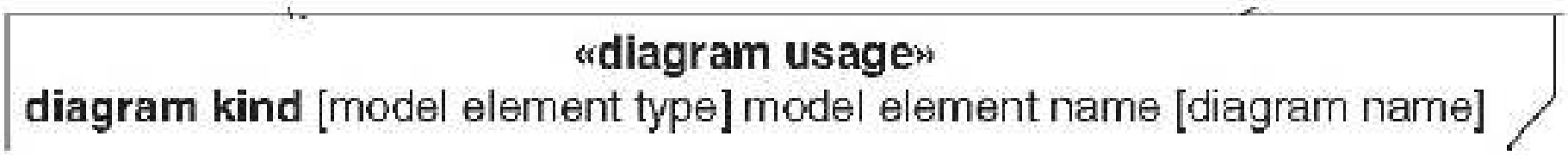
- Frame
- Header
- Content
- Description



SysML diagram header

A rectangle with lower right corner cut-off. It includes the following:

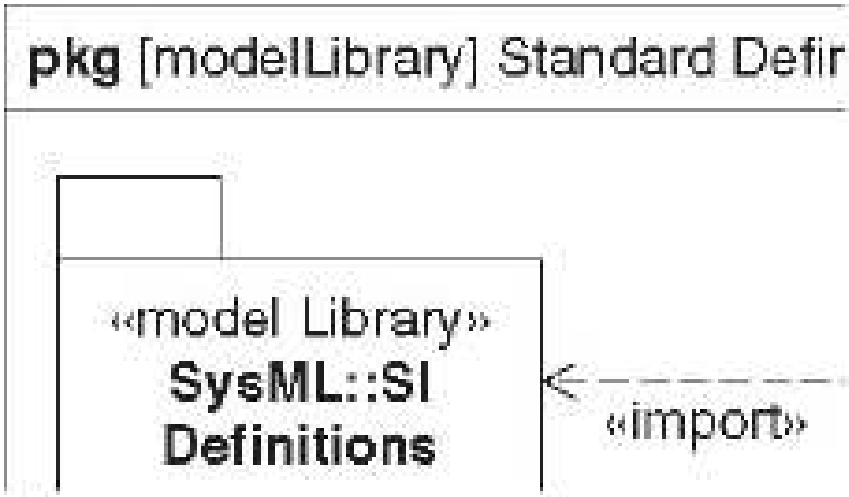
- – **diagramKind** – an abbreviation indicating the Type of Diagram
- – [**modelElementType**] – type of Model Element the diagram represents
- – **modelElementName** – Name of the represented model element
- – [**diagramName**] – Name of the diagram
- – <<**diagram usage**>> - Keyword indicating specialized diagram use



SysML diagram header

The diagram header **diagramKind** indicates the type of diagram:

- **req** – Requirement diagram
- **act** – Activity diagram
- **sd** – Sequence diagram
- **stm** – State Machine diagram
- **uc** – Use Case diagram
- **bdd** – Block Definition diagram
- **ibd** – Internal Block diagram
- **par** – Parametric diagram
- **pkg** – Package diagram



SysML diagram header

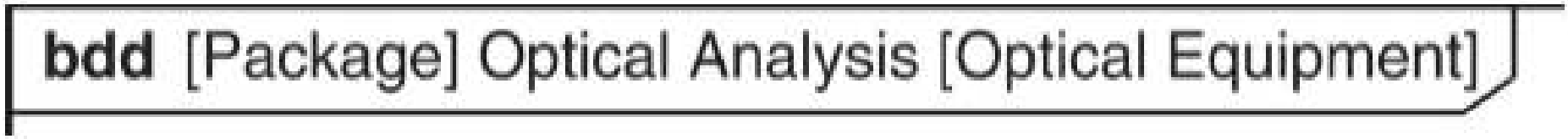
The [modelElementType] indicates the type of model element that the diagram represents

Not all combinations are possible:

- – **req** – package, model, model library, requirement
- – **act** – activity, control, operator
- – **sd** – interaction
- – **stm** – state machine
- – **uc** – package, model, model library
- – **bdd** – block, constraint block, package, model, model library
- – **ibd** – block
- – **par** – block, constraint block
- – **pkg** – package, model, model library, profile, view

SysML diagram header

- SysML Diagram Header ‘modelElementName’
  - – Indicates the Name of the represented model element
  - – A diagram targets a single model element
- SysML Diagram Header [diagramName]
  - – Indicates the Name of the diagram
  - – Used to provide a description of the diagram purpose
  - – There may be many diagrams for one model element
  - – Modelers may want to only highlight selected features in a diagram for a specific purpose, and may hide irrelevant features



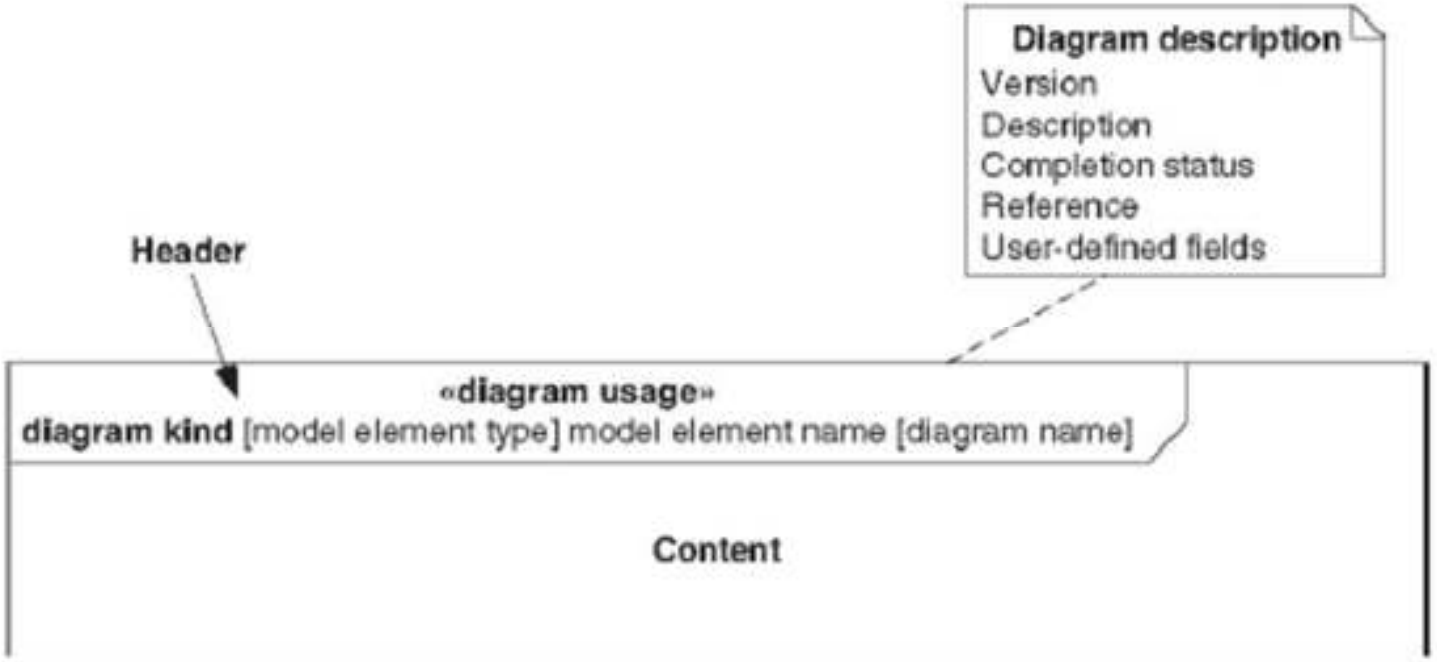
SysML diagram header

<<diagram usage>> is a stereotype that can be used in the SysML diagram header.

- ▶ – Indicates a specialized use of a diagram
- ▶ – For example, <<Context Diagram>> may be a specific use of a Use Case Diagram, Block Definition Diagram, or an Internal Block Diagram

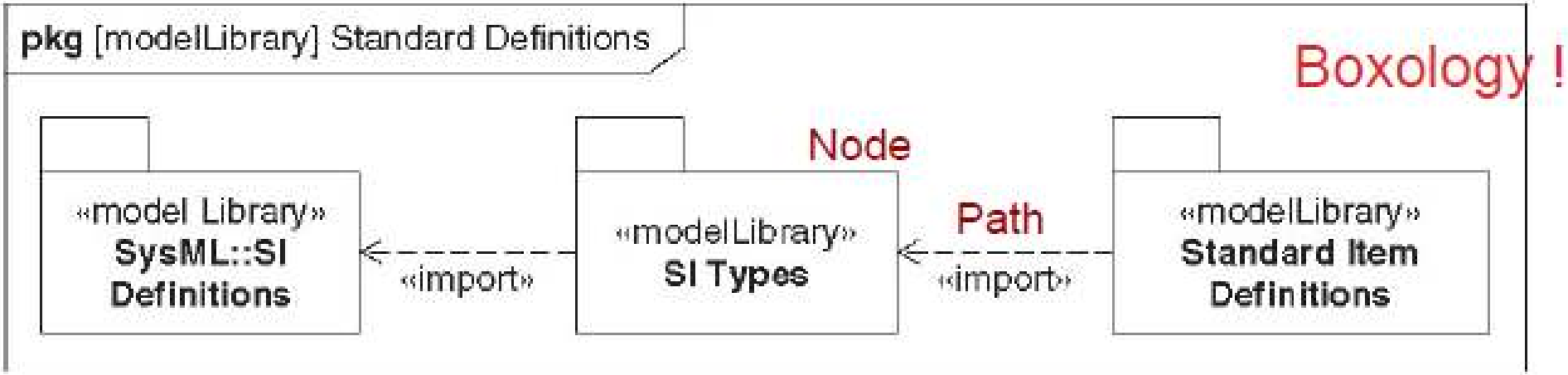
SysML

- **Diagram Description** is an optional note attached either to the inside or outside of the diagram frame
- Enables modelers to capture additional information about the diagram
- Pre-defined fields for the Diagram Description. It includes:
  - – Version
  - – Completion status
  - – Description
  - – Reference



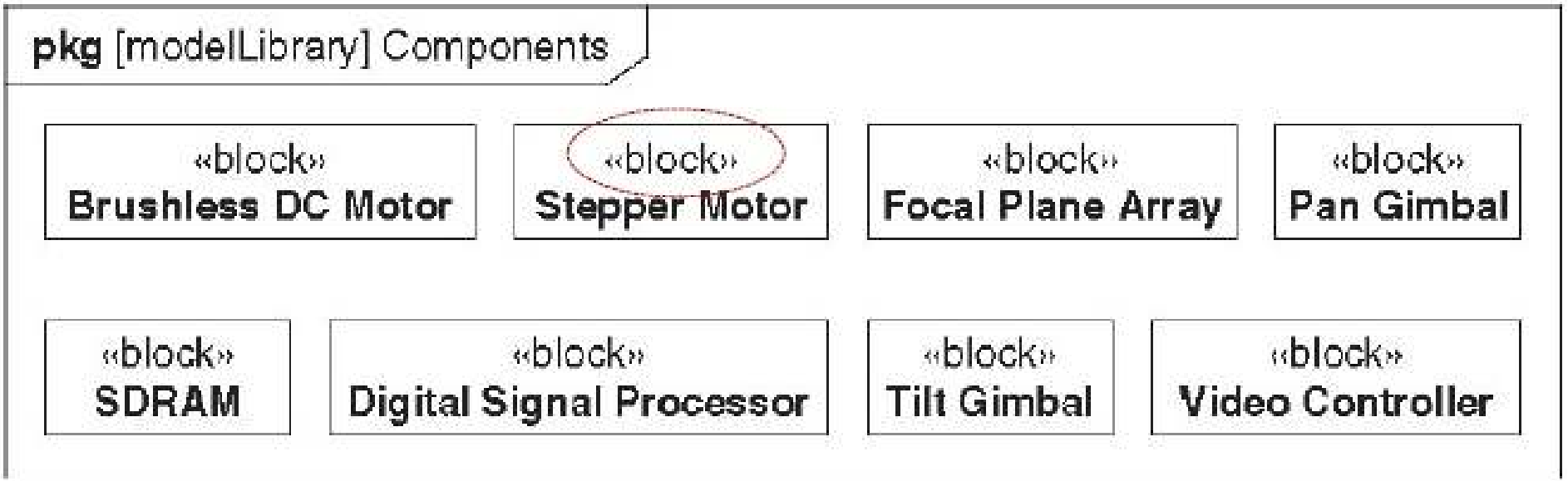
SysML diagram content

- Contains graphical elements that represent underlying elements in the model
- Two basic types of elements: **Nodes** and **Paths**
  - – Node: symbol that can contain text and/or other symbols to represent the internal details of the represented model element
  - – Path: line associated with Nodes that may have additional aspects such as arrows and text strings

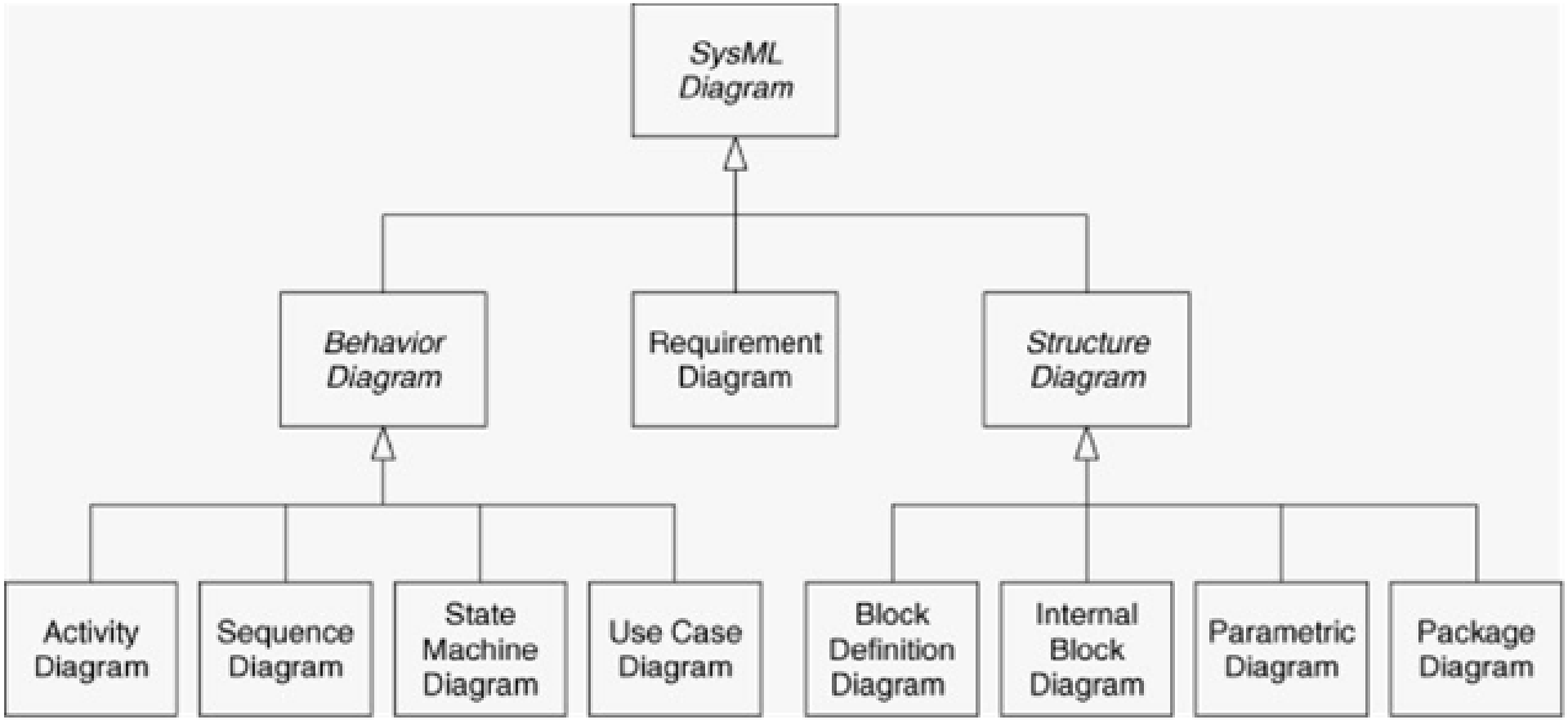


SysML diagram content

- Stereotypes are used to qualify elements
- <<keyword>> Identifies the type of model element
- Examples: <<block>>,<<modellibrary>>

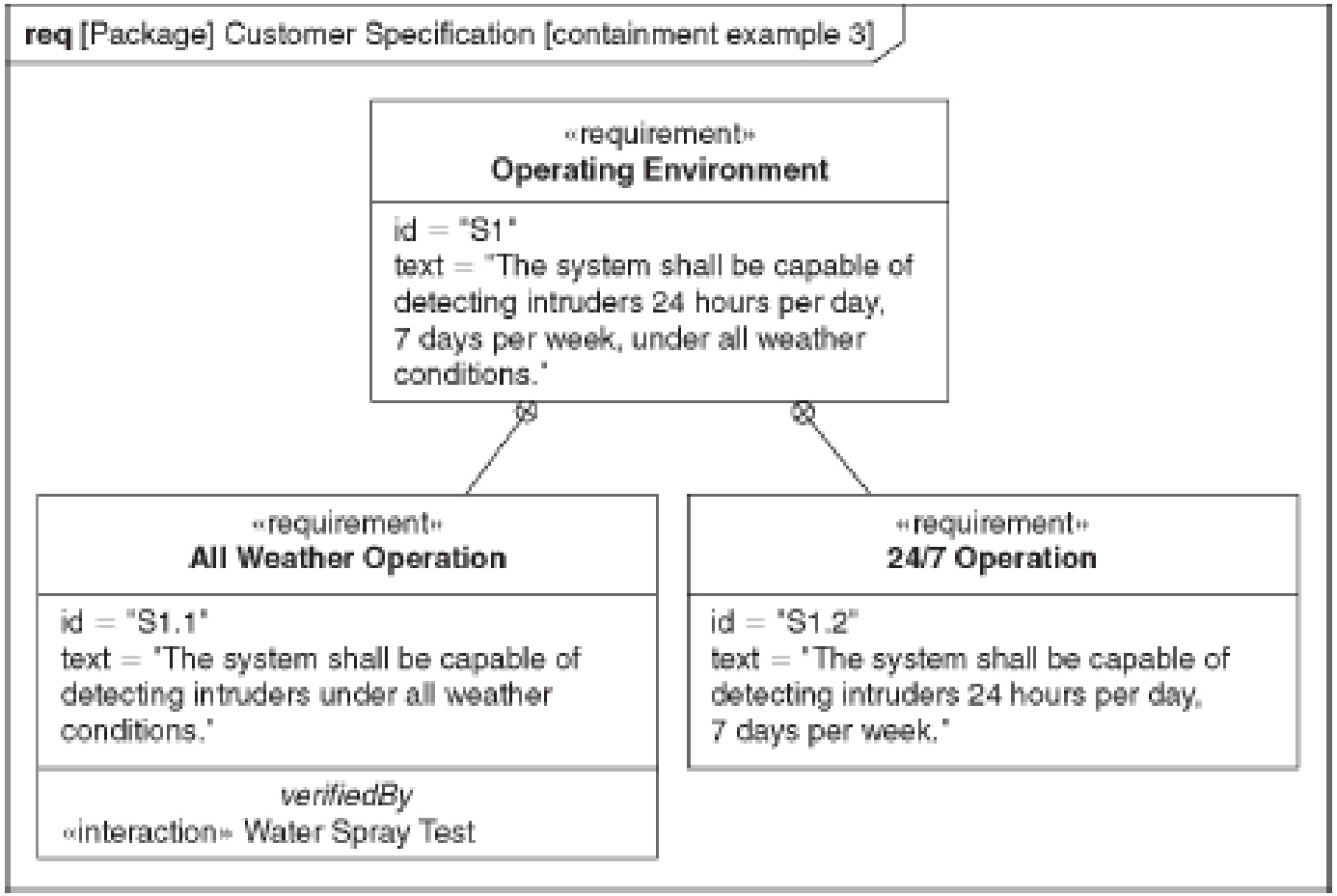


SysML diagram overview: Getting familiar with them



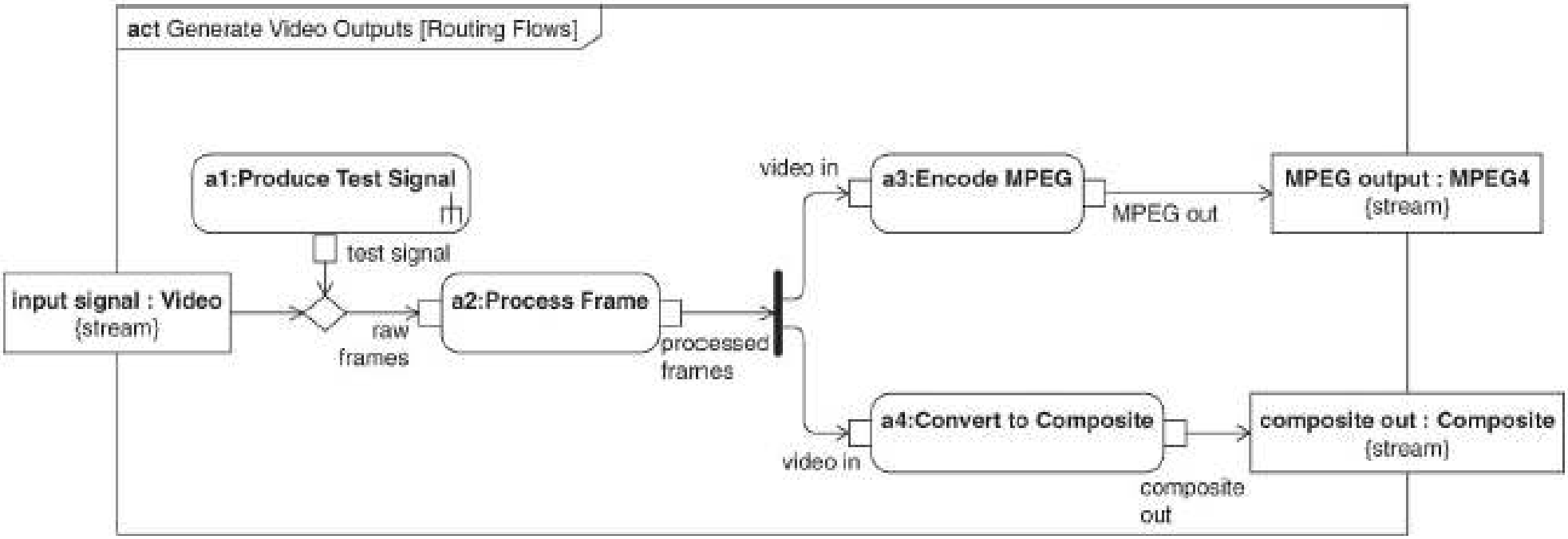
SysML requirements diagram

- Represents **text-based requirements** and their relationships with other requirements, design elements, and test cases to support requirements **traceability**



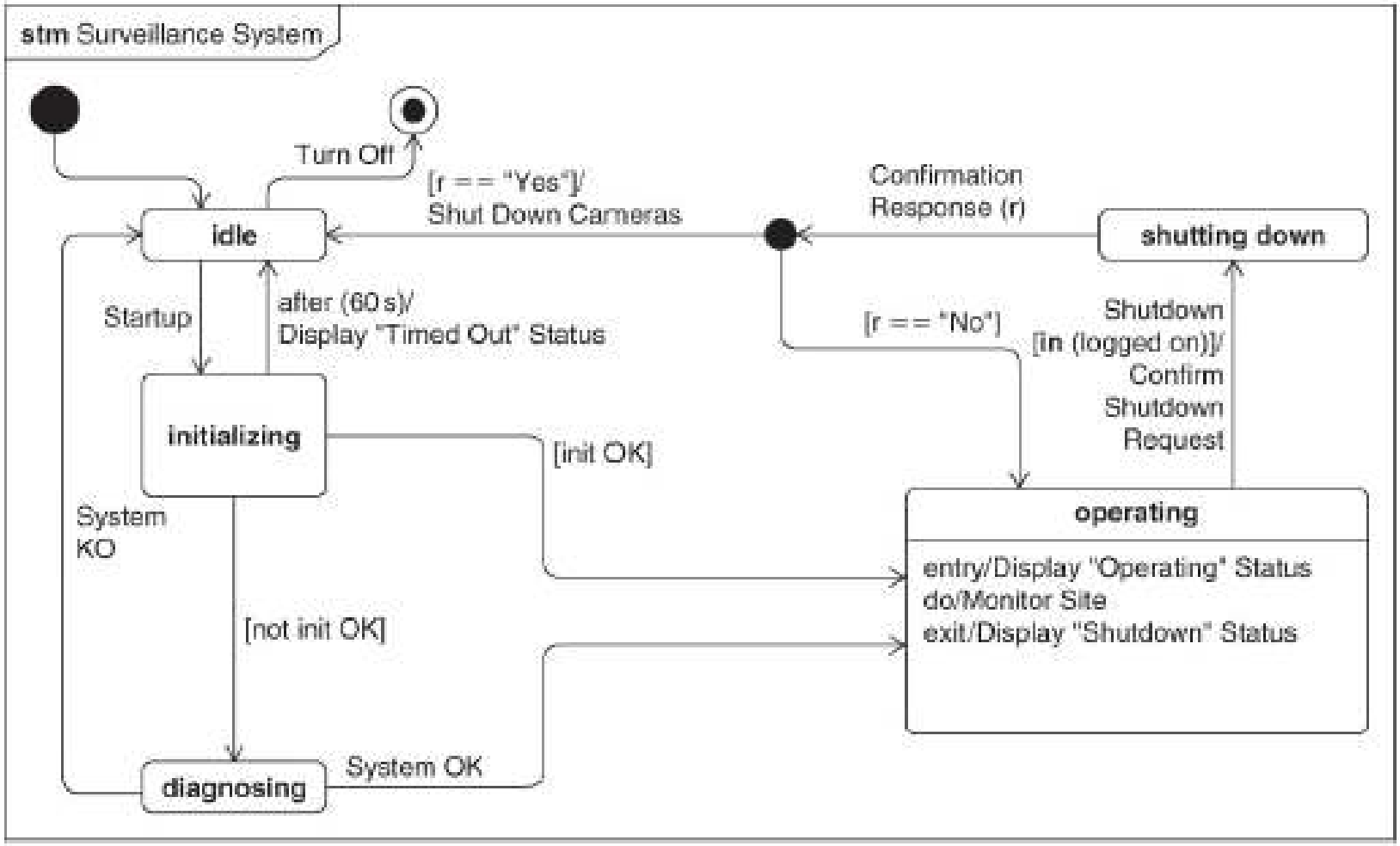
SysML activity diagram

- Represents **behavior** in terms of the **ordering of actions** based on the availability of inputs, outputs, and control, and how the actions transform the inputs to outputs



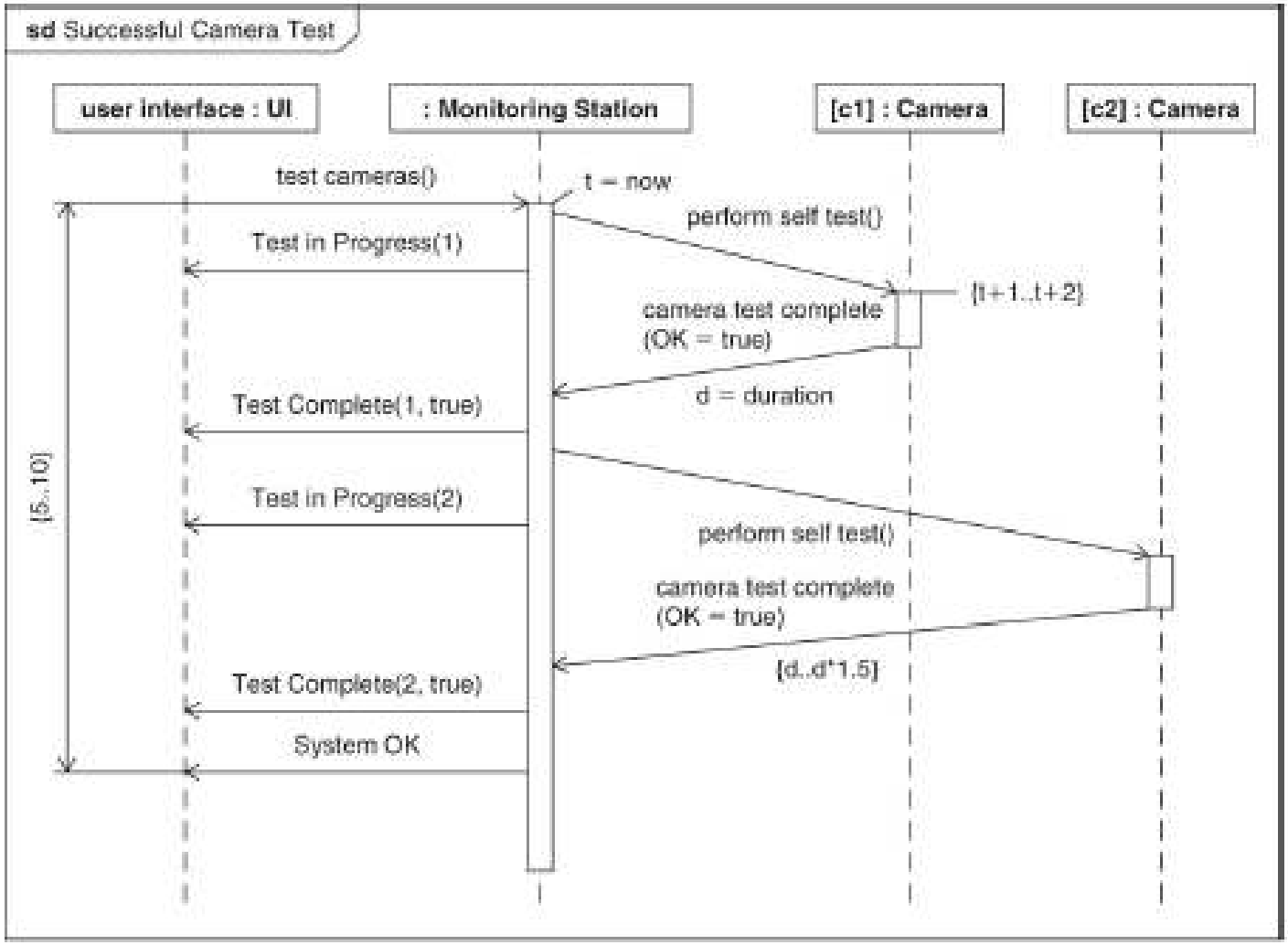
SysML state machine diagram

- Represents **behavior** of an entity in terms of its transitions between **states** triggered by **events**



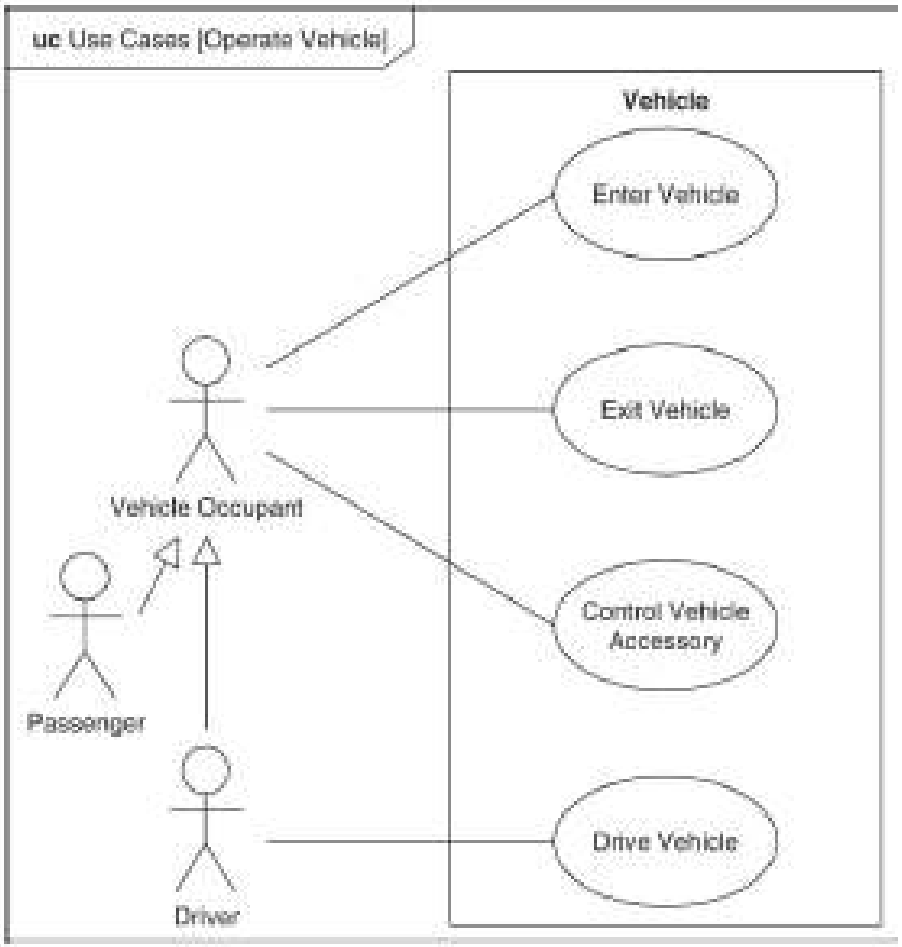
SysML sequence diagram

- Represents **behavior** in terms of a sequence of **messages exchanged** between parts



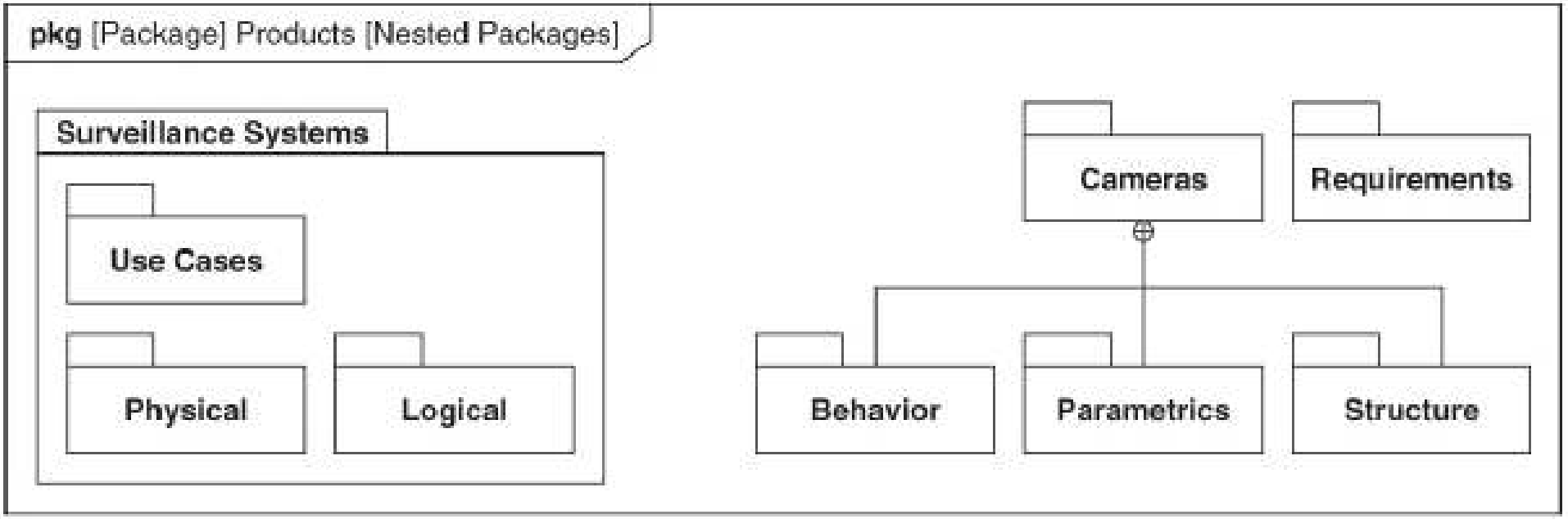
SysML use-case diagram

- Represents **functionality** in terms of how a system or other entity is used by external entities (i.e., actors) to accomplish a set of goals



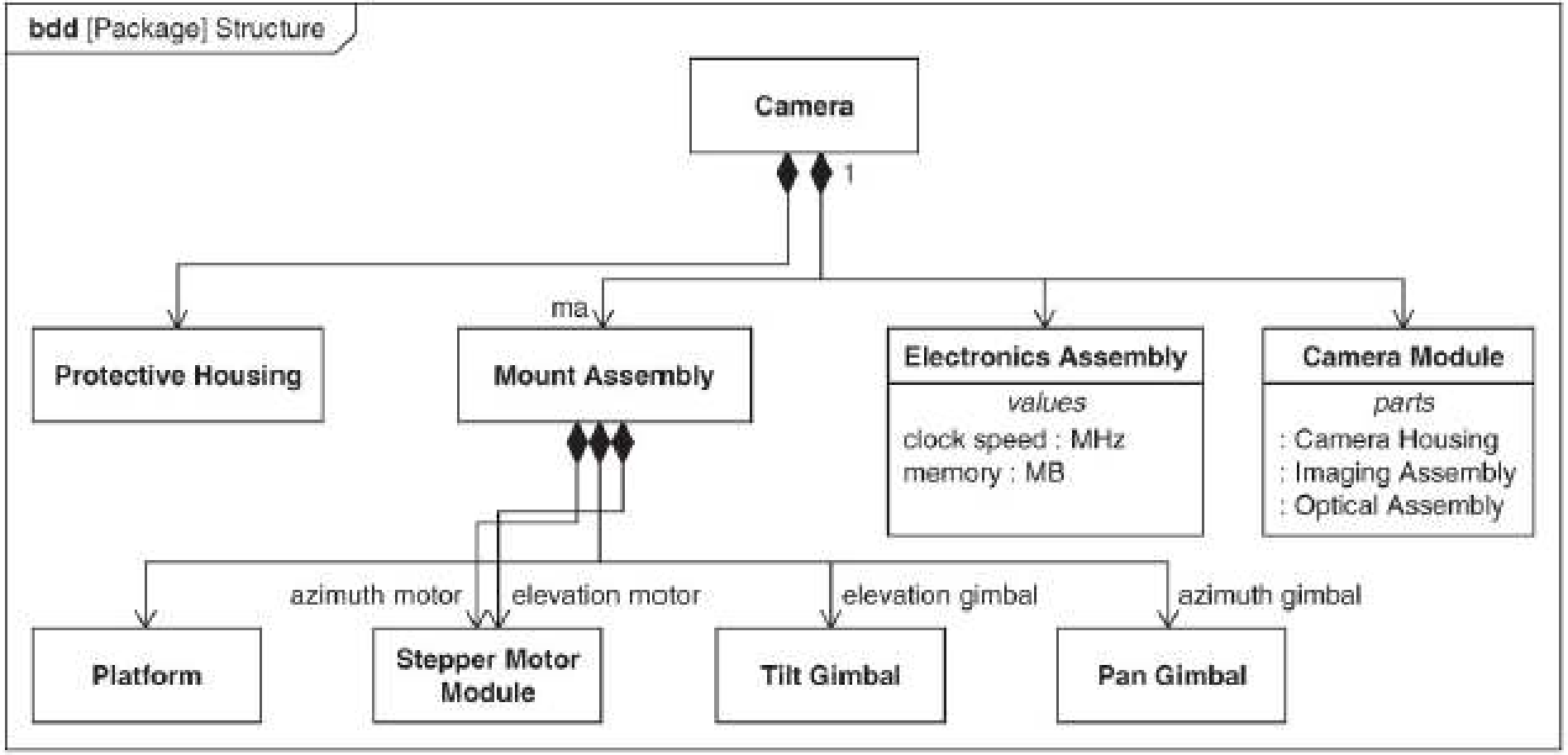
SysML package diagram

- Represents the **organization** of a model in terms of packages that contain model elements
- Note the use for model structuring, not system structuring



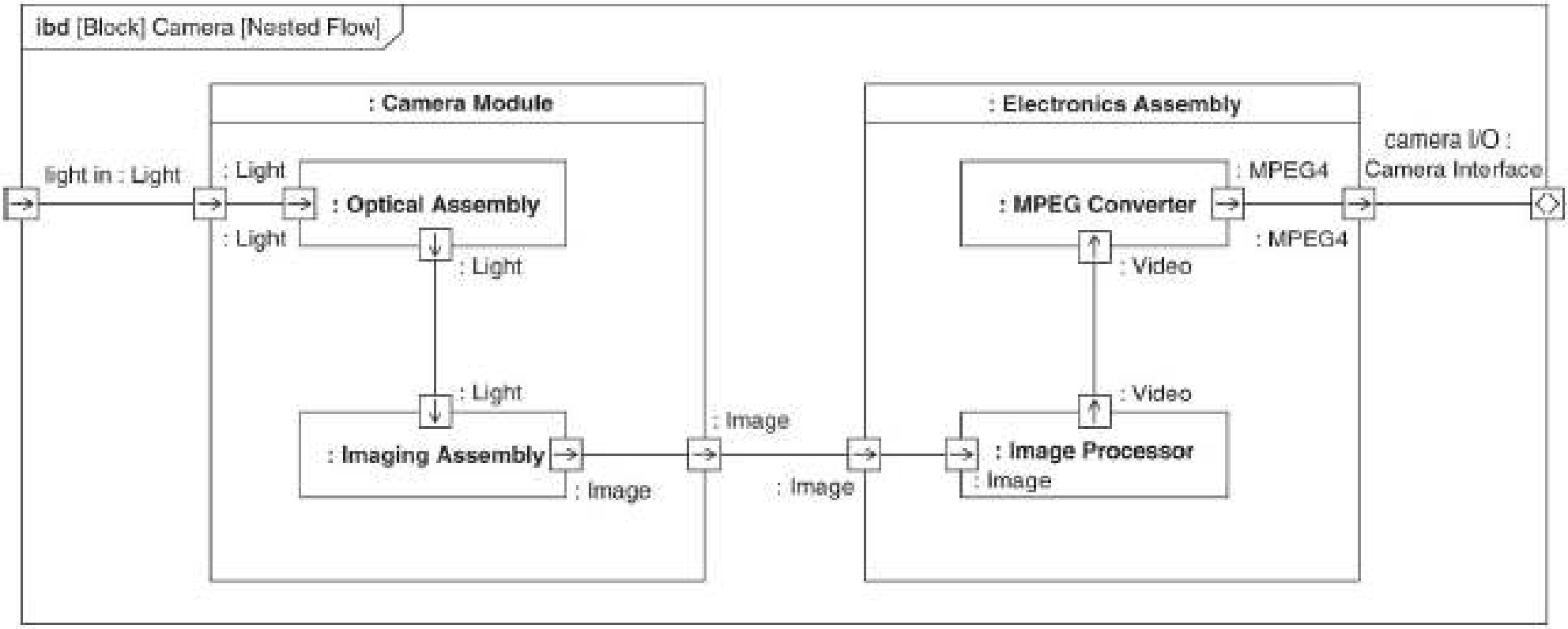
SysML block definition diagram (bdd)

- Represents **system elements** called blocks, and their composition, classification and navigation



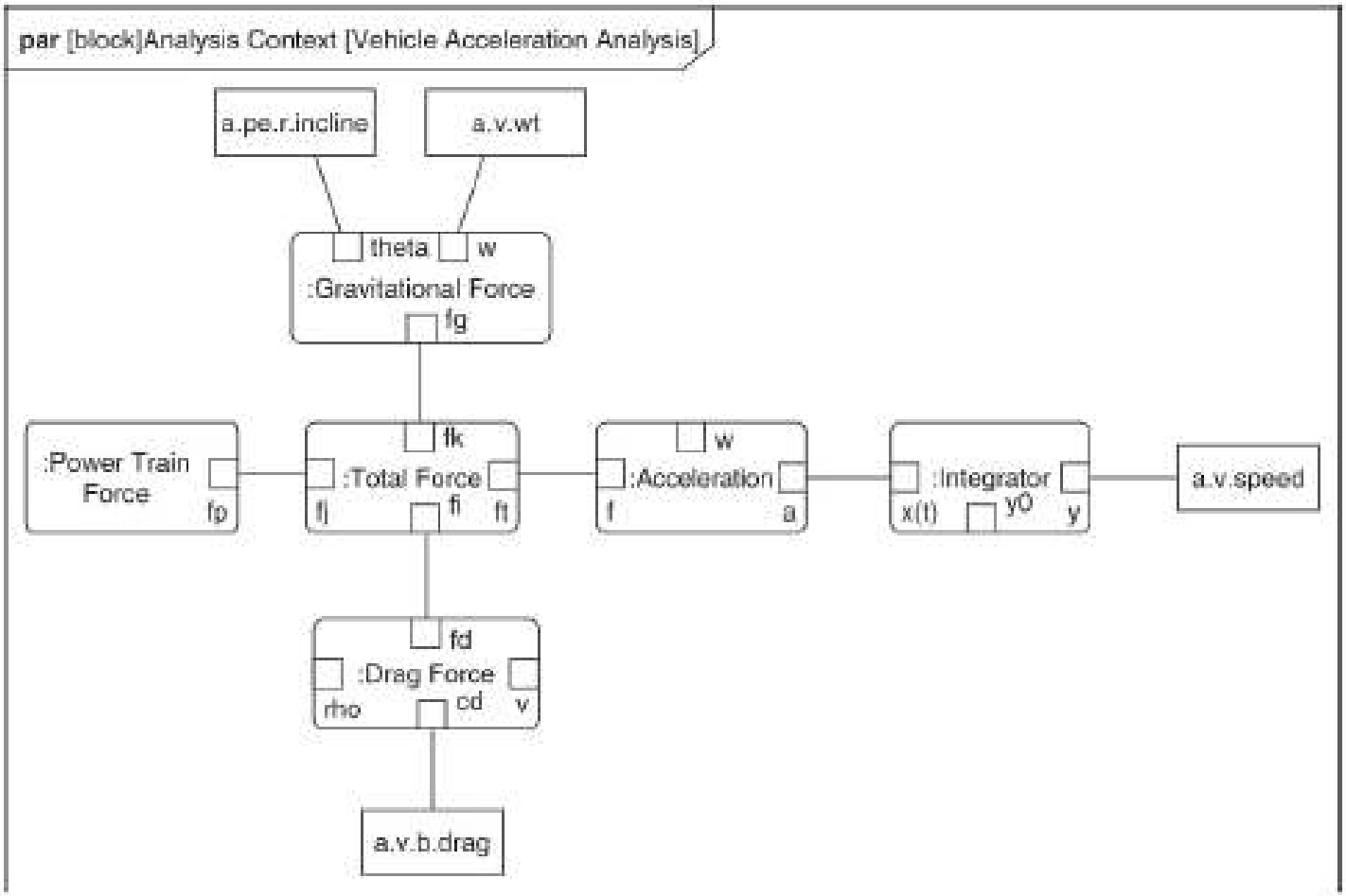
SysML internal block diagram (ibd)

- Represents **interconnection** and **interfaces** between the parts of a block
- Also shows **external ports** of the block



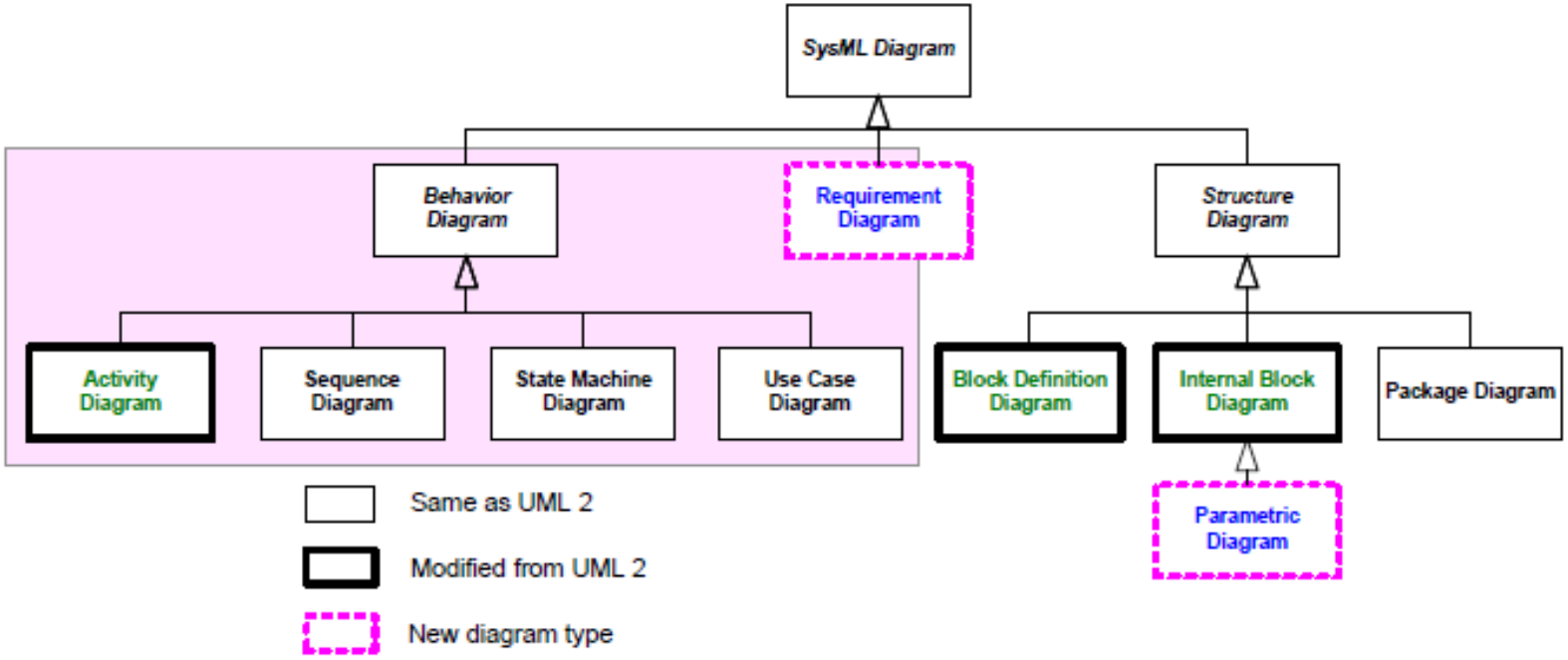
SysML parametric diagram (par)

- Represents **constraints** on property values, such as *F = m*a*, used to support engineering analysis

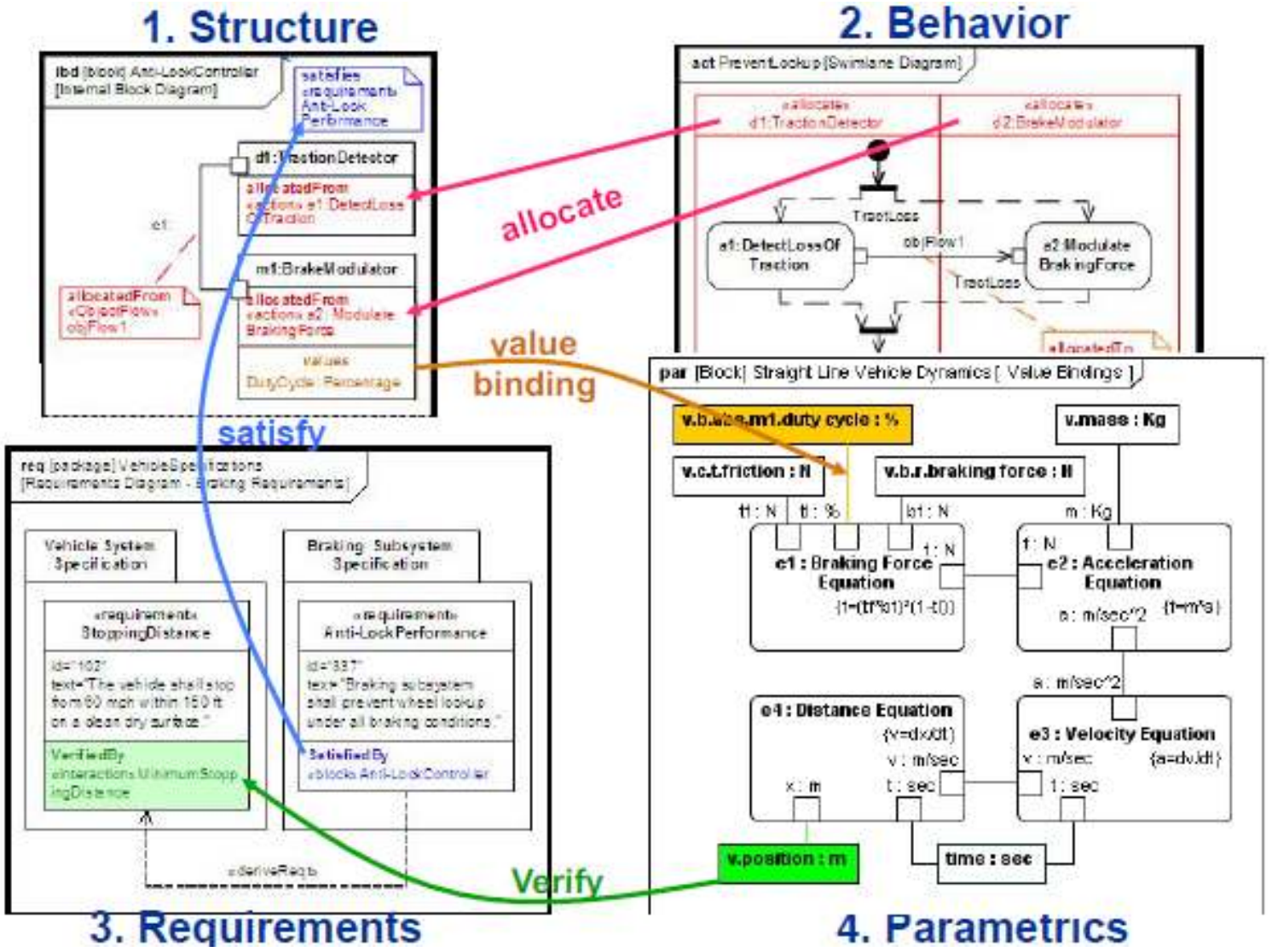


SysML: cross-cutting aspects

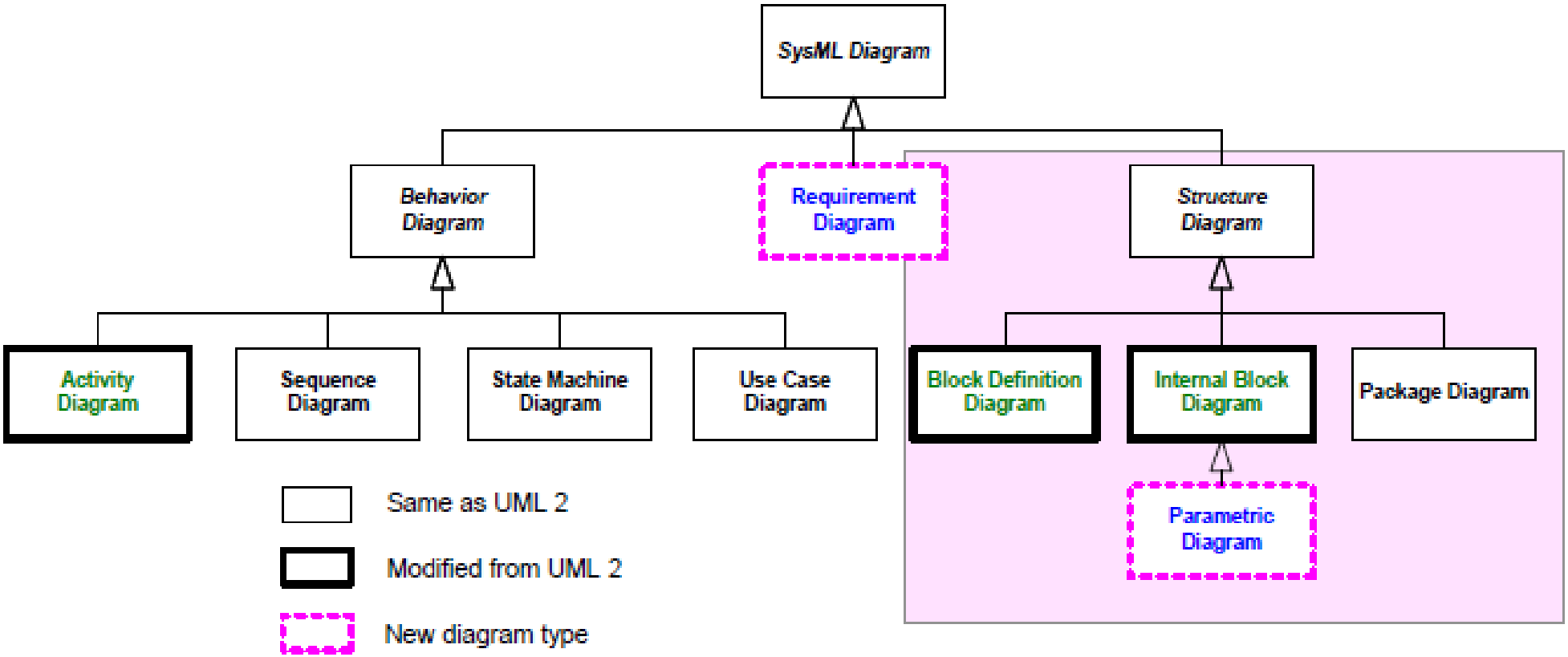
- Allocations:
  - – Represent general relationships that map one model element to another
- Requirements
  - – Text-based specification and relationships
- Dependencies
  - – Represents a relationship between client and supplier elements



SysML: Cross-connecting model elements

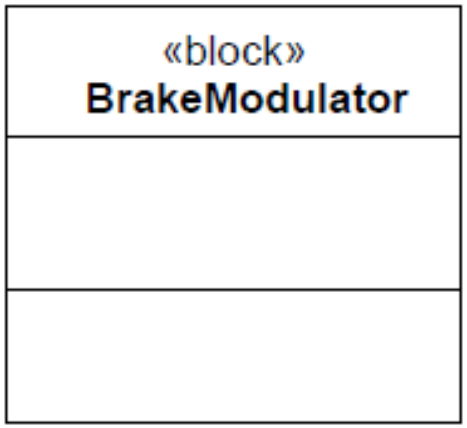


SysML: Structural diagrams



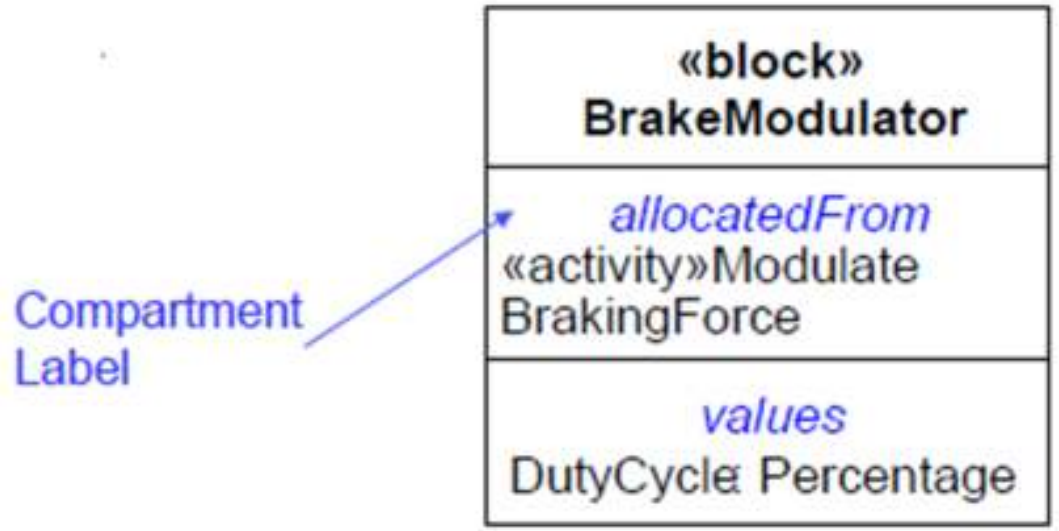
SysML: Blocks

- Are the cornerstone of SysML models
- They define what exists, as reality or idea - ontology
- Provides a unifying concept to describe the structure of an element or subsystem
  - – System, Hardware, Software, Data, Procedure, Facility, Person
- Unification of classes and assemblies from UML
- Enable hierarchy by nesting
- Specification of value types with units, dimensions, and probabilities
- Blocks are used to Specify Entities, Hierarchies and Interconnection

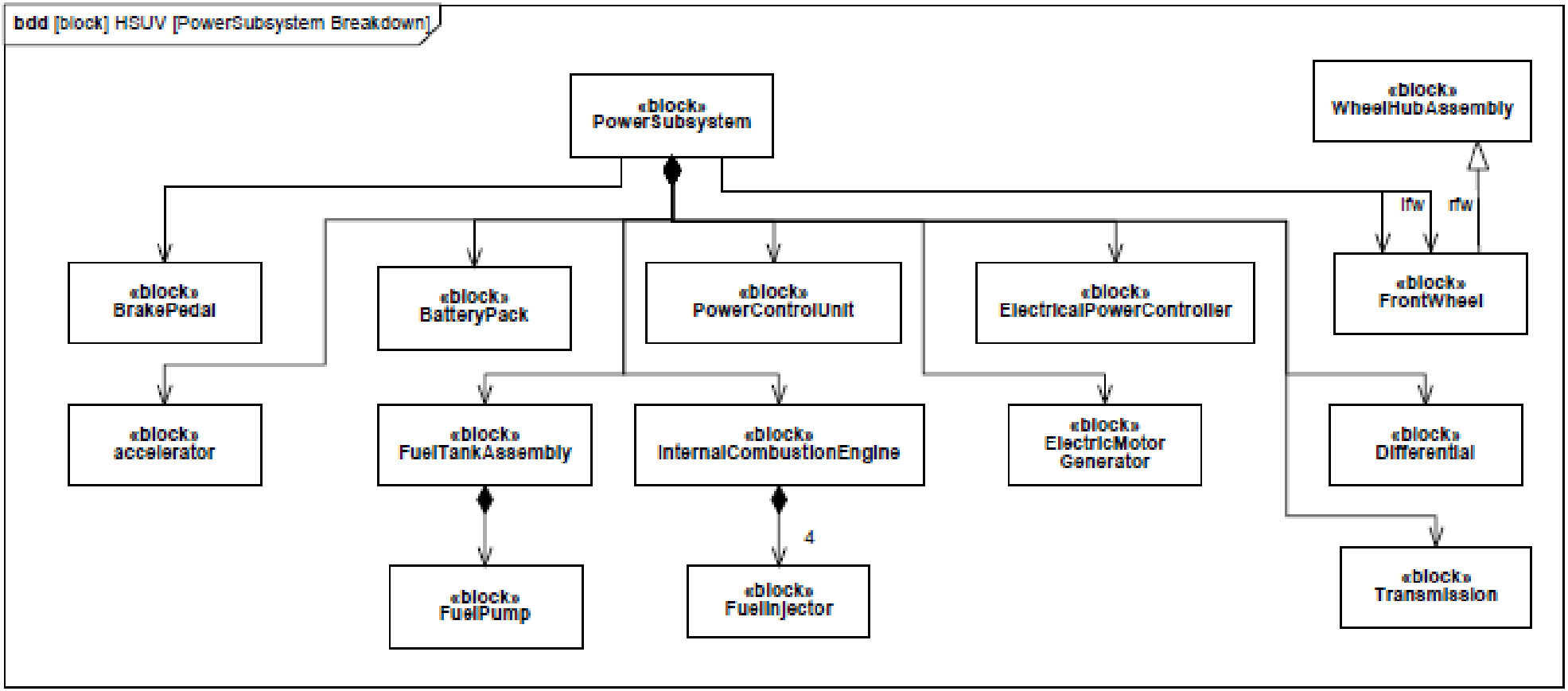


SysML: Blocks are basic structural elements

- Multiple standard compartments can describe the block characteristics
  - – Properties (parts, references, values, ports)
  - – Operations
  - – Constraints
  - – Allocations from/to other model elements (e.g. activities)
  - – Requirements the block satisfies
  - – User defined compartments

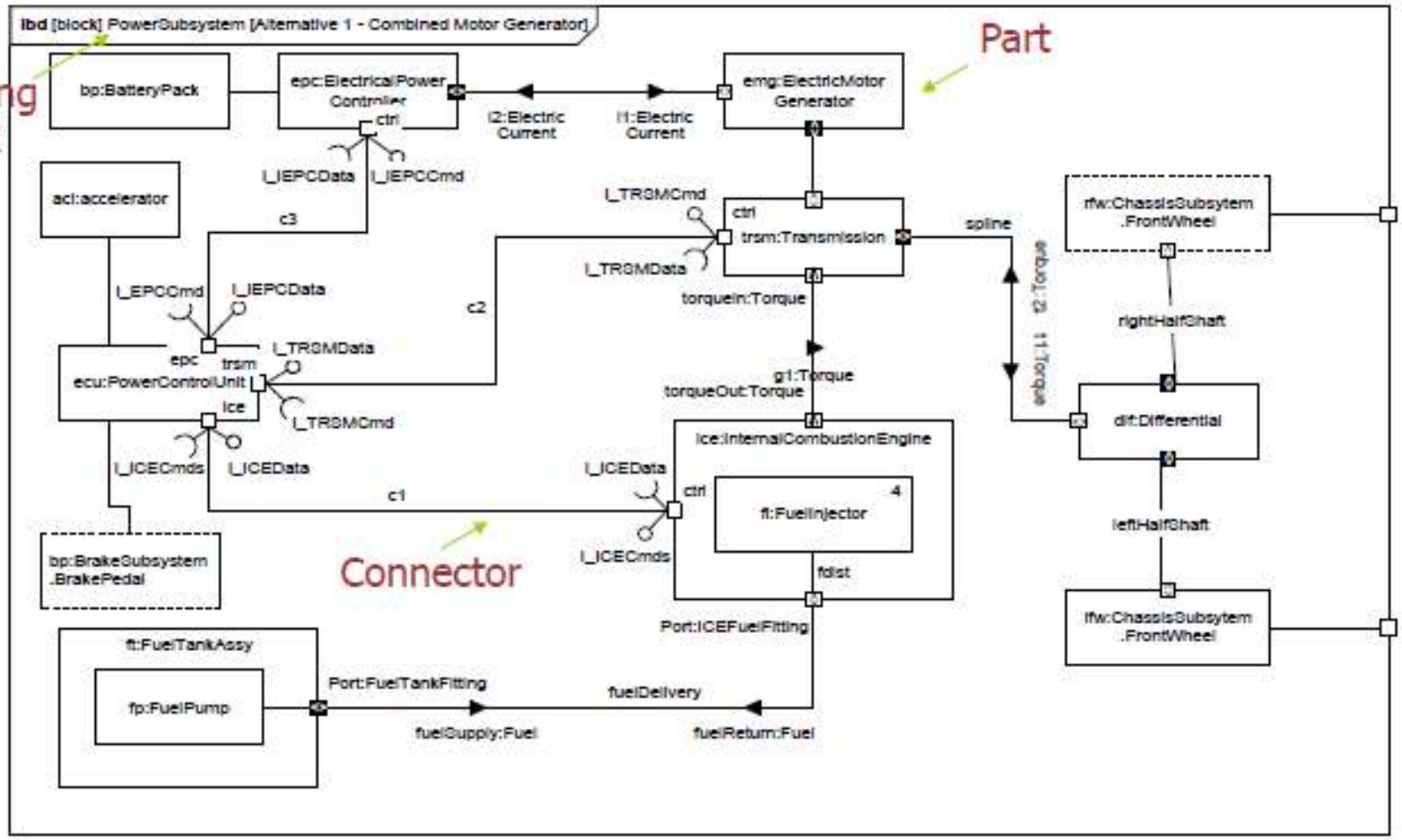


SysML: Power sub-system bdd



Block Definition Diagram Used to Specify System Hierarchy and Classification

SysML: Power sub-system ibd



Internal Block Diagram Used to Specify Interconnection Among Parts in Context of Enclosing Block

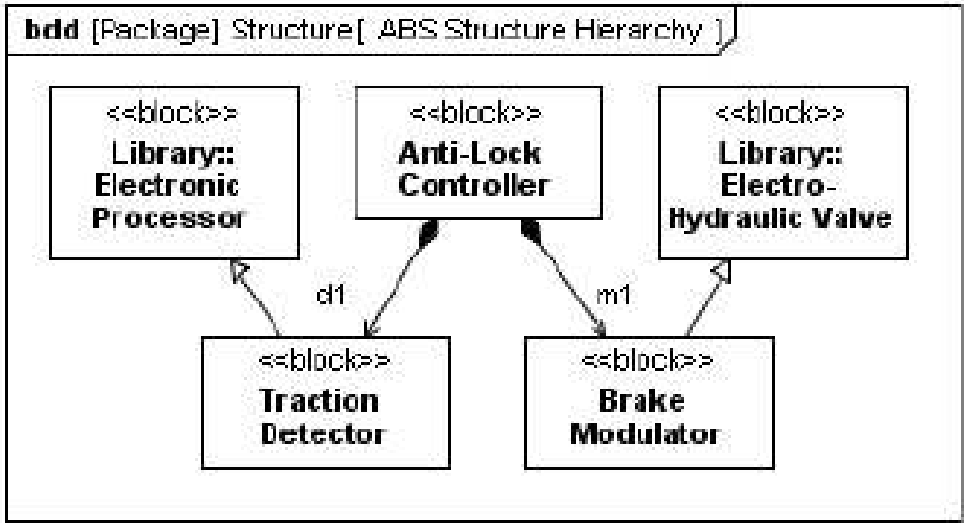
SysML: Property types

Property is a structural feature of a block

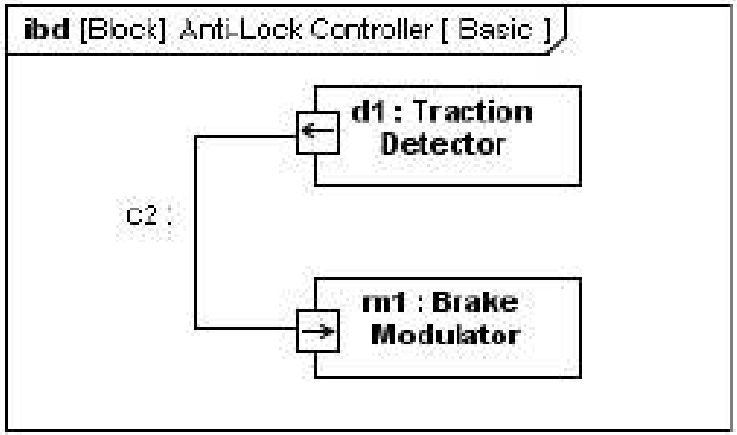
- – **Part property** aka. part (typed by a block)
  - • Usage of a block in the context of the enclosing (composite) block
  - • Example - right-front:wheel
- – **Reference property** (typed by a block)
  - • A part that is not owned by the enclosing block (not composition)
  - • Example – aggregation of components into logical subsystem
- – **Value property** (typed by value type)
  - • A quantifiable property with units, dimensions, and probability
  - distribution
  - • Example
  - – *Non-distributed value:* tirePressure:psi=30
  - – *Distributed value:* «uniform» {min=28,max=32} tirePressure:psi

SysML: Block definition vs. usage

Block Definition Diagram



Internal Block Diagram



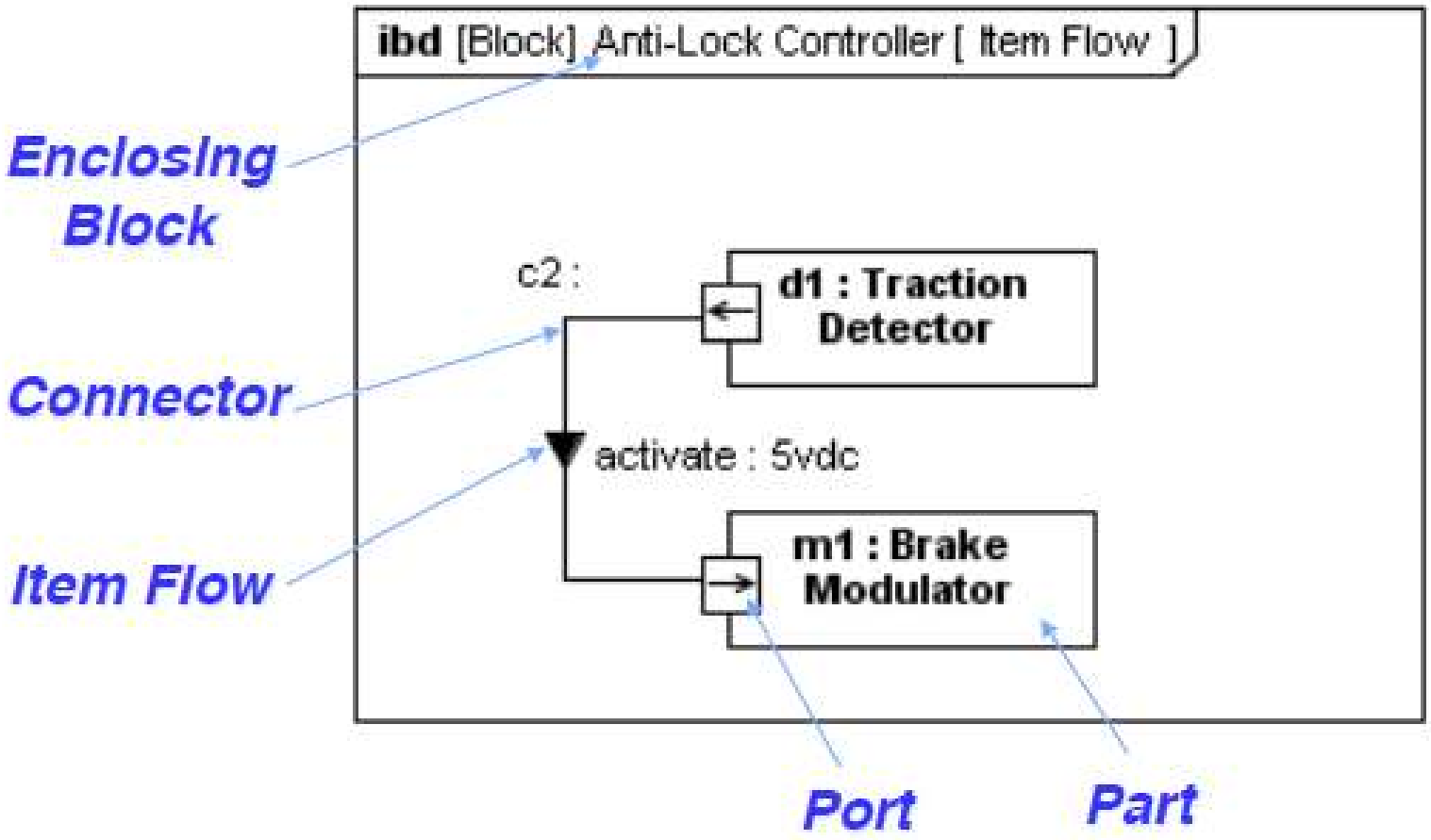
Definition

- Block is a definition/type
- Captures properties, etc.
- Reused in multiple contexts

Usage

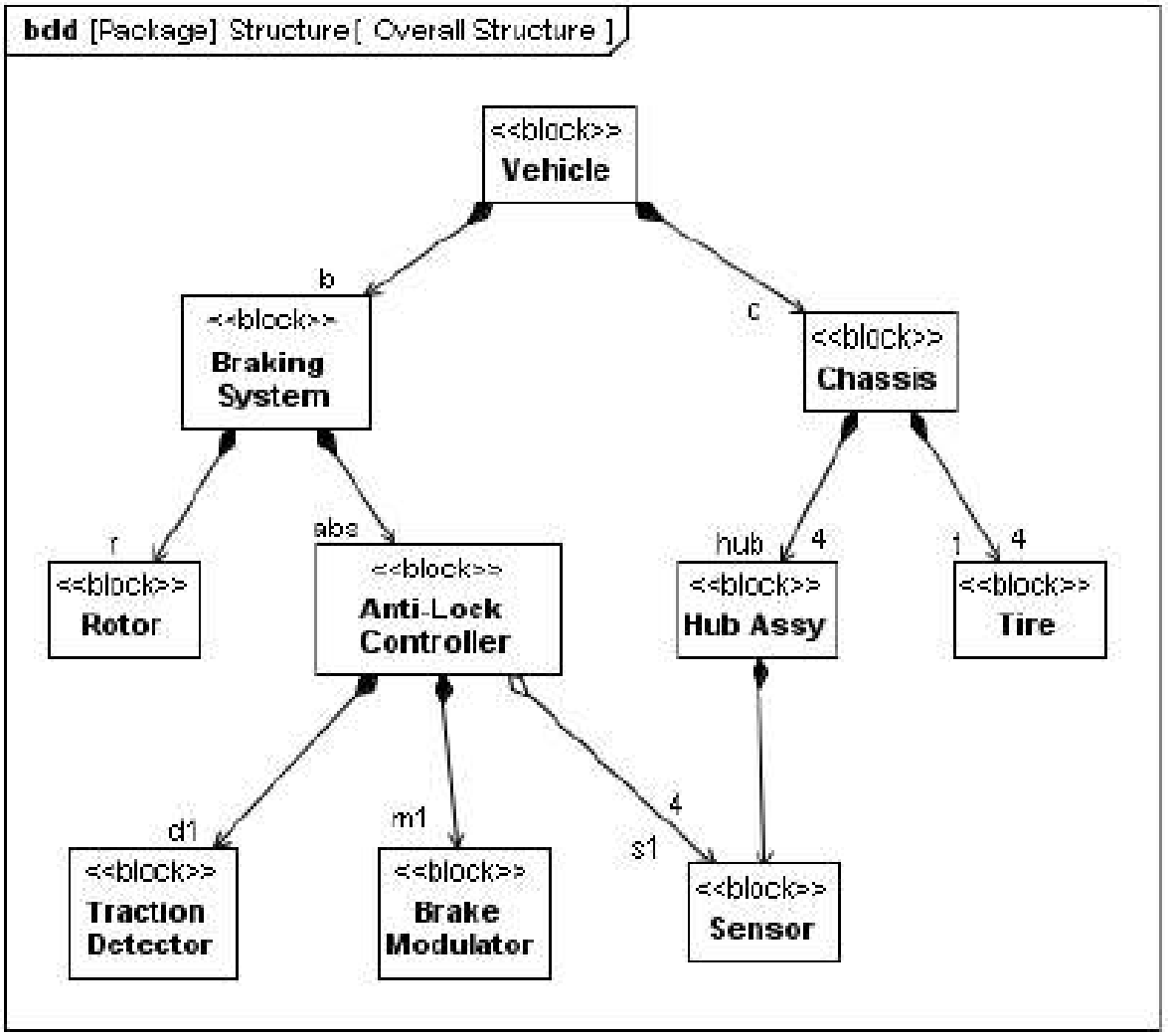
- Part is the usage of a block in the context of a block
- Also known as a role

SysML: Internal block diagram (ibd)

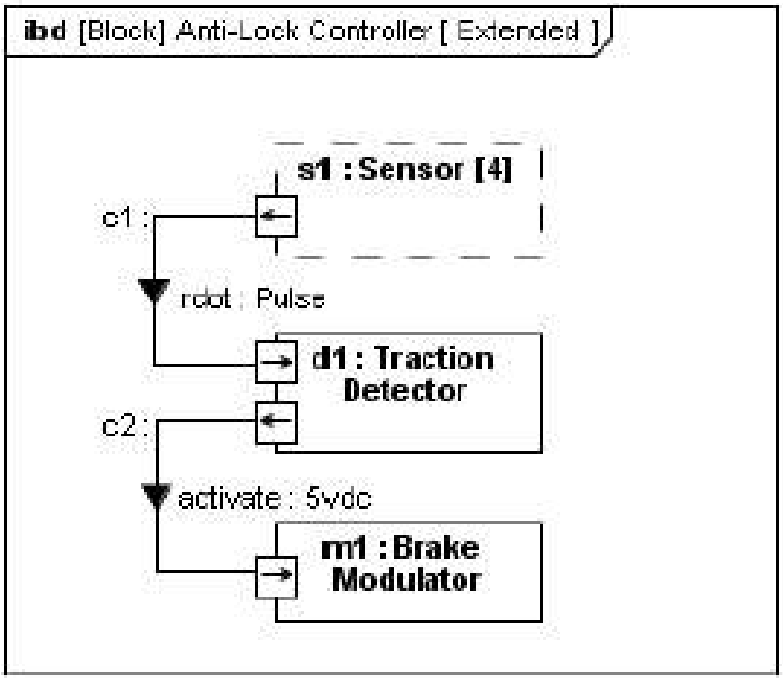


Internal Block Diagram Specifies Interconnection of Parts

SysML: Reference property



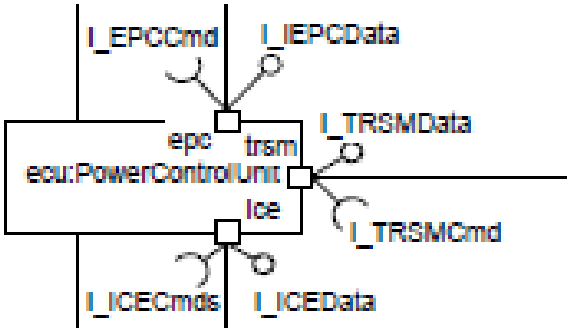
- S1 is a reference part*
- Shown in dashed outline box



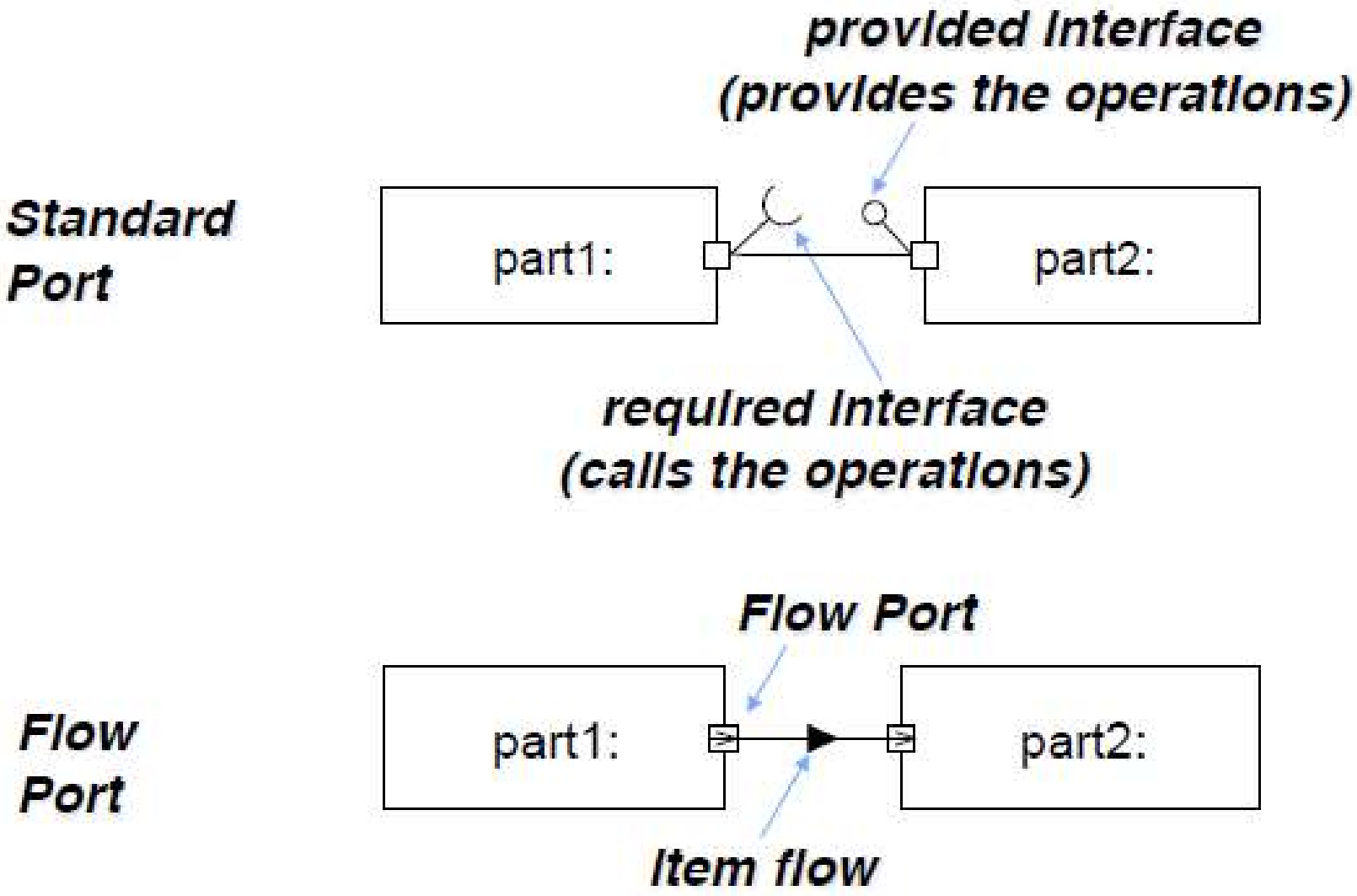
*Actual name is reference property

SysML: Ports

- Specifies interaction points on blocks and parts
  - – Integrates behavior with structure
  - – portName:TypeName
- Kinds of ports
  - – Standard (UML) Port
  - • Specifies a set of required or provided operations and/or signals
  - • Typed by a UML interface
  - – Flow Port
  - • Specifies what can flow in or out of block/part
  - • Typed by a block, value type, or flow specification
  - • Atomic, non-atomic, and conjugate variations

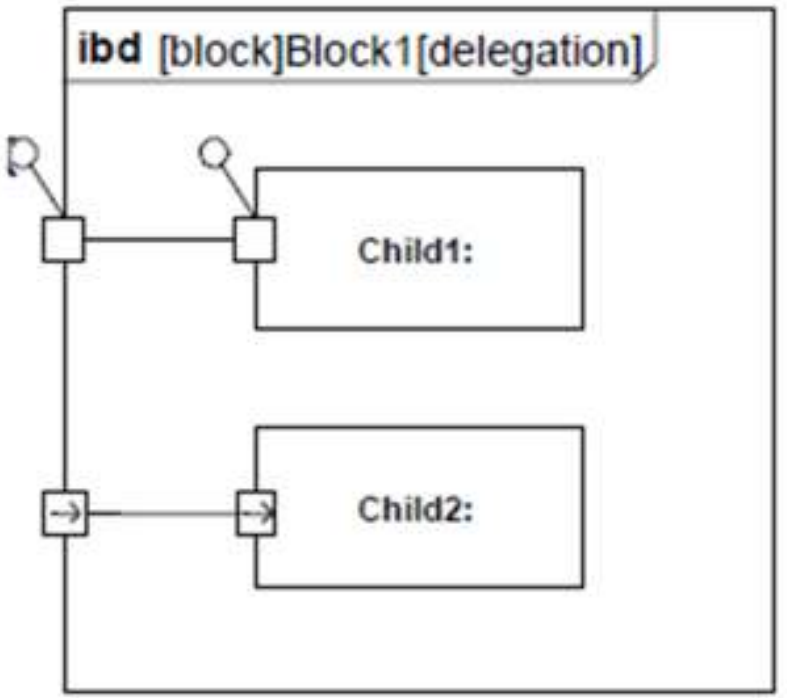


SysML: Port Notation

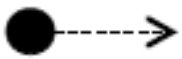


SysML: Delegation through ports

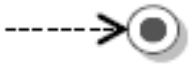
- Delegation can be used to preserve encapsulation of block (black box vs white box)
- Interactions at outer ports of Block1 are delegated to ports of child parts
- Ports must match (same kind, type, direction, etc.)
- Connectors *can* cross boundary without requiring ports at each level of nested hierarchy



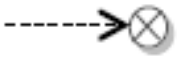
SysML: Routing Flows (activity diagrams)



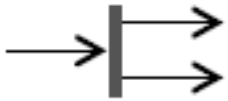
Initial Node – On execution of parent control token placed on outgoing control flows



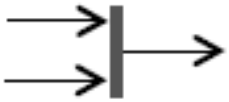
Activity Final Node – Receipt of a control token terminates parent



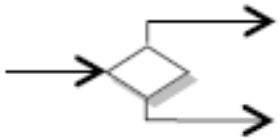
Flow Final Node – Sink for control tokens



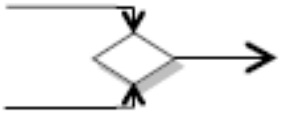
Fork Node – Duplicates input (control or object) tokens from its input flow onto all outgoing flows



Join Node – Waits for an input (control or object) token on all input flows and then places them all on the outgoing flow



Decision Node – Waits for an input (control or object) token on its input flow and places it on one outgoing flow based on guards

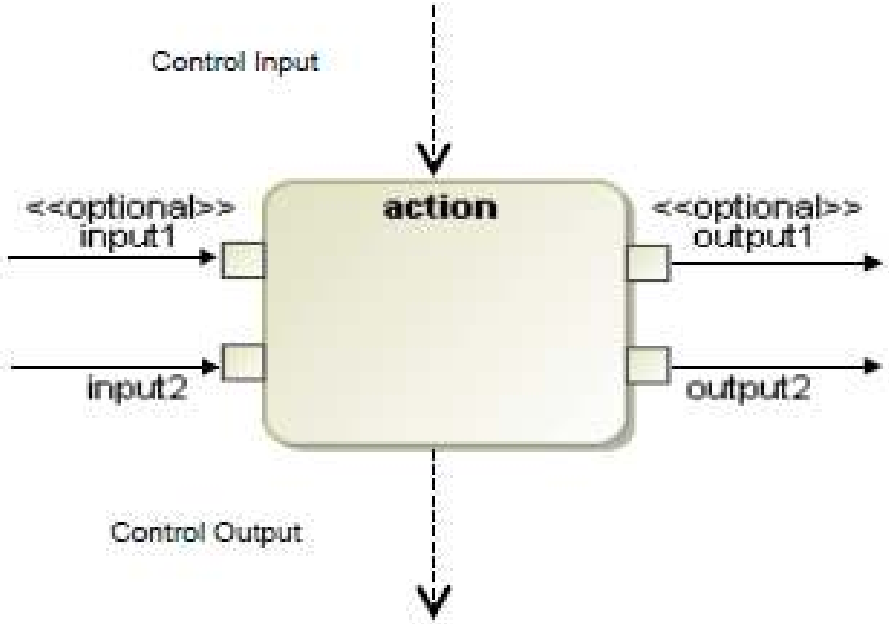


Merge Node – Waits for an input (control or object) token on any input flows and then places it on the outgoing flow

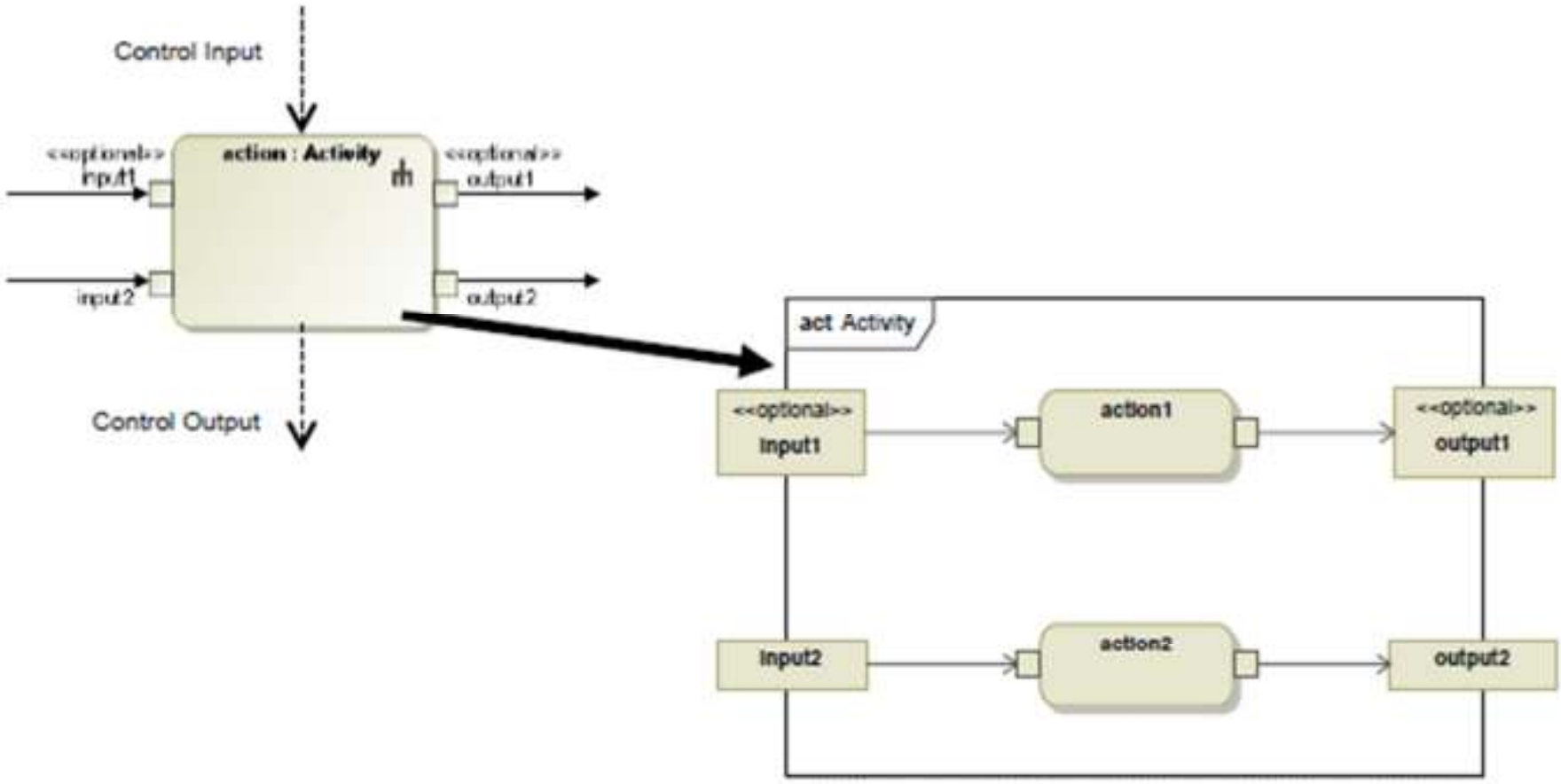
Guard expressions can be applied on all flows

SysML: Action process flows of control and data

- Two types of flows
  - – Object / Data
  - – Control
- Unit of flow is called a “token” (consumed & produced by actions)
- Actions Execution Begins When Tokens Are Available on “all” Control Inputs and Required Inputs

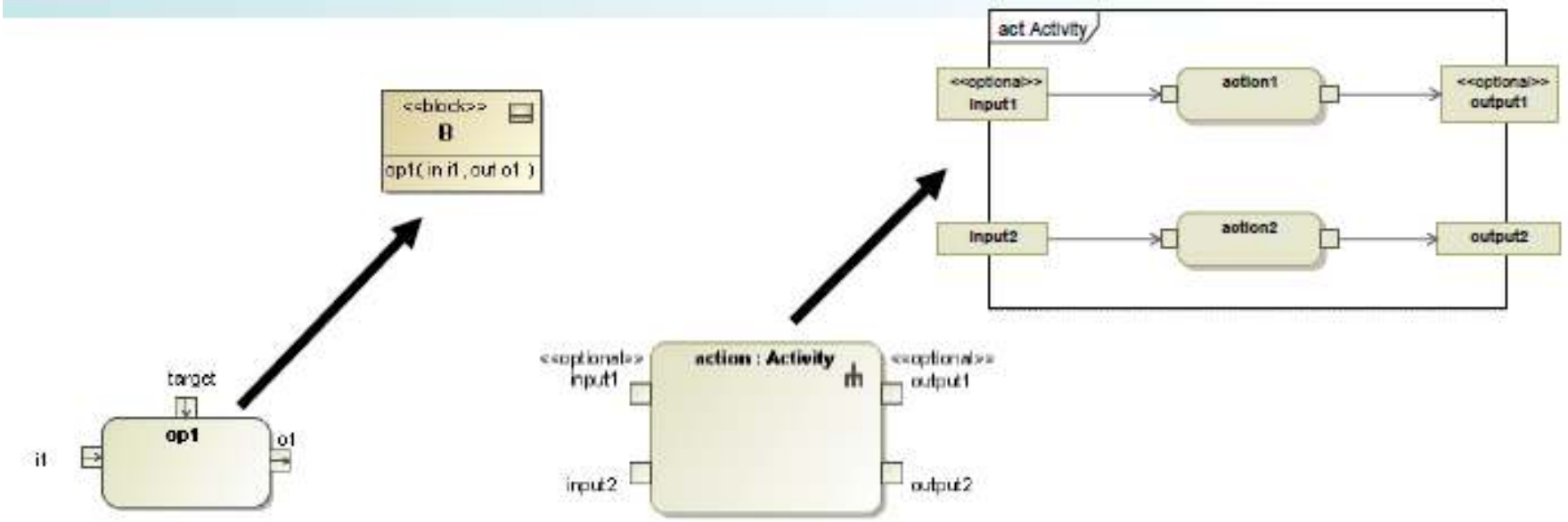


SysML: An Action can invoke another activity



Activity is Invoked When an Action Begins to Execute

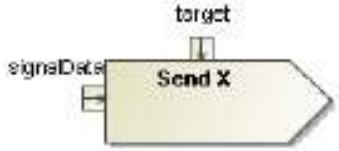
SysML: Common Actions



Call Operation Action
(can call leaf level function)

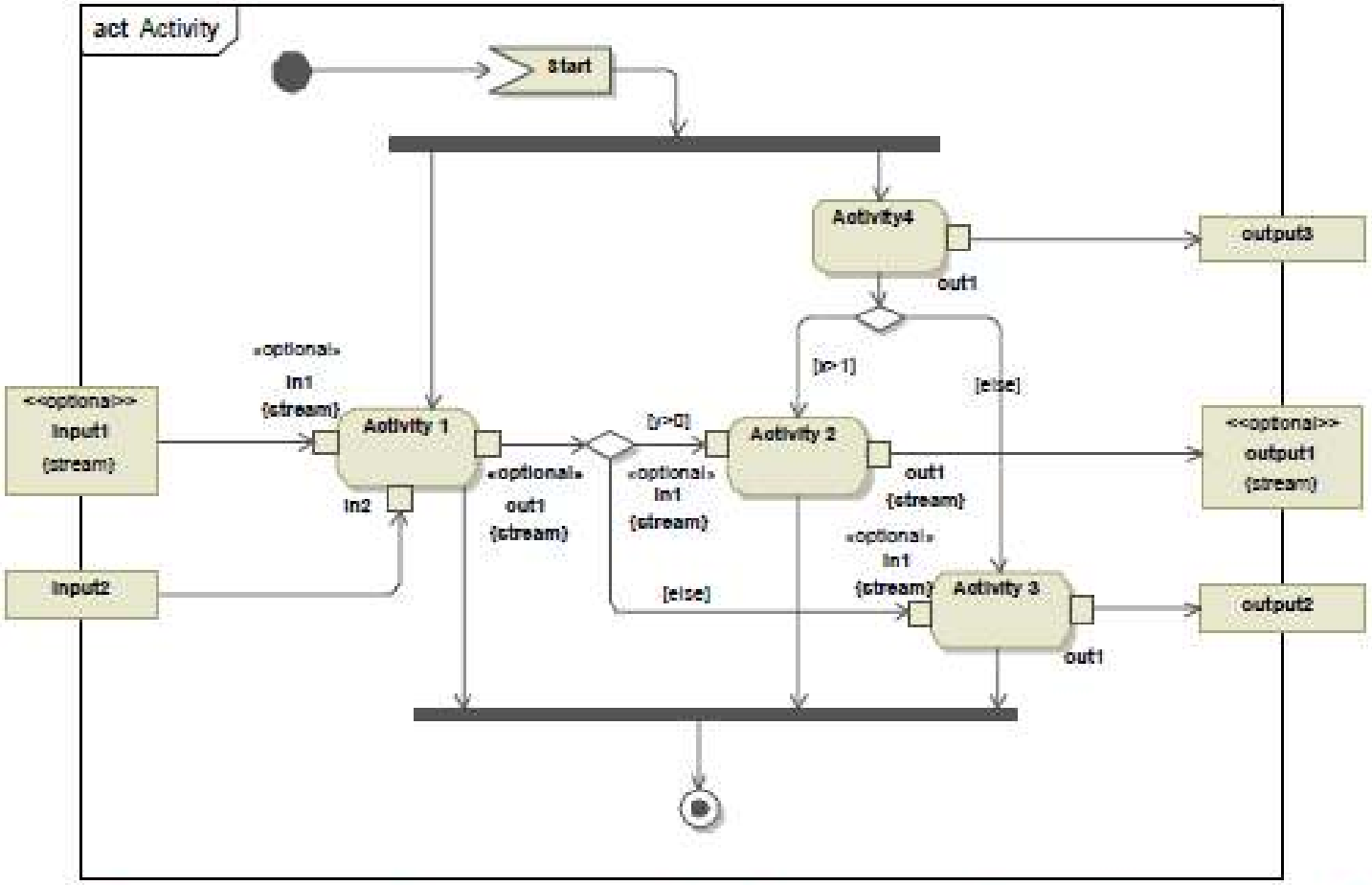
Call Behavior Action

Accept Event Action
(Event Data Pin often elided)



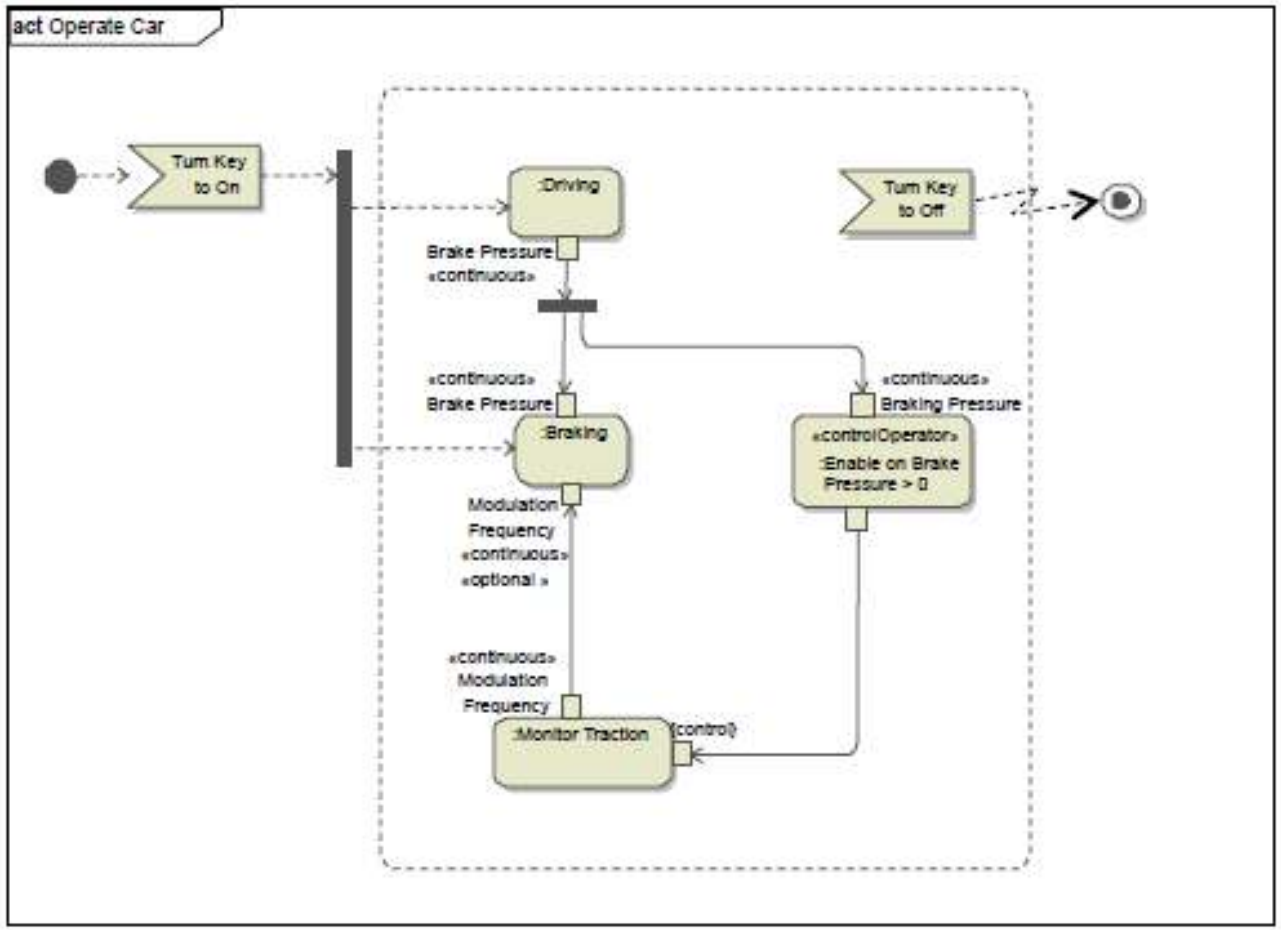
Send Signal Action
(Pins often elided)

SysML: Activity diagram example with streaming inputs and outputs



Streaming Inputs and Outputs Continue to Be Consumed and Produced While the Action is Executing

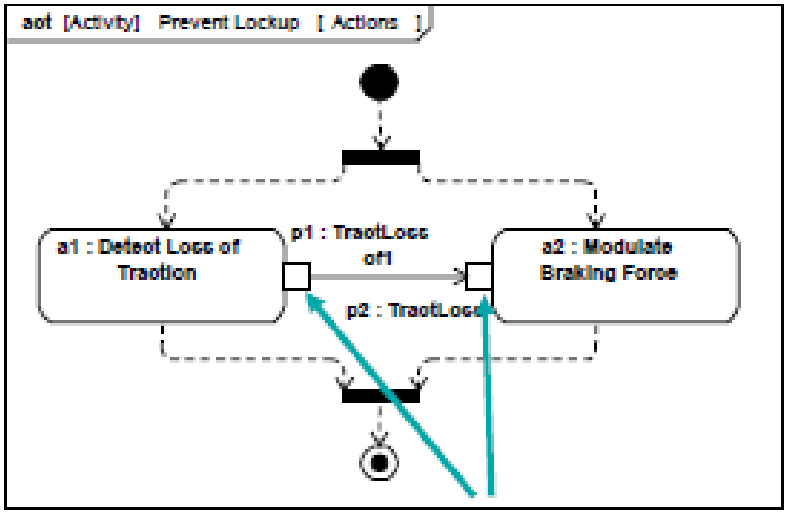
SysML: Example – Operate car



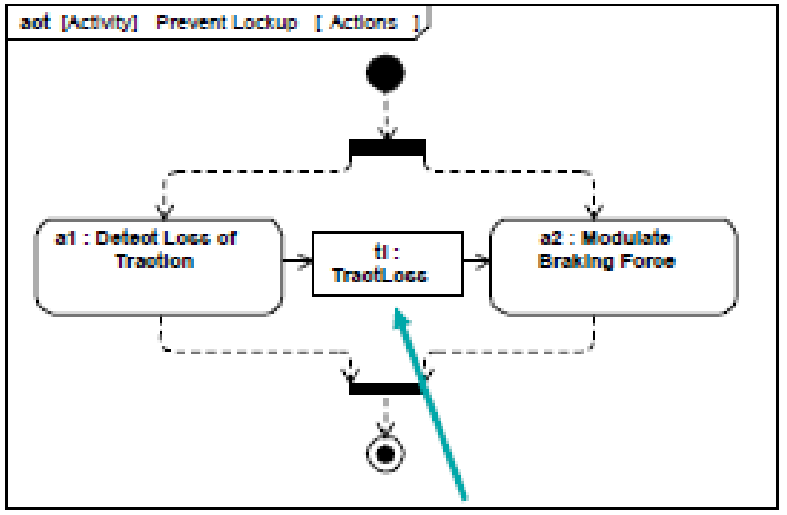
Enabling and Disabling Actions With Control Operators

SysML: Activity diagrams – pin vs. object node notation

- Pins are kinds of Object Nodes
  - – Used to specify inputs and outputs of actions
  - – Typed by a block or value type
  - – Object flows connect object nodes
- Object flows between pins have two diagrammatic forms
  - – Pins shown with object flow between them
  - – Pins elided and object node shown with flow arrows in and out



Pins

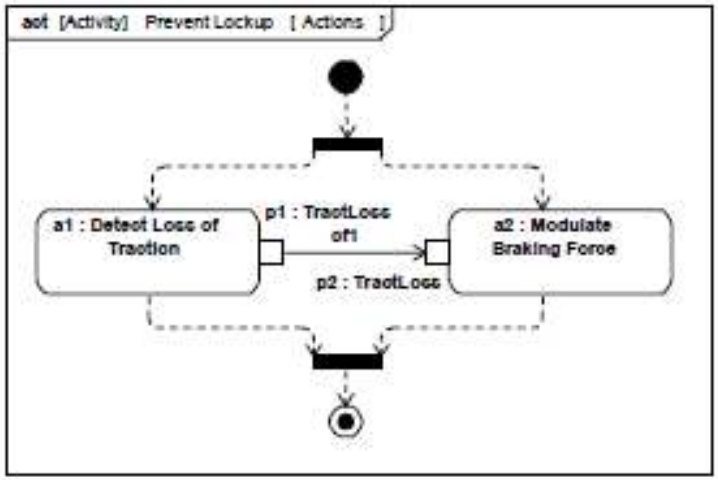


ObjectNode

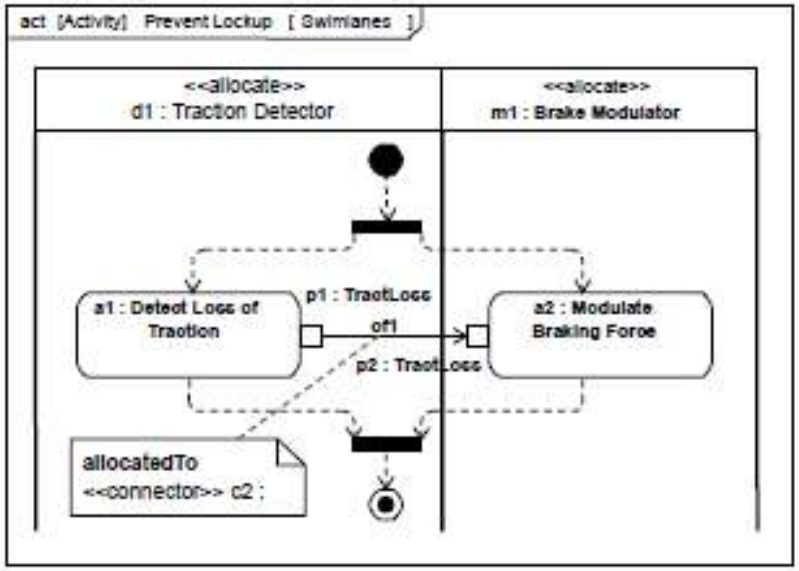
Pins must have same characteristics (name, type etc.)

SysML: Explicit allocation of behavior to structure using swimlanes

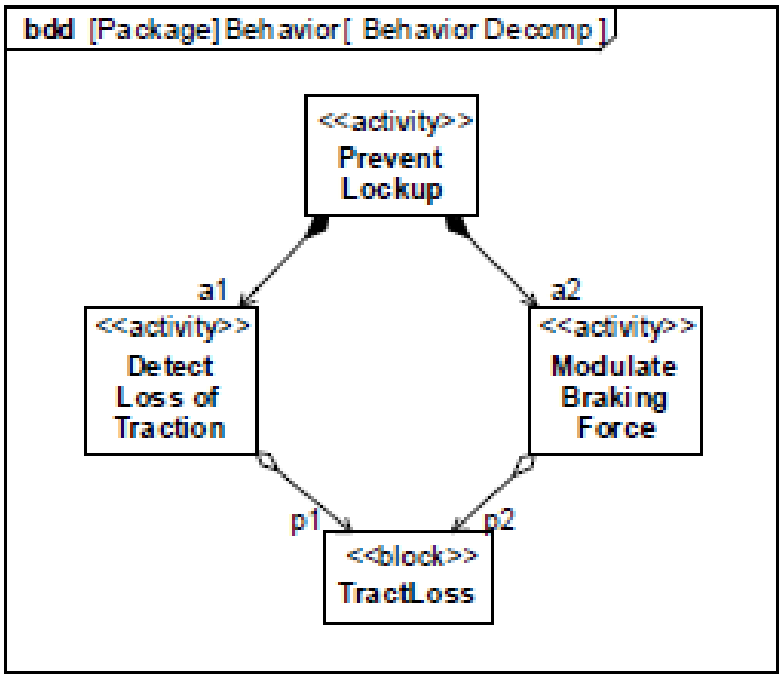
Activity Diagram (without Swimlanes)

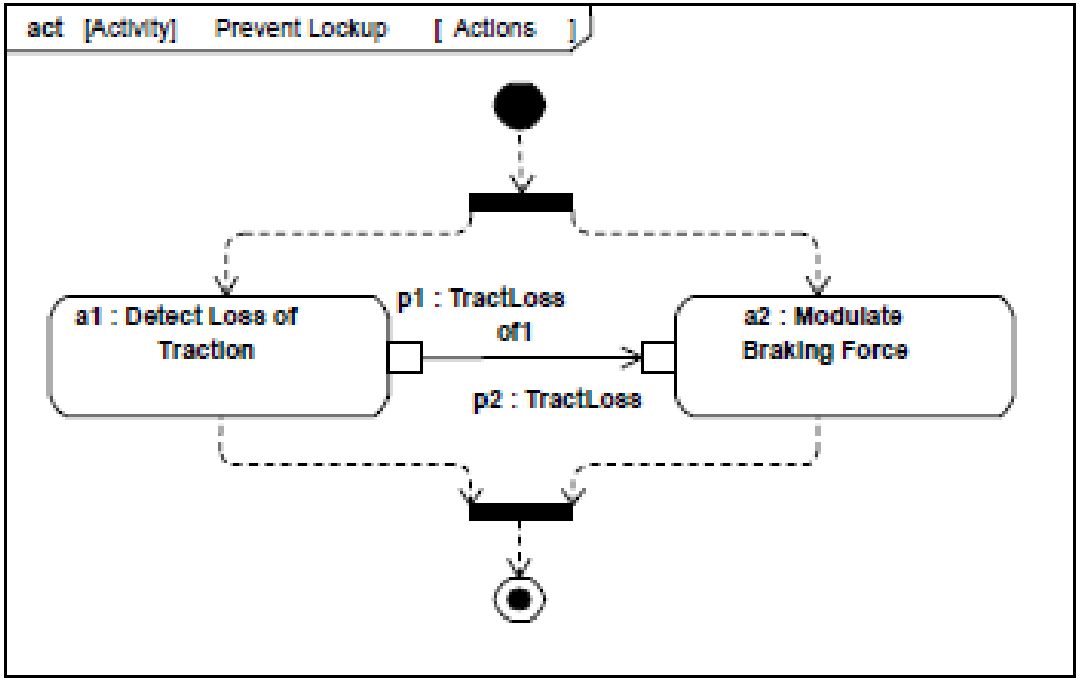


Activity Diagram (with Swimlanes)



SysML: Activity definition and use





Definition

Use

Developing SysML models: Structure, Behavior, Requirements, and Parameters.

- This will be performed through the exercises and workshop for each day.