INCOSE Healthcare Working Group

5th Annual Systems Engineering in Healthcare Conference

May 1-2, 2019
Minneapolis, MN

An Introduction to MBSE Using SysML

- Matthew Hause
- PTC Engineering Fellow
- MBSE Specialist

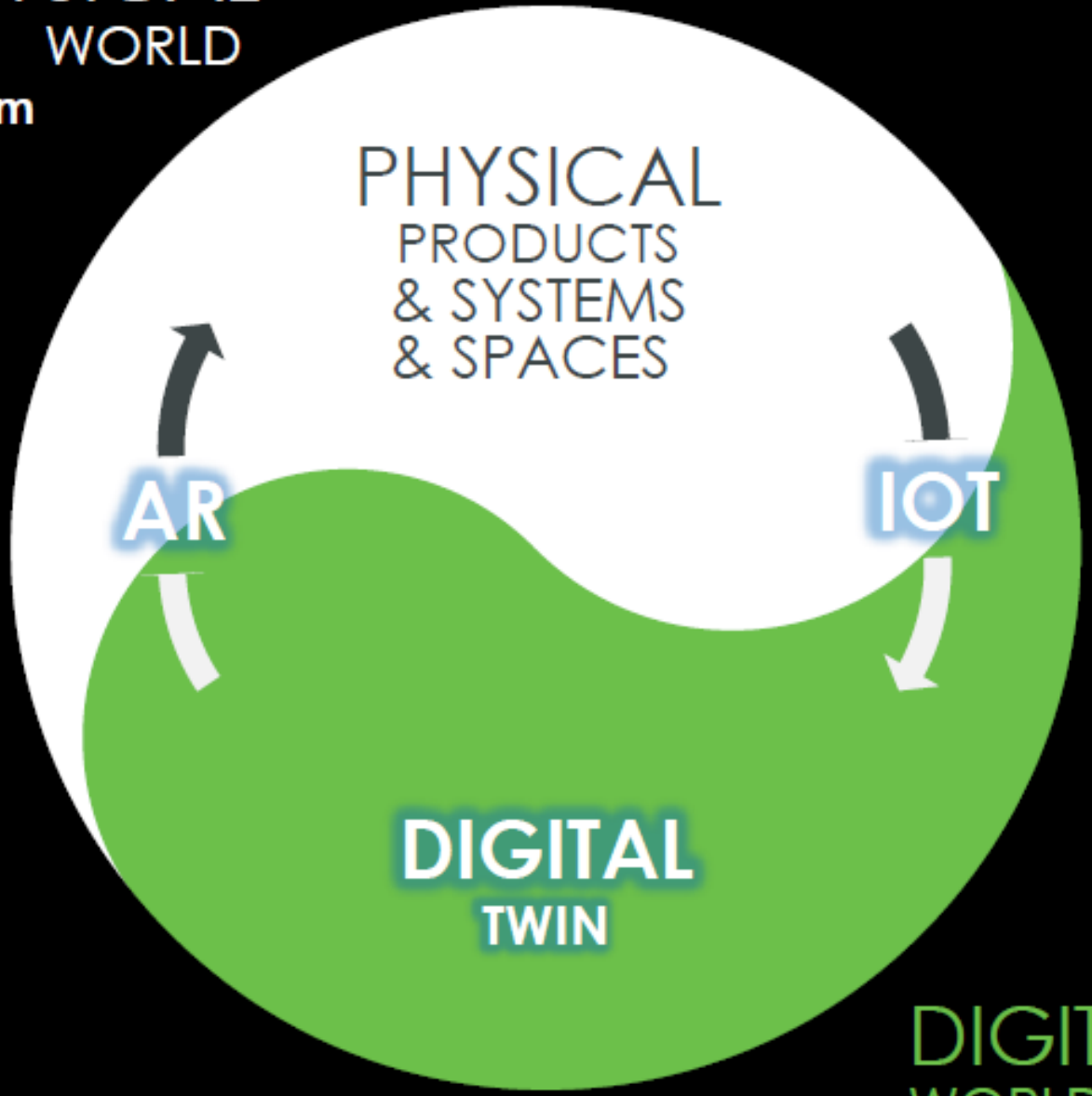
ptc

PHYSICAL WORLD

Industrial Innovation Platform

>$100M Revenue
> 50% Bookings Growth FY16
1,200 End Customers
250 OEMs/Resellers
Ecosystem of SI's, partners

IoT & ANALYTICS | thingworx | Cummins
AUGMENTED REALITY | vuforia
INDUSTRIAL CONNECTIVITY | kepware



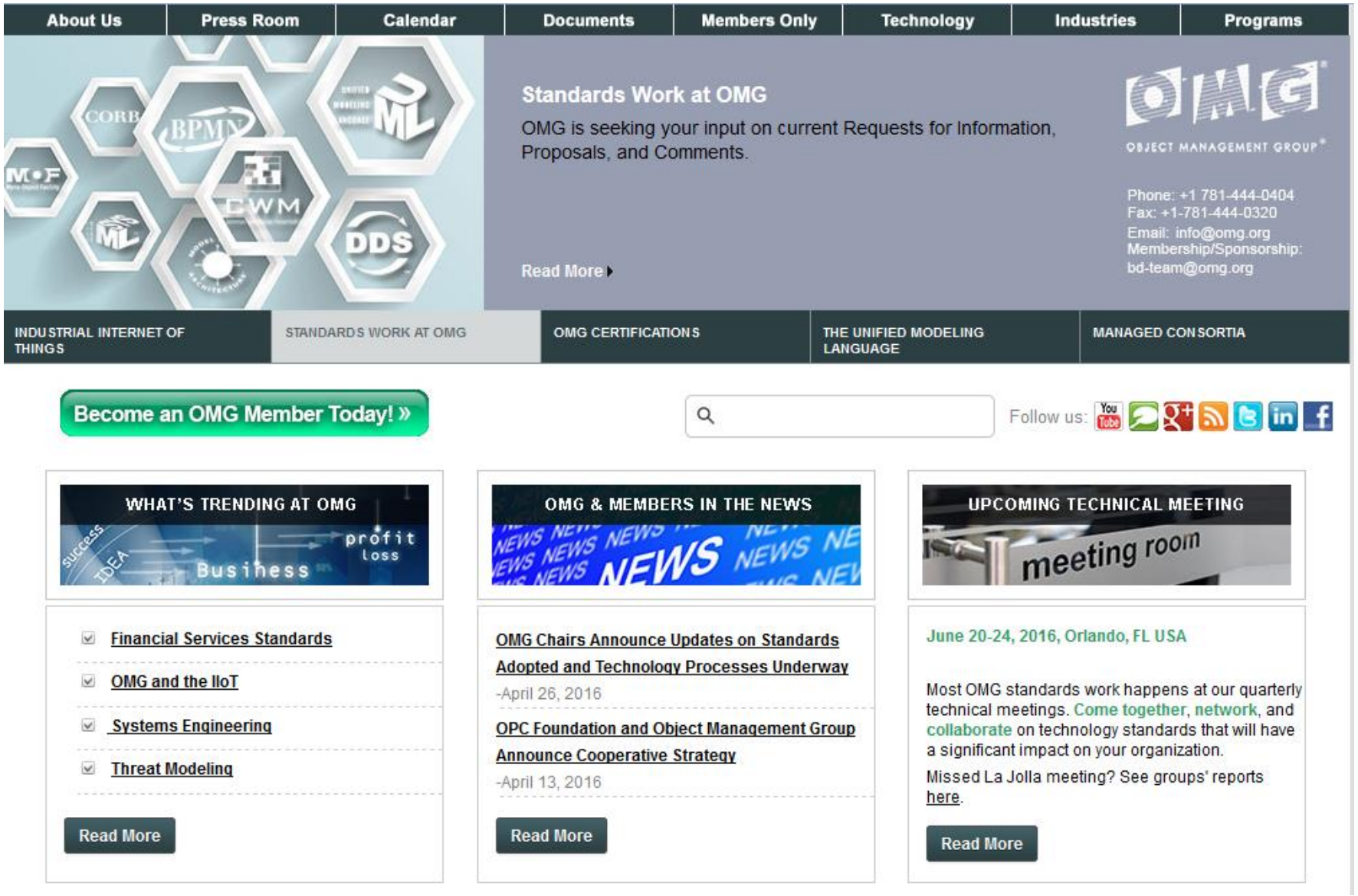
DIGITAL WORLD

PLM Solutions

>$1B Revenue
10% Bookings Growth FY16
28,000 End Customers
70% Direct Sales
30% VARs (~400)
Ecosystem of SI's, partners

CAD | creo | Cummins
PLM | windchill | Cummins
ALM | integrity | Cummins
SLM | servigistics | Cummins

OMG.ORG



- PTC on Board of Directors
- PTC on Architecture Board
- PTC Chairs & Co-chairs Task Forces & Working Groups

- PTC is also an INCOSE Corporate Advisory Board Member & Active in Chapters Worldwide

AGENDA

- Course Introduction
- SysML Overview
- Requirements Modeling (Requirements Diagram)
- Using Packages (Package Diagram)
- Views and Viewpoints
- Use Cases (Use Case Diagram)
- Activities (Activity Diagram)
- Blocks (Block Definition Diagram, Internal Block Diagram)
- Ports and Interfaces (Internal Block Diagram)
- Interactions (Sequence Diagram)
- State Machines (State Machine Diagram)
- Constraint Blocks and Parametrics (Parametric Diagram)
- Cross-cutting Constructs
  - Requirements Traceability
  - Allocations
- Product Lines
- Summary

WHAT IS A MODEL?

MODEL-BASED ENGINEERING

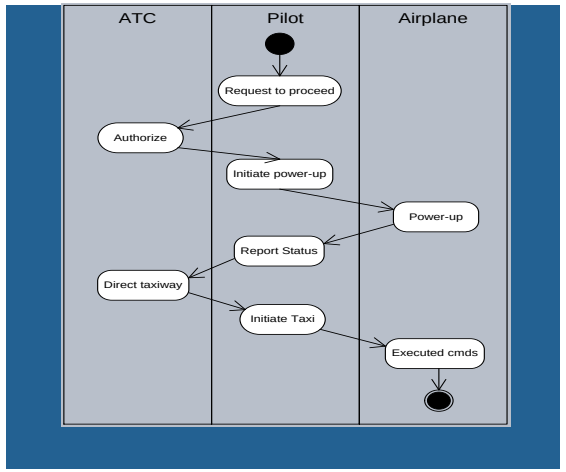
- Model-based Systems Engineering (MBSE) is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing through-out development and later lifecycle phases." (INCOSE, 2007).

- Modeling is at the heart of all aspects of the development effort
  - Covers the complete product and project lifecycle
  - Has a direct effect on any generated artifacts.
  - MBE encompasses architecture, systems and software development.

CHANGES IN SYSTEMS ENGINEERING PRACTICE

Change from Document centric to Model centric

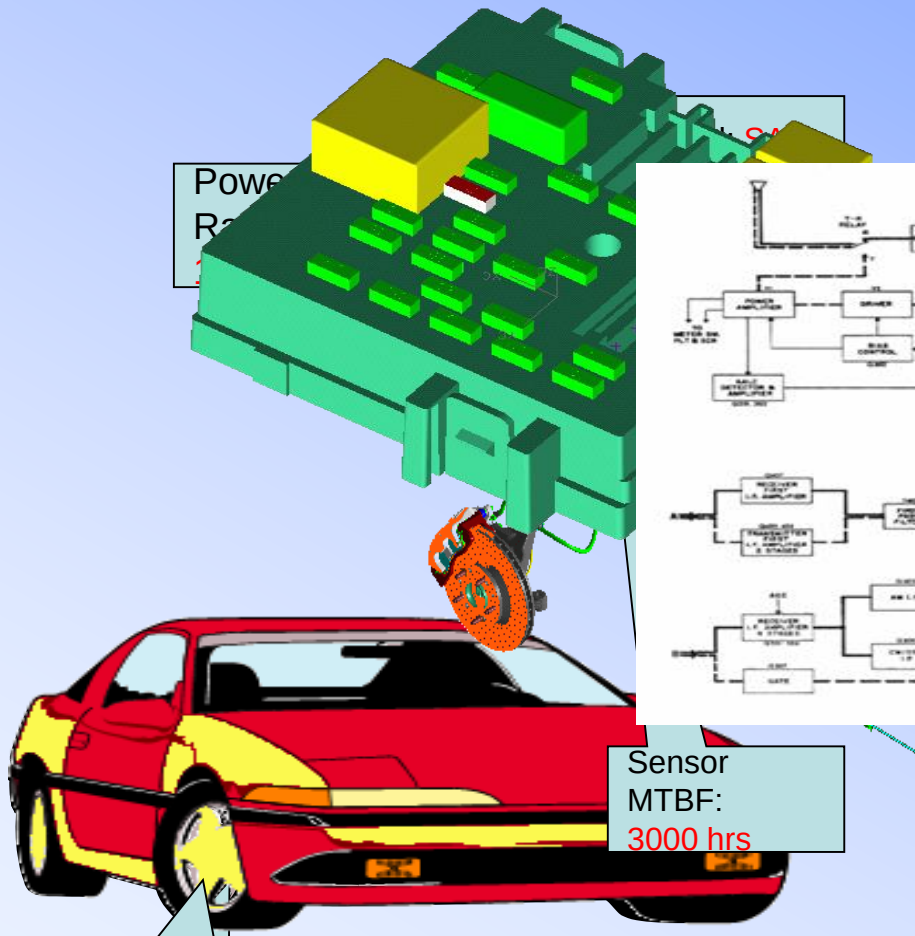
Requirement Specifications
Interface Definitions
System Architecture
System Functionality
Trade-off Analysis
Test Specifications
Etc.

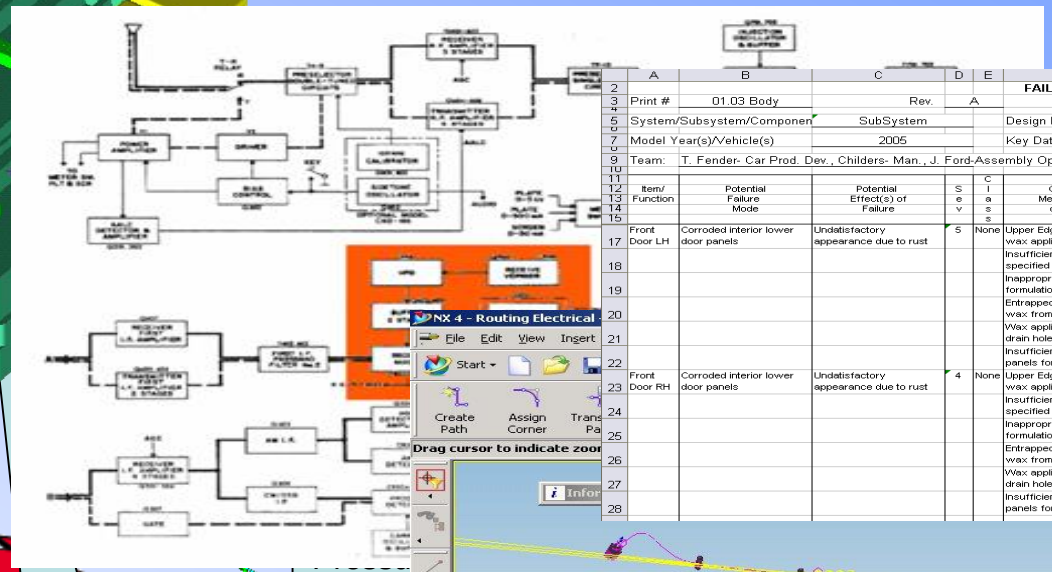


Old Approach

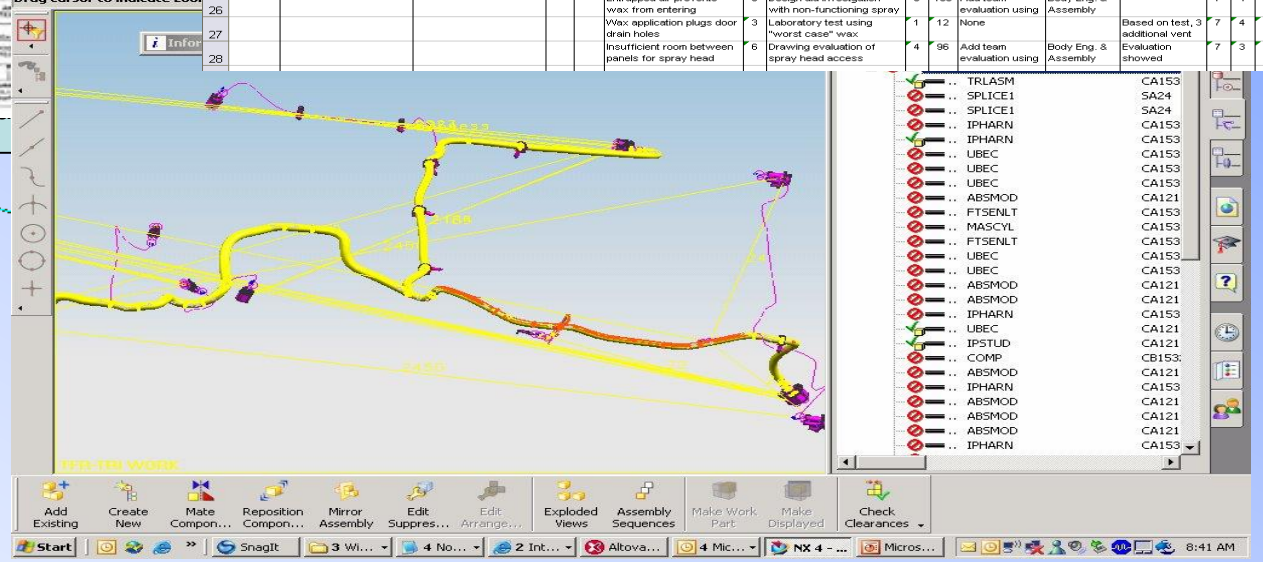
New Approach

Integrated Systems Engineering Vision





	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
2						FAILURE MODE AND EFFECTS ANALYSIS											
3	Print #	01.03 Body		Rev.	A	(DESIGN FMEA)						FMEA Number:	1234				
5	System/Subsystem/Componen		SubSystem			Design Responsibility:		Body Engineering				Prepared by:	J. Ford-Assembly Opps				
7	Model Year(s)/Vehicle(s)		2005			Key Date		9/3/04				Date (Orig.)	8/3/04				
9	Team:	T. Fender- Car Prod. Dev., Childers- Man., J. Ford-Assembly Opps										Date (Rev.)	8/22/04				
12	Item/ Function	Potential Failure Mode	Potential Effect(s) of Failure	S e v	C l a s s	Potential Cause(s)/ Mechanism(s) of Failure	O c c u r	Current Design Controls	D e t e c	R. P. N.	Recommended Actions	Responsibility & Target Date	Action Results Actions Taken	S e v	O c c	D e t	R. P. N.
17	Front Door LH	Corroded interior lower door panels	Undatisfactory appearance due to rust	5	None	Upper Edge of protective wax application specified	2	Vehicle general durability test T-118 T-109 T-301	8	80	Add laboratory accelerated	A. Tate-Body Eng 8/9/04	Based on test results upper	4	2	3	24
18						Insufficient was thickness specified	4	Vehicle general durability testing as above	8	160	Added laboratory	Combine w/test for	Test results (Test No. 1481)	4	1	2	8
19						Inappropriate wax formulation specified	2	Physical and chemical lab test report No. 1265	2	20	None			3	1	4	12
20						Entrapped air prevents wax from entering	5	Design aid investgation with non-functioning spray	8	200	Add team evaluation using	Body Eng. & Assembly		7	2	2	28
21						Wax application plugs door drain holes	3	Laboratory test using "worst case" wax	1	15	None		Based on test, 3 additional vent	7	2	2	28
22						Insufficient room between panels for spray head	4	Drawing evaluation of spray head access	5	100	Add team evaluation using	Body Eng. & Assembly	Evaluation showed	7	3	2	42
23	Front Door RH	Corroded interior lower door panels	Undatisfactory appearance due to rust	4	None	Upper Edge of protective wax application specified	5	Vehicle general durability test T-118 T-109 T-301	8	160	Add laboratory accelerated	A. Tate-Body Eng 8/9/04	Based on test results upper	3	2	3	18
24						Insufficient was thickness specified	4	Vehicle general durability testing as above	3	48	Added laboratory	Combine w/test for	Test results (Test No. 1481)	7	1	4	28
25						Inappropriate wax formulation specified	2	Physical and chemical lab test report No. 1265	2	16	None			7	6	3	126
26						Entrapped air prevents wax from entering	5	Design aid investgation with non-functioning spray	5	100	Add team evaluation using	Body Eng. & Assembly		7	4	1	28
27						Wax application plugs door drain holes	3	Laboratory test using "worst case" wax	1	12	None		Based on test, 3 additional vent	7	4	2	56
28						Insufficient room between panels for spray head	6	Drawing evaluation of spray head access	4	96	Add team evaluation using	Body Eng. & Assembly	Evaluation showed	7	3	2	42



Sensor MTBF: 3000 hrs

Minimum Turn Radius: 24 ft.
Dry Pavement Braking Distance at 60 MPH:
110 ft. 90 ft

THE FUTURE OF MBSE: SMART AND CONNECTED

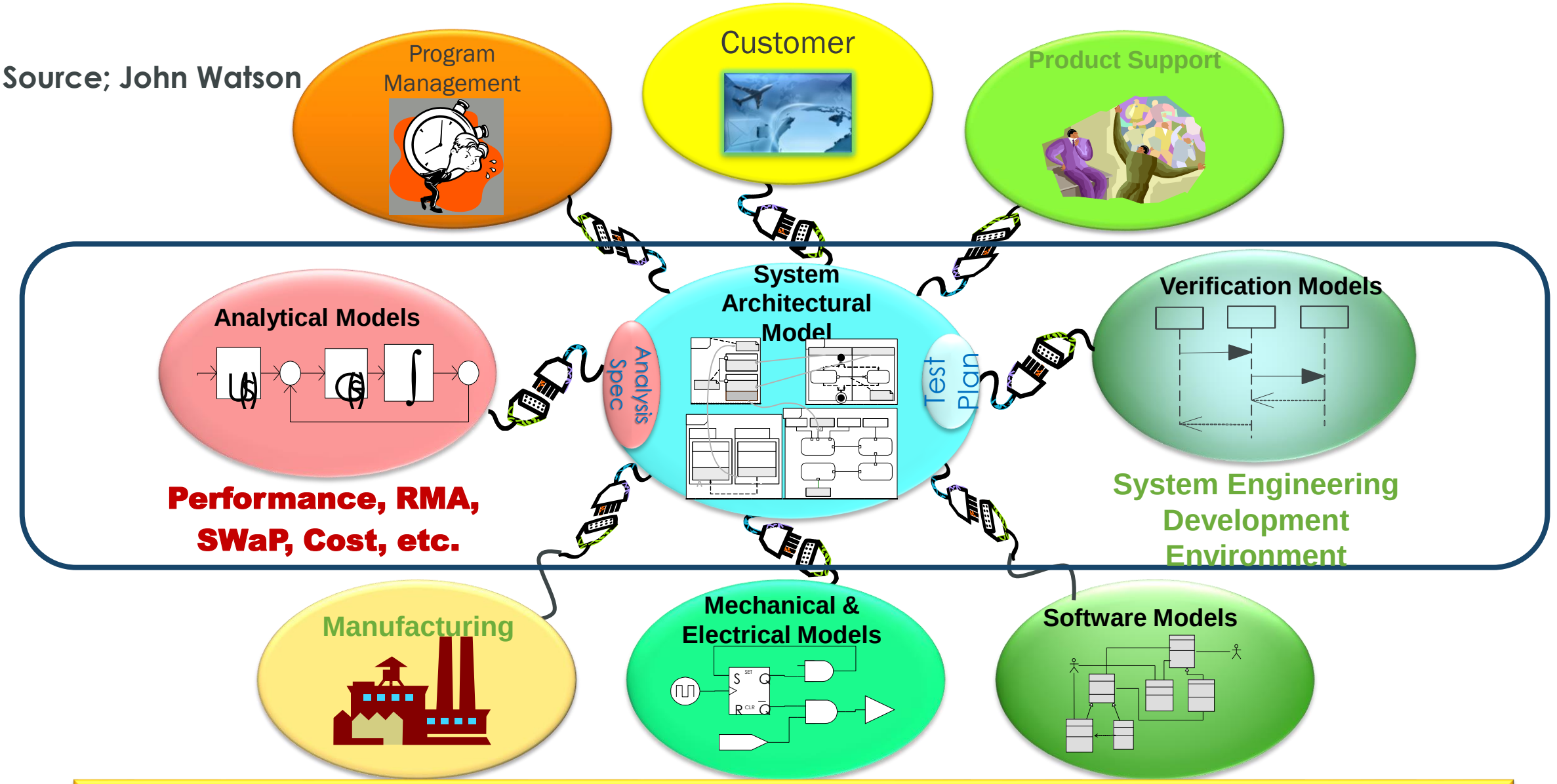
- Systems Engineering requires a multidisciplinary approach:
  - At all levels of the architecture
  - Throughout the development lifecycle

- Smart
  - Enables quantitative and qualitative analysis
  - FMEA, Risk, Safety, Human Factors, PLM, Mechanical Engineering, Electrical Engineering, etc.
  - Data driven rather than diagram driven or document driven

- Connected
  - To virtual teams
  - As part of an integrated tool chain
  - Providing data when and where it is needed
  - Supporting openness as well as security

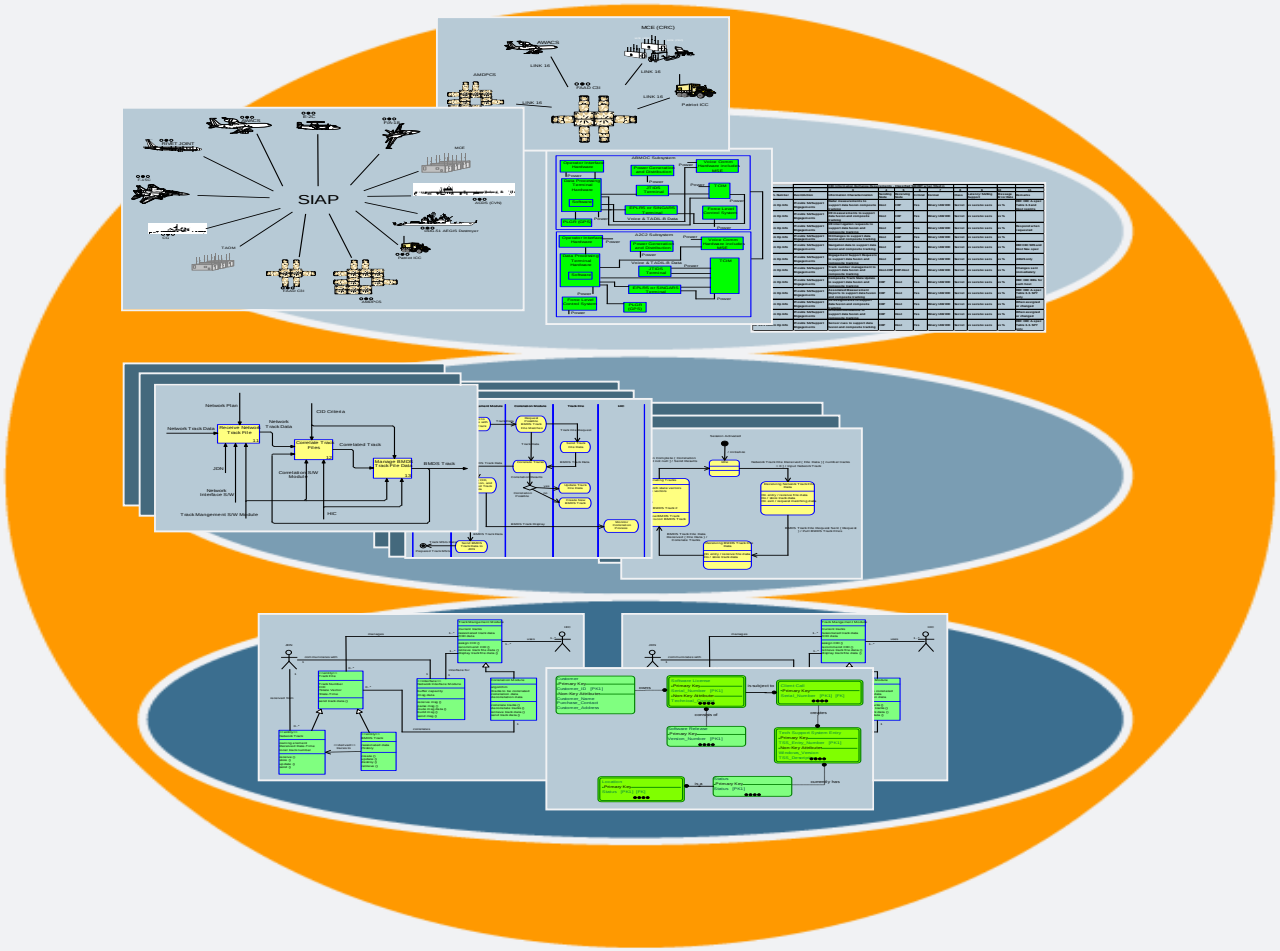
EVOLVING MBSE USE CASES

Source; John Watson



To measure MBSE effectiveness we need to understand the context of how it is used

MODELING AT MULTIPLE LEVELS OF THE SYSTEM

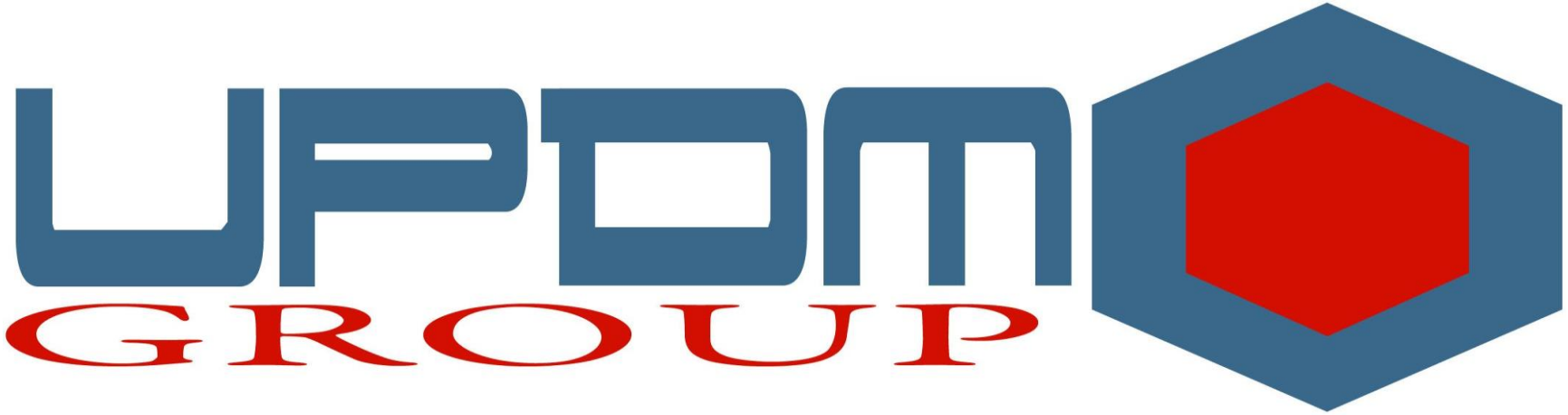


Architecture Models

Systems Models

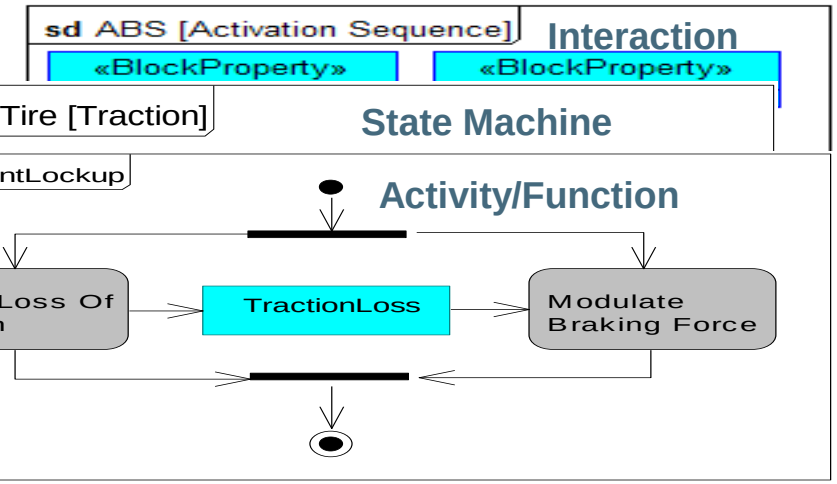
Component Models

Modeling Language for these Multiple Levels of the System

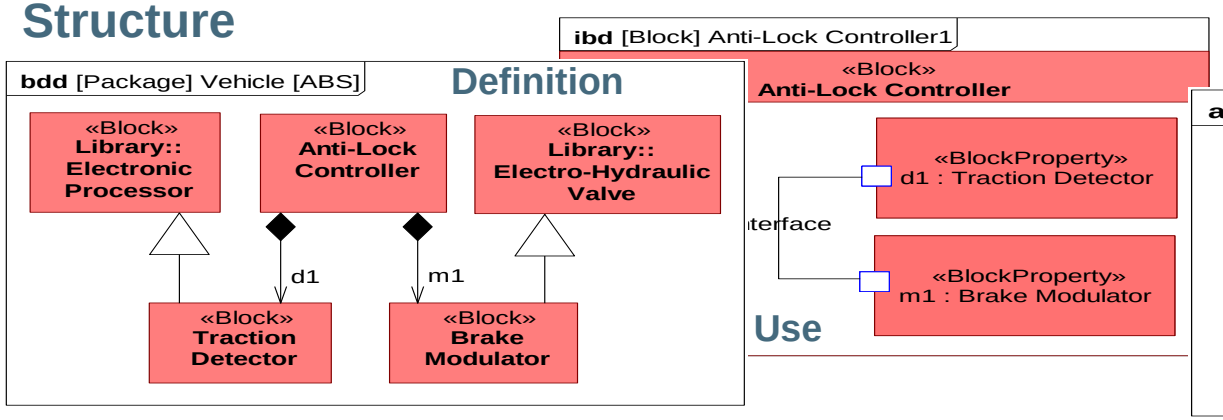


THE FOUR PILLARS OF SYSML

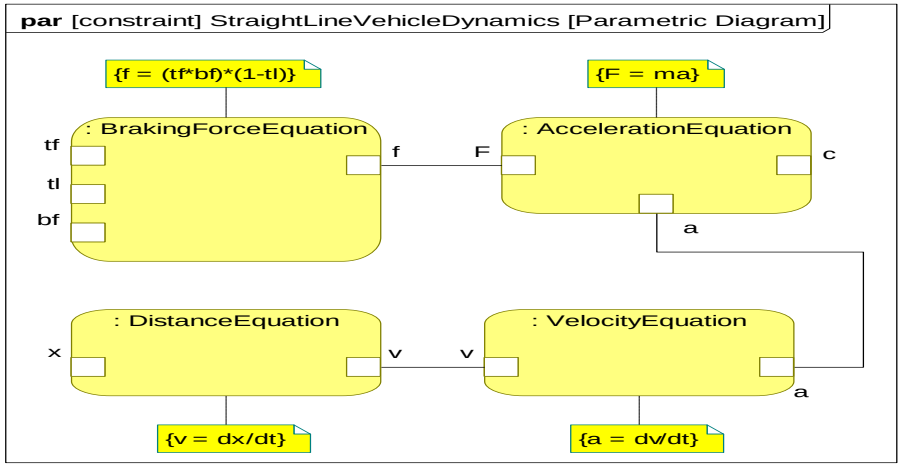
Behavior



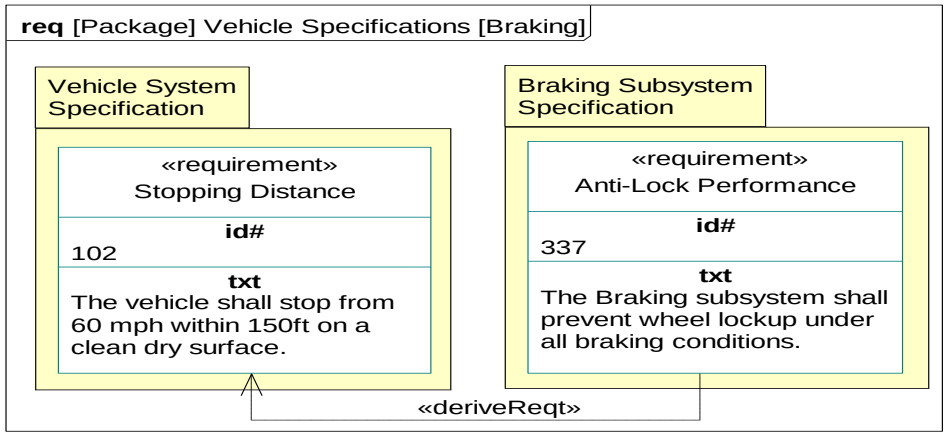
Structure



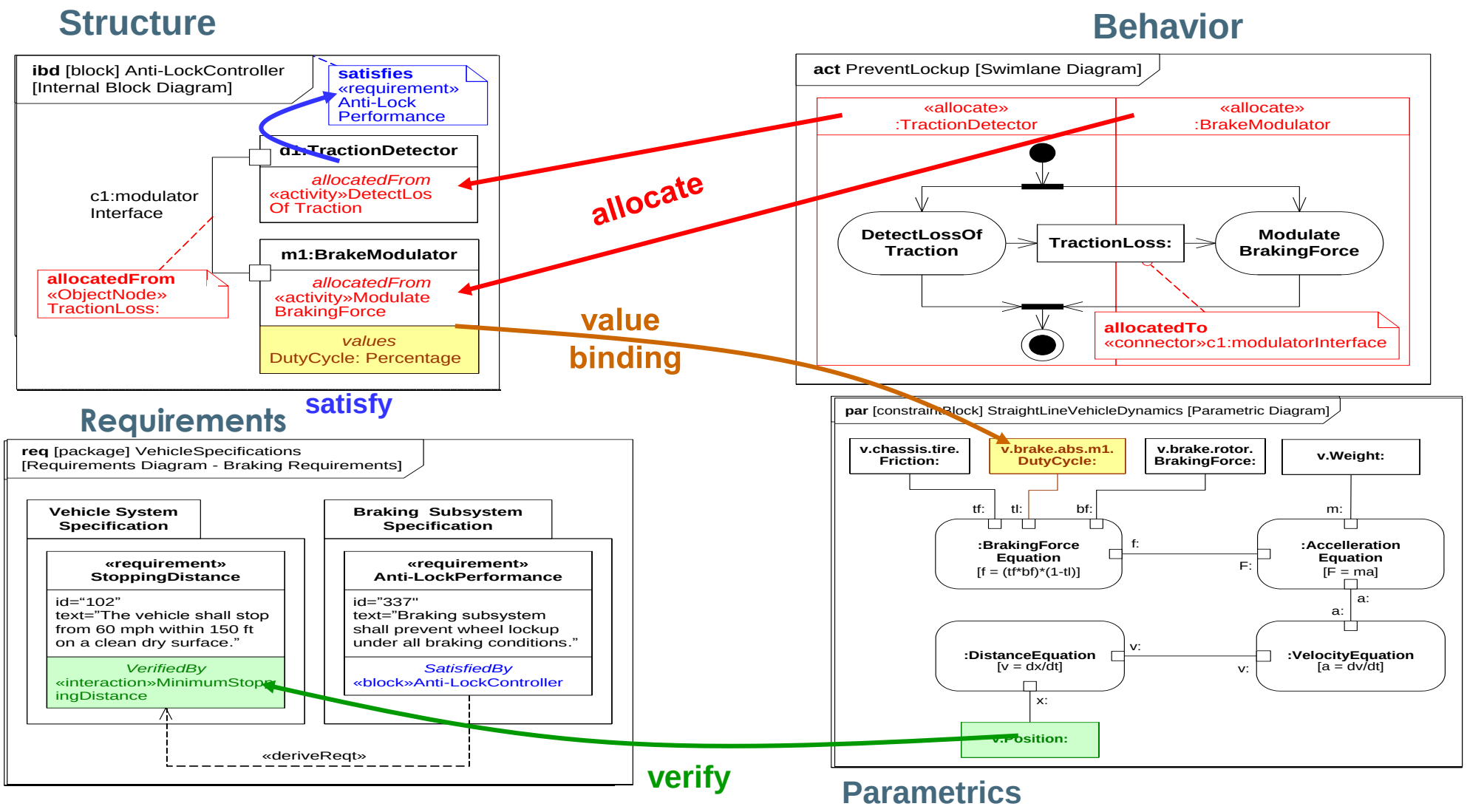
Parametrics



Requirements



CROSS CONNECTING MODEL ELEMENTS



MBSE INTEROPERABILITY

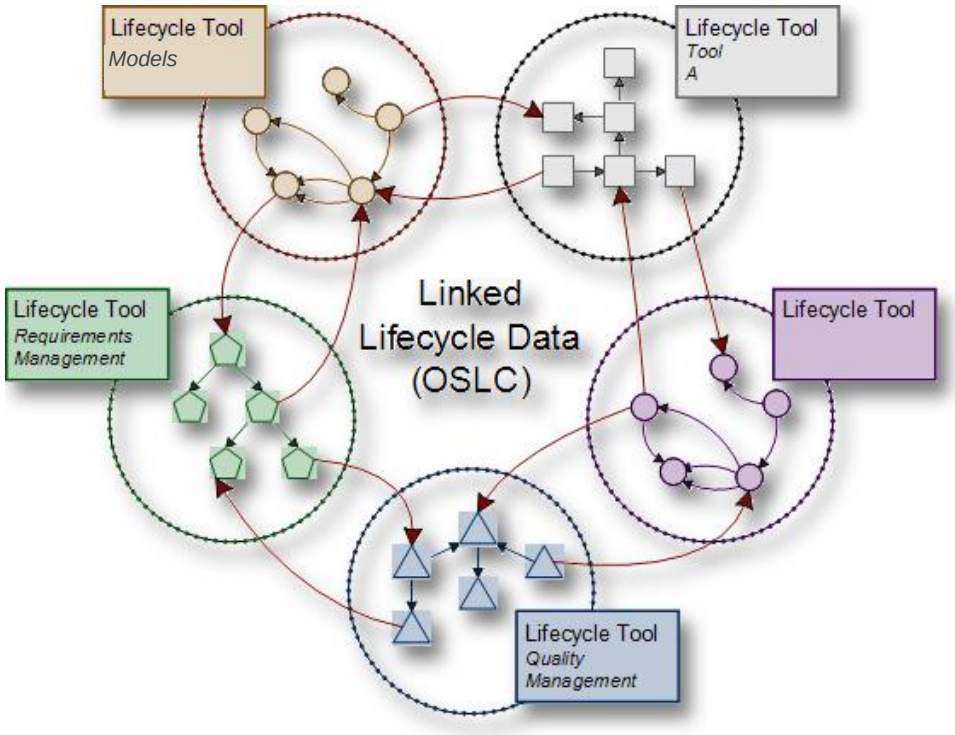
OSLC OVERVIEW

- OASIS is an international non-profit consortium that brings companies, governments, academia, and individuals together to define industry standards, including OSLC
- OSLC is a set of standards that make it easy and practical for lifecycle software tools to share data with one another
- OSLC allows you to more easily & effectively integrate disparate tool and enable end-to-end processes that cover the entire product development lifecycle

http://open-services.net

OSLC – KEY PRINCIPLES

- OSLC focuses on the *interaction* between systems, not the internal behavior of the systems themselves
- OSLC integrations are loosely coupled
  - data is linked, not synchronized or replicated
- OSLC uses the notion of upstream and downstream systems and data
- Data items called ‘Resources’ and have ‘Unique Resource Identifiers’ (URIs)
- OSLC Links are defined as ‘URI References from one Resource to another Resource’
- OSLC systems may have Resource Providers and/or Resource Consumers



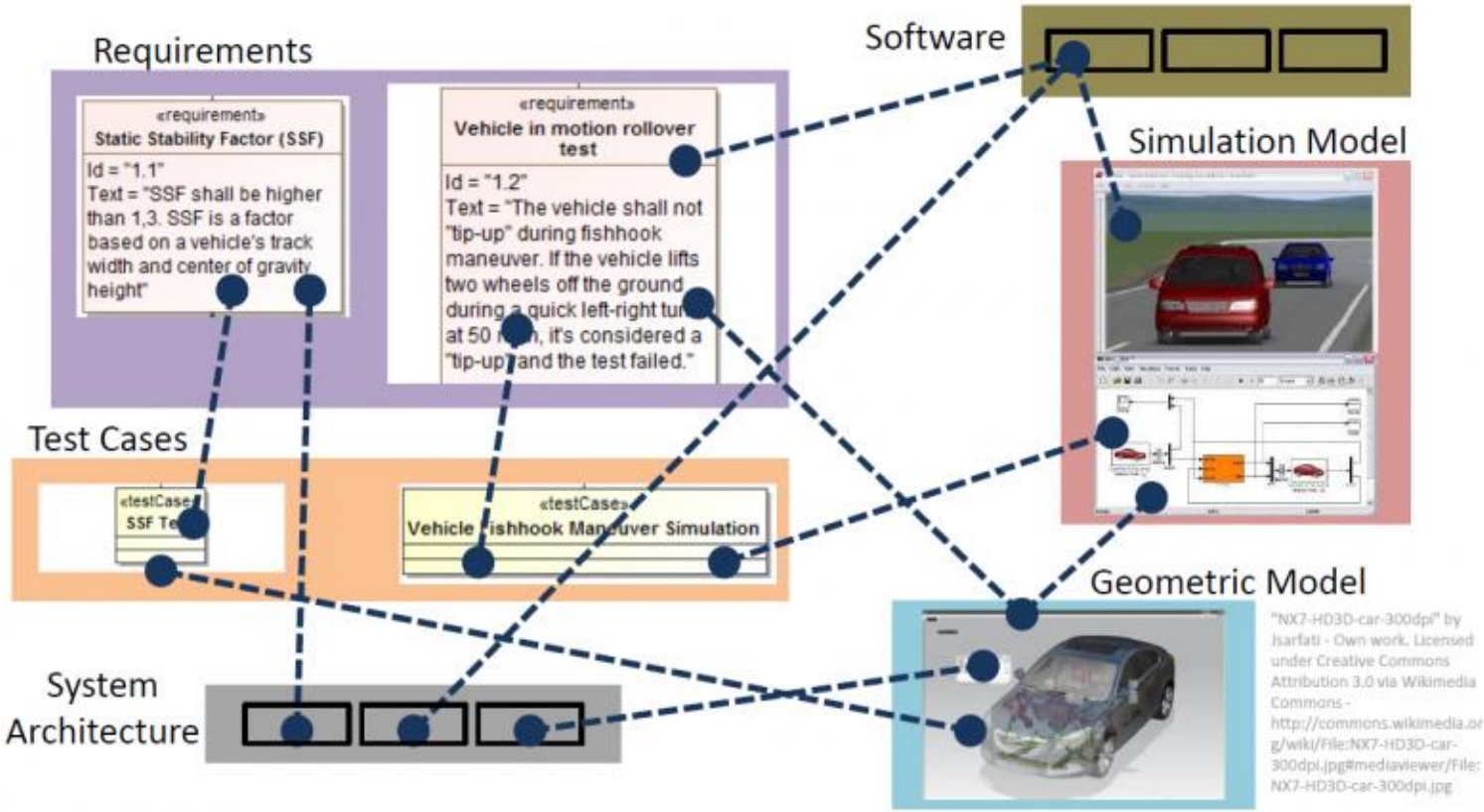
Domain specific OSLC Specifications

SYSML 2.0 INTEROPERABILITY WORKING GROUP

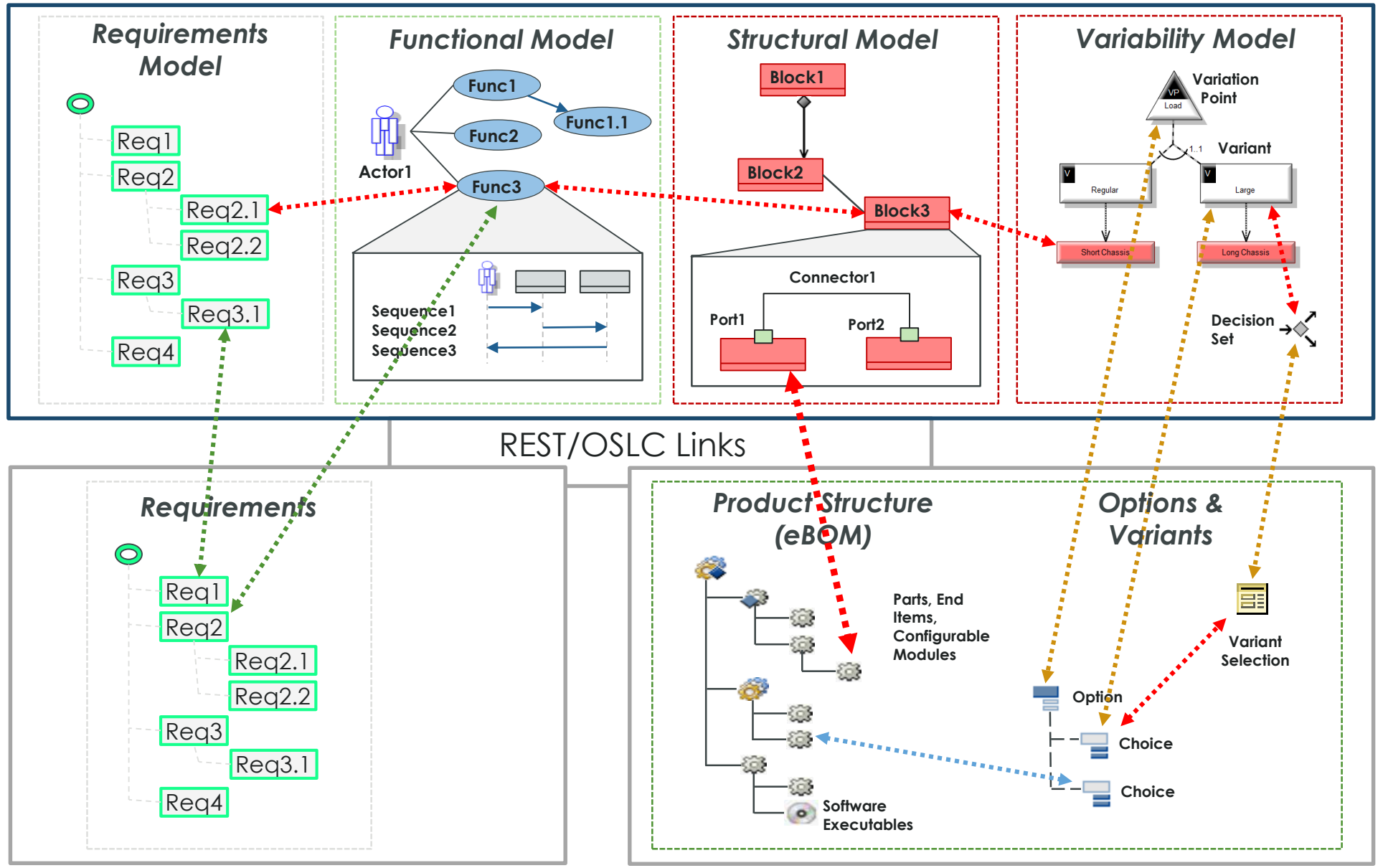
- **Axel Reichwein et al**

- Objective; Identify requirements related to interoperability such that SysML can support standards for data interoperability, traceability, analytics in a multi-disciplinary engineering context

- The next-generation modeling language must provide a standard (i.e. OSLC) application programming interface (API) to provide dynamic access to the model, while providing appropriate access controls



MBSE, ALM & PLM DIGITAL PRODUCT TRACEABILITY

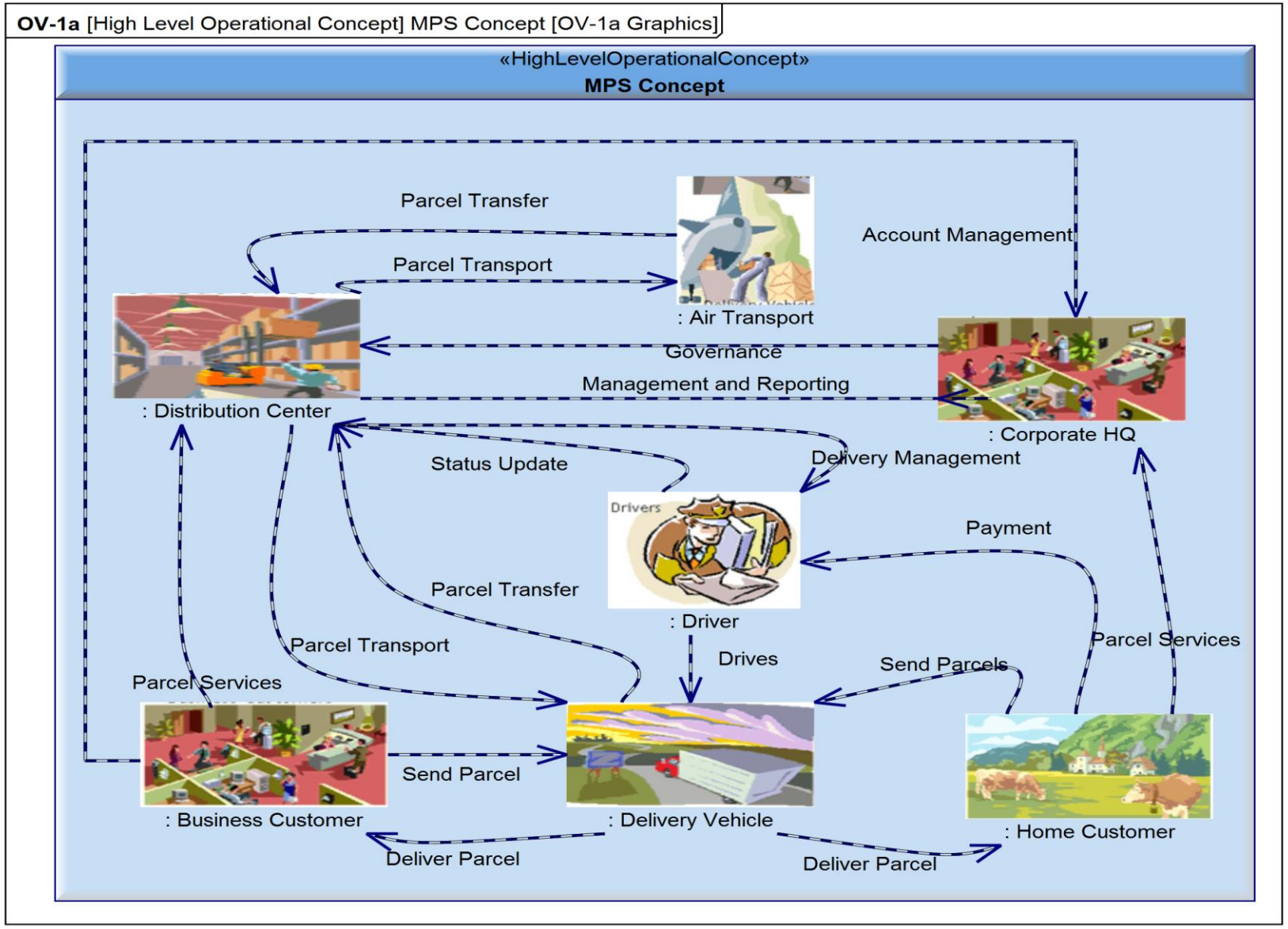


THE EXAMPLE PROBLEM

THE MARVELOUS PARCEL SYSTEM

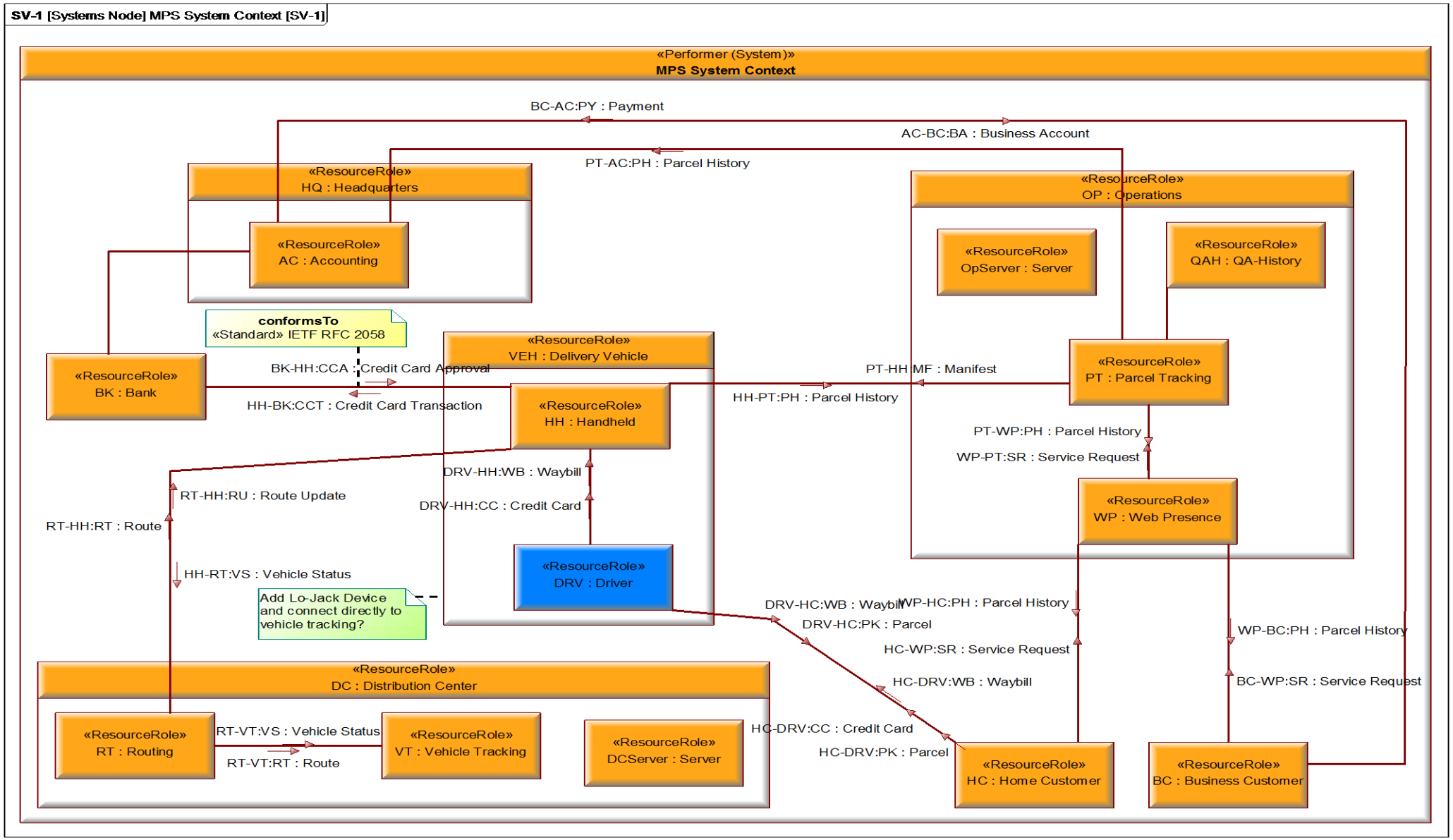
- Concept of Operations (CONOPS)
- The Marvelous Parcel Service (MPS) is replacing their nationwide manual, paper-based parcel tracking with automated capability. MPS consists of:
  - Corporate Headquarters carrying out standard business functions
  - Regional Distribution Centers responsible for warehousing, fleet management, tracking and transfer
  - Delivery Vehicle Fleet composed of the vehicles that make deliveries for a particular distribution center
  - Storefronts and Drop Boxes
  - Customers - business and residential.

OPERATIONAL CONTEXT GRAPHIC



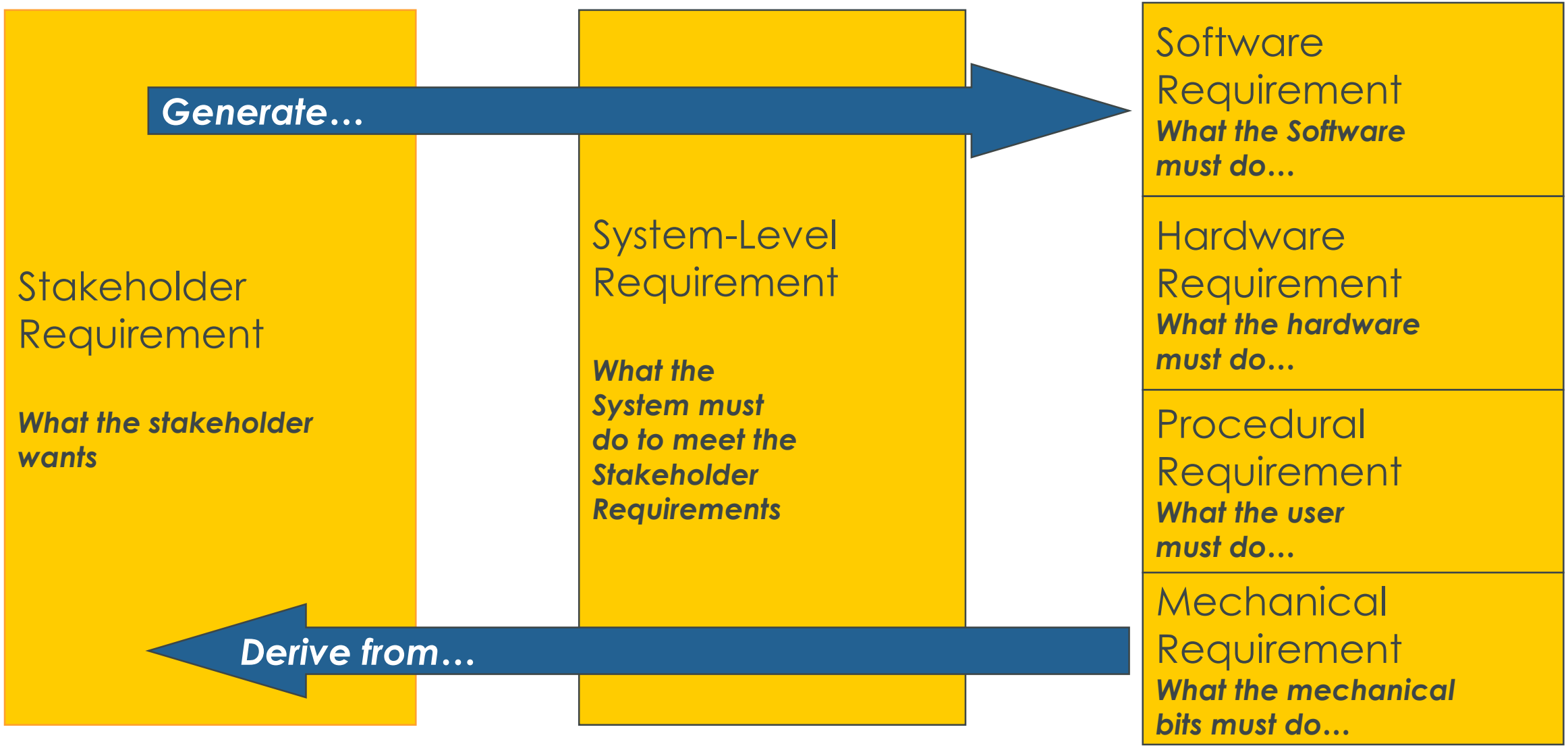
Provides a means to communicate with non-technical stakeholders while maintaining model consistency

SYSTEM OF SYSTEMS CONTEXT



REQUIREMENTS MODELING

LEVELS OF REQUIREMENTS



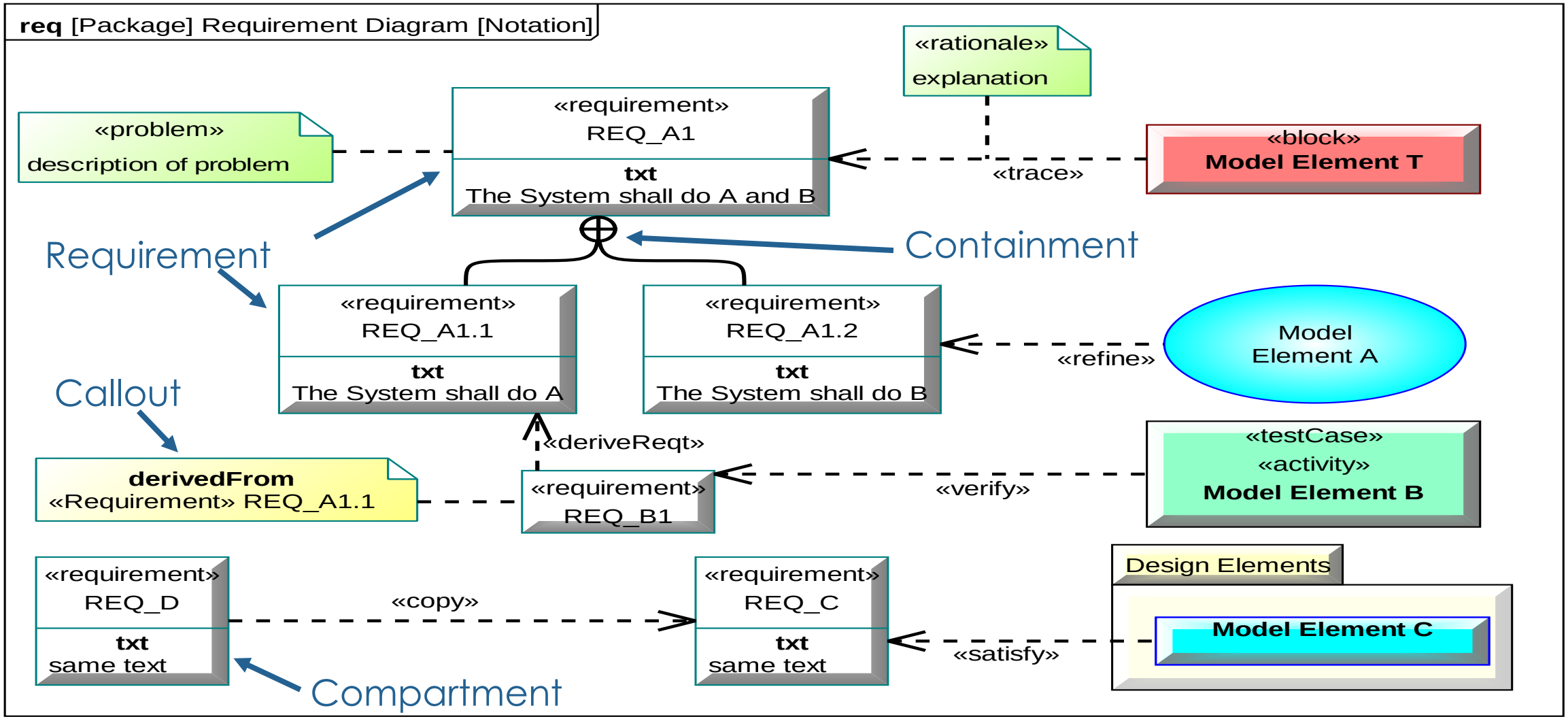
REQUIREMENTS MODELING

- Allows capture of textual requirements in a SysML model
  - From different types of external sources
  - Can show levels and structure of requirements

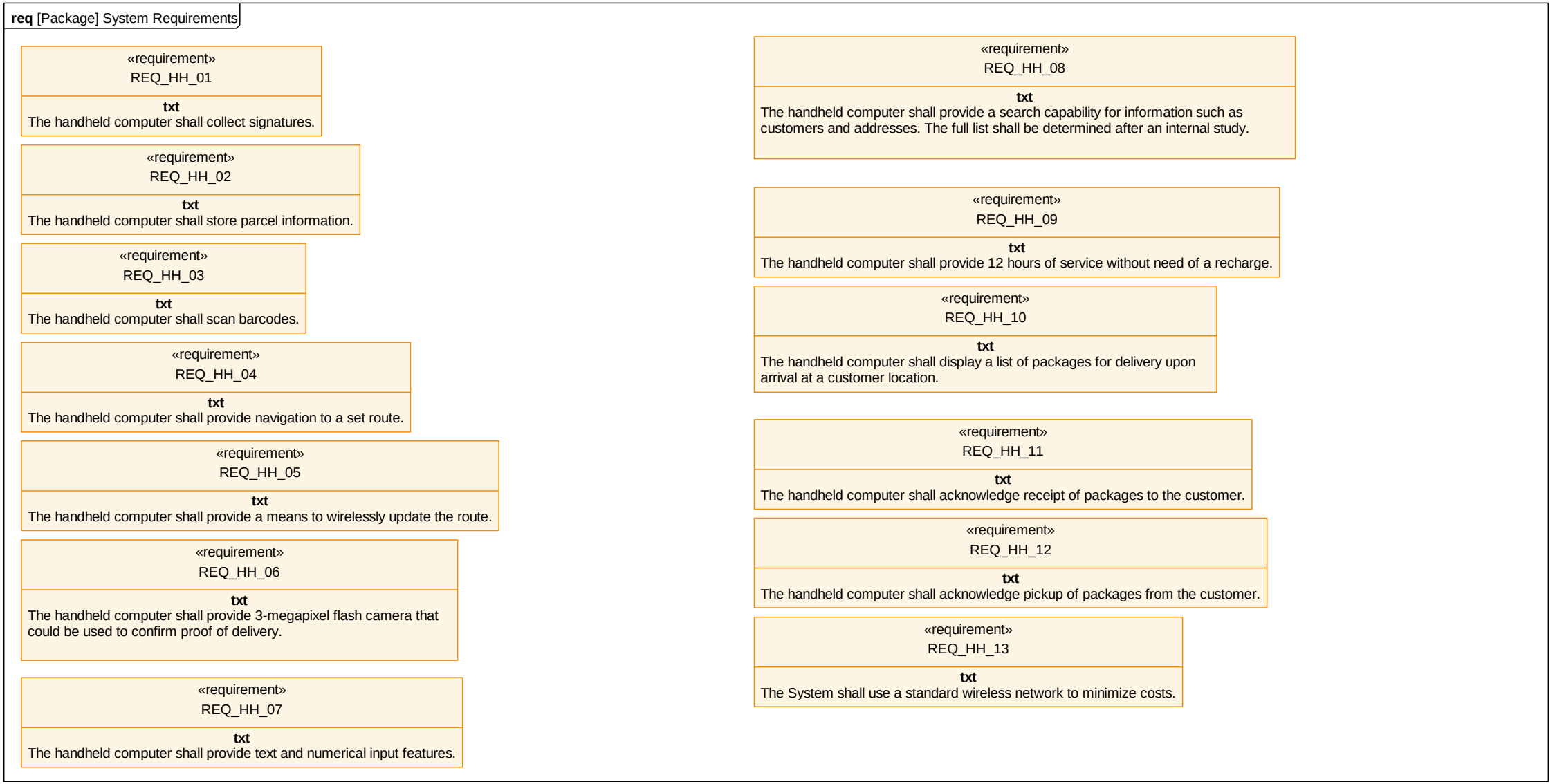
- Allows tracing of requirements to other model items
  - E.g. items that satisfy or test requirements
  - Traceability visible either way (requirement to item, item to requirement)

- Diagrammatic and tabular representation

SYSML REQUIREMENTS DIAGRAM ELEMENTS



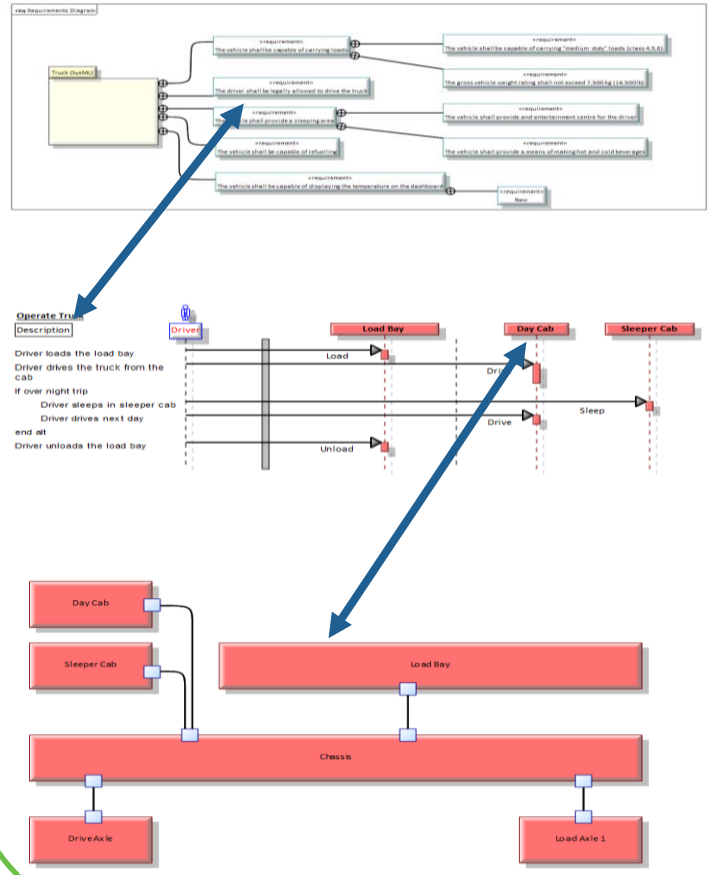
SYSML REQUIREMENTS DIAGRAM



PTC INTEGRITY MODELER – LIFECYCLE MANAGER INTEGRATION

Modeler Release 8.2

PTC Integrity™ Modeler™



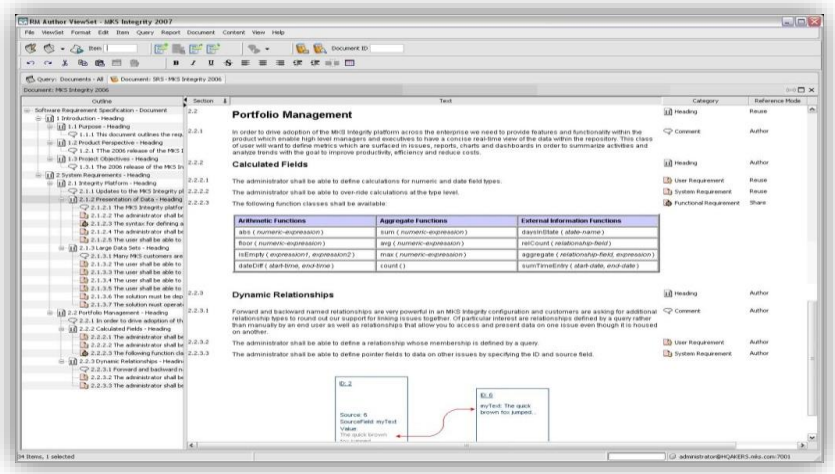
PTC Integrity Lifecycle Manager Synchronizer

Requirements

Additional Model Elements

Model Trace Links

PTC Integrity Lifecycle Manager



CONTEXT MODELING

WHY DEFINE THE SYSTEM CONTEXT?

- To clearly define what's in, and what's not in, the system under construction.
  - To clearly define information-flow across the system boundary.
  - To define the internal architecture of the system under construction.

- To identify and define all inputs and outputs to be handled by the control system

Note: In this section we use the terms "Scope" and "Context" interchangeably.
Scope is "the way that we describe the boundaries of the project."
Context is "the complete environment in which a system exists, including its location, its surroundings, and its relationship to other artefacts."
In other words, we will use the Context Diagram to help define the System Scope.

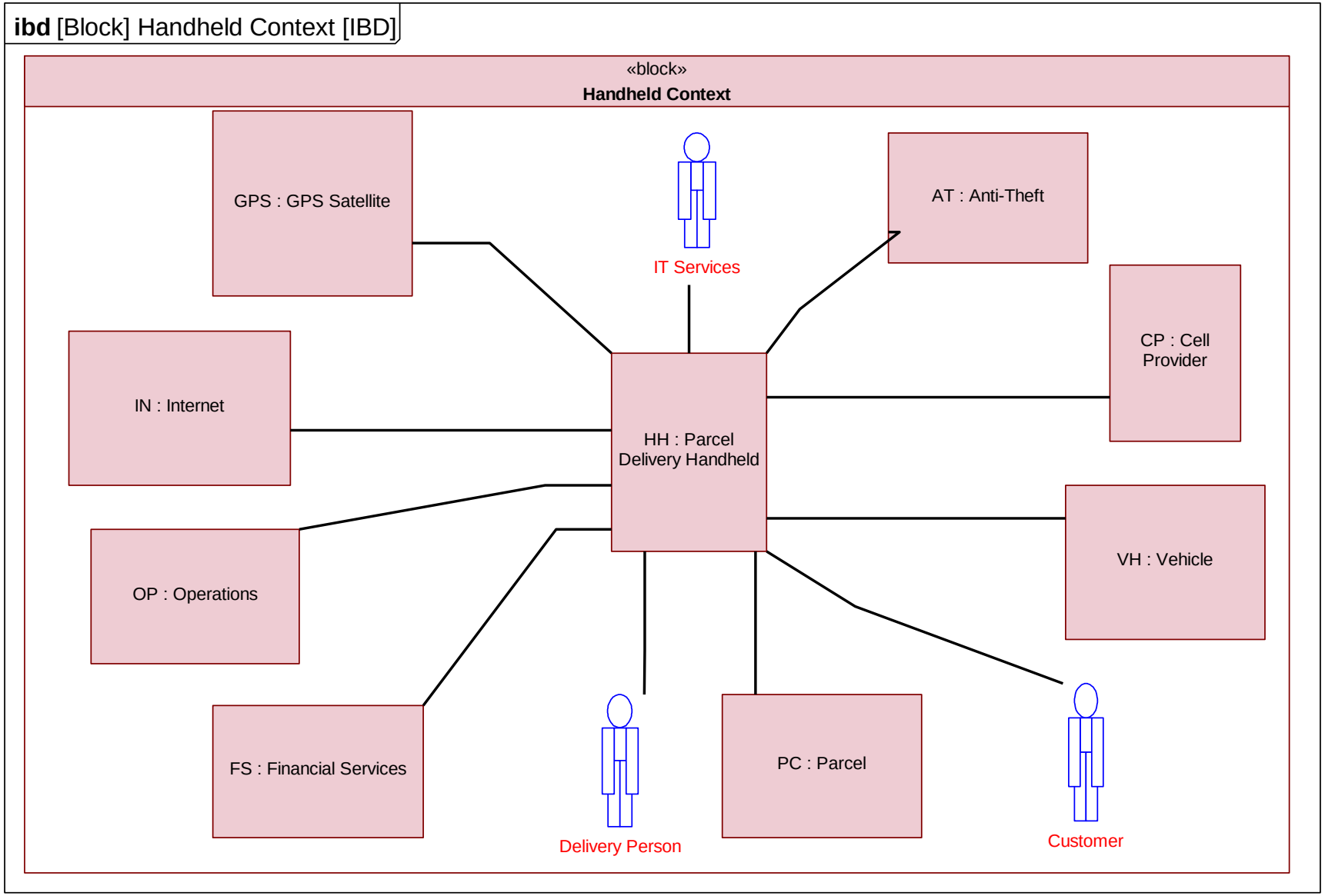
CONTEXT DIAGRAM MODELING ELEMENTS

What sort of things do we typically see on a Context Diagram?

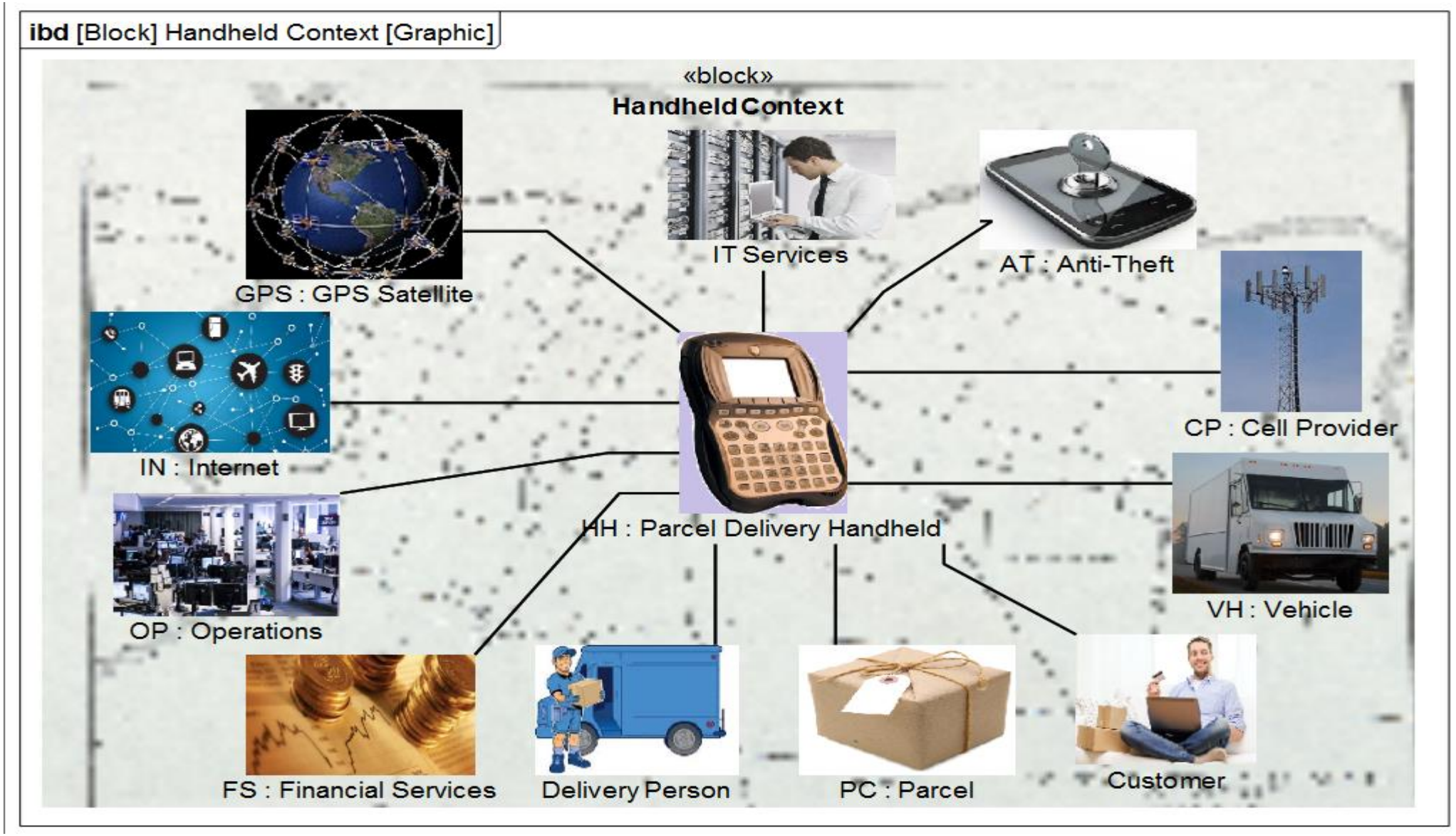
- Actor(s)
  - Representing entities outside the scope of the system under construction (e.g. people, external systems, etc.).
- The System Under Construction
  - The boundary between actors and internal elements
- Subsystems
  - Those internal (container) parts of the system which are further decomposed.
- Interface Devices
  - Internal parts which are not further decomposed – typically have a single specific port/memory address.
  - May be 'typed' by a class defining device properties
- Connectors
  - A physical or logical connection between parts or part(s) and actor
- Flows
  - Abstract representation of information, physical, force, etc. flow along a connector

Work in groups to define a context diagram for the handheld device

OPERATIONAL CONCEPT WITH BOXES



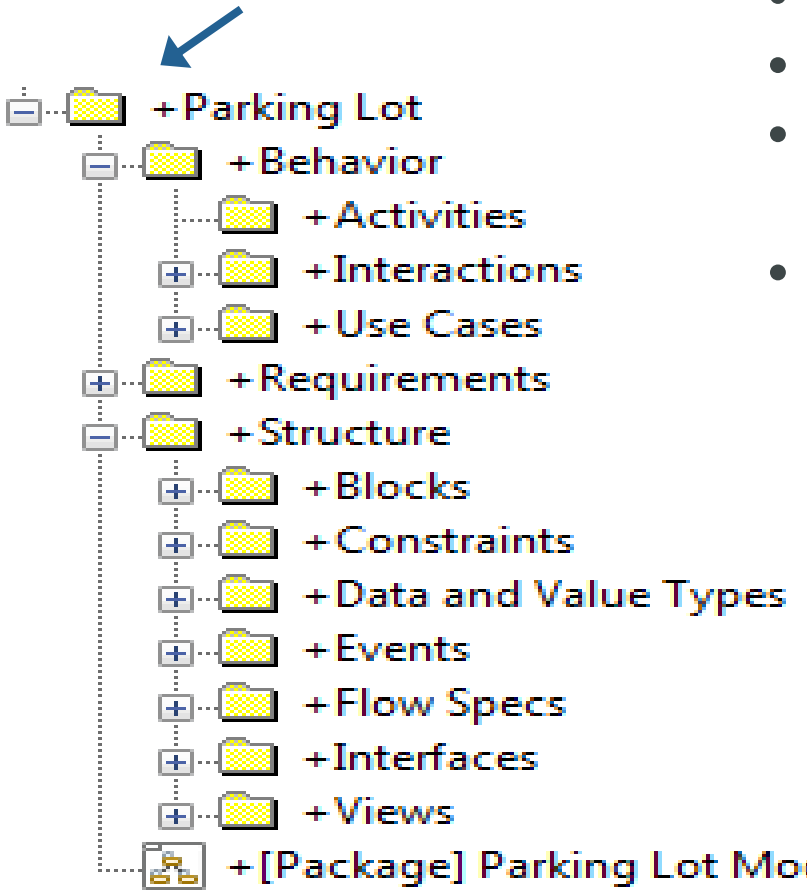
OPERATIONAL CONCEPT WITH GRAPHICS



MODEL STRUCTURE

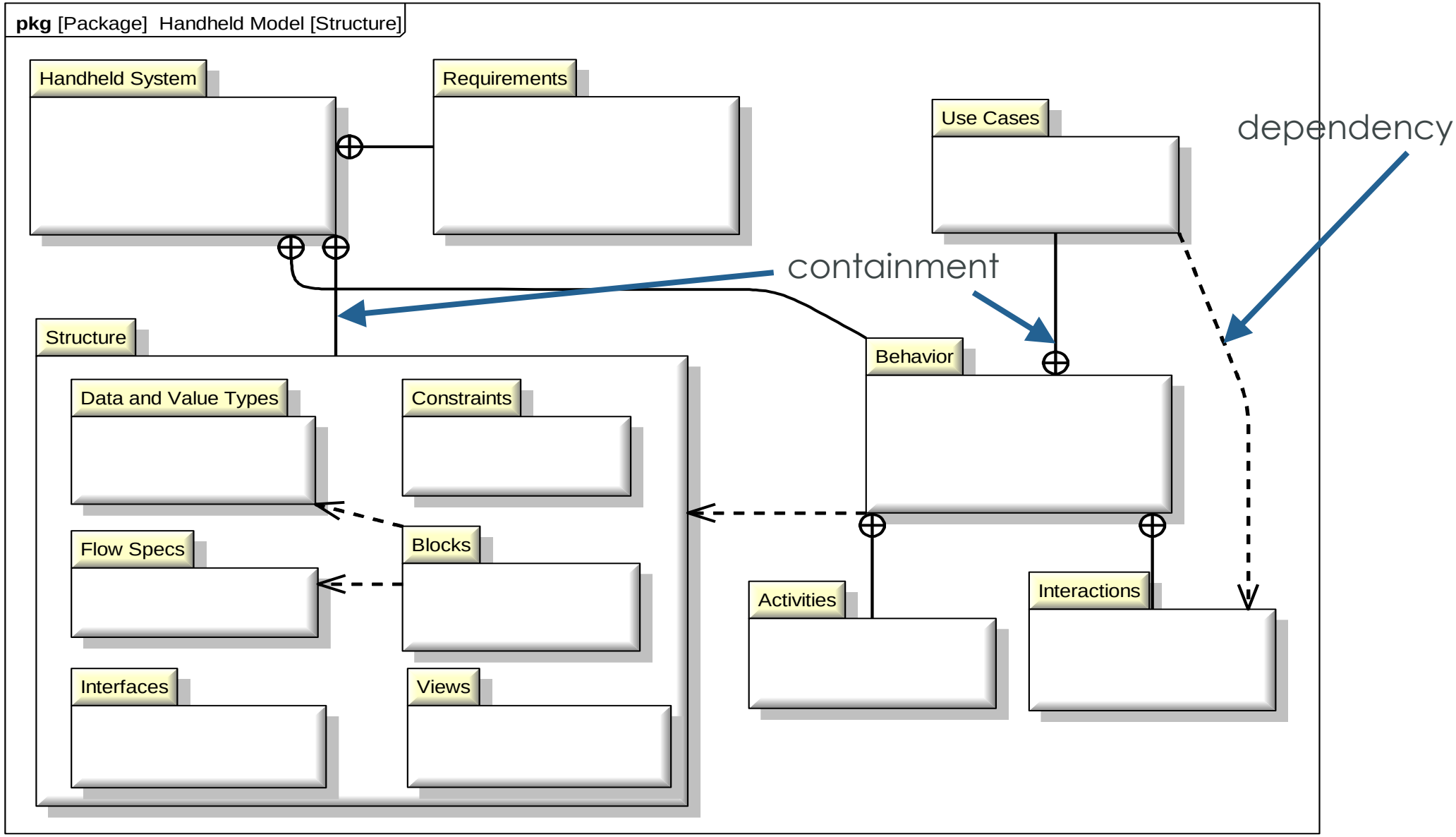
PACKAGES

Packages Browser view



- Can group any number and type of model elements in any way appropriate to the needs of project
- Can contain other (nested) packages
- Similar concept to folders in a hierarchical file system
- Implicit and Explicit Dependencies may exist between packages
- Package structure and explicit dependencies can be shown on Package Diagram

PACKAGE DIAGRAM - EXAMPLE



USE CASE MODELING

USE CASES – OVERVIEW

- A Use Case encapsulates the functionality of a system ‘capability’ or a system ‘service’.
- The use case defines a purpose or goal with respect to an external user (actor) of the system .
- Use Cases can form the basis of a system-level test, simplistically:
  - System Acceptance Test = Test all the Use Cases
- For the systems engineer, Use Case modeling provides a convenient vehicle to discuss capabilities with the system stakeholders.
- Use Case concepts and modeling are inherited from UML.

ACTORS

External entities with respect to the system

Expected Actors

A Person

An External System

Abstract such as Time

Unexpected Actors

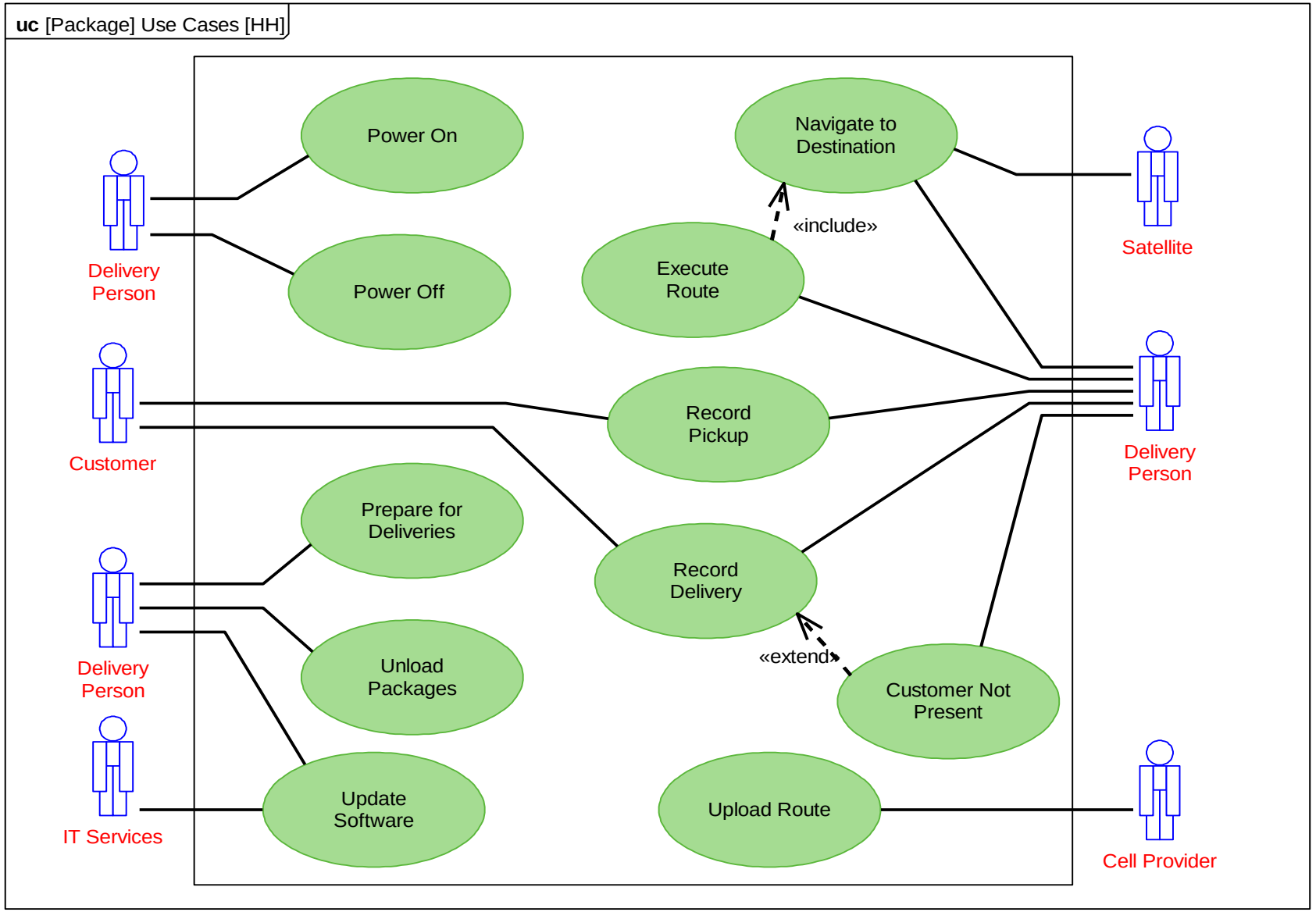
Unauthorized Users

Power (loss)

Adverse Environmental Conditions

Work in groups to define the actors and use cases for the handheld device

HANDHELD USE CASES AND STAKEHOLDERS



ACTIVITY MODELING

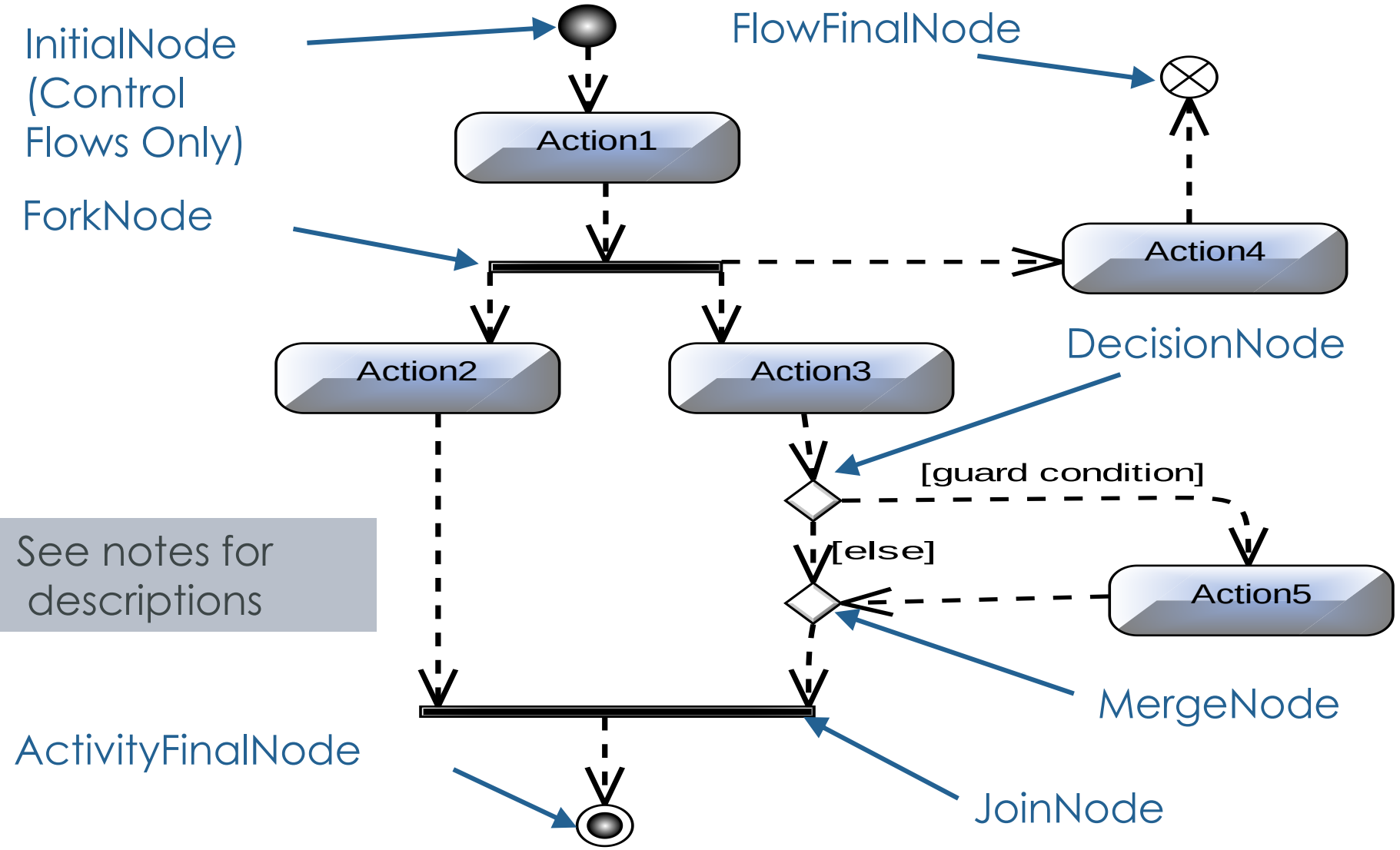
ACTIVITY MODELING - OVERVIEW

- "Activity modeling emphasizes the inputs, outputs, sequences, and conditions for coordinating other behaviors." (SysML)
  - 'behavior' as in something the system does
  - describes flow of activity and objects (matter, information, etc.)
    - Uses the concept of 'tokens' in a Petri net like way (see Notes)
  - rich notation set allows detailed modeling
  - can be linked with structural modeling
- An activity is a behavior element that specifies control and object flow between subordinate actions
- An activity diagram is owned by an activity and describes the detail of that activity
- Useful for many purposes, including:
  - Describing system functionality
  - Defining block operation behavior
  - Data/item flow modeling
  - etc.

ACTIVITIES

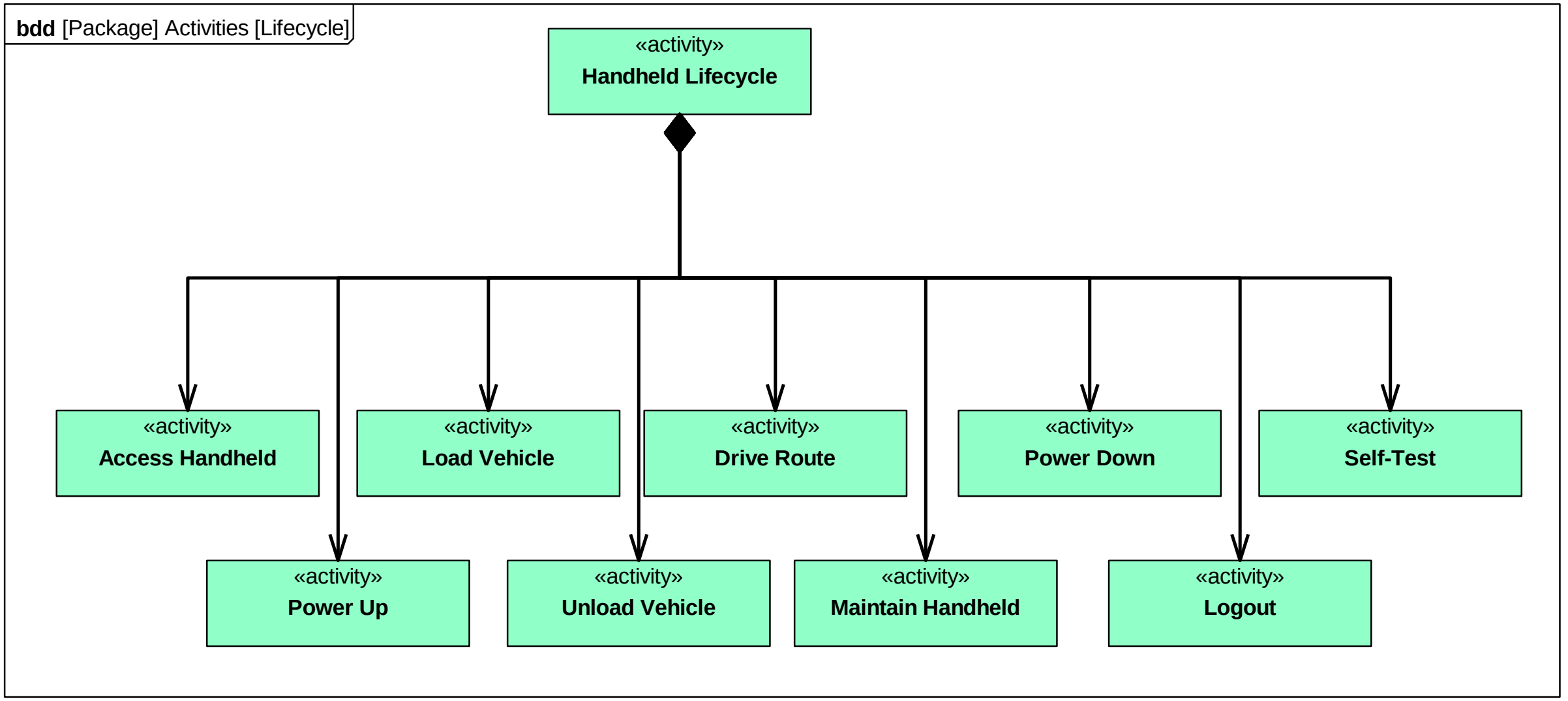
- An activity :
  - is a type of behavior
  - defines the activation (and so coordination or sequencing) of subordinate behaviors (actions) using control and object (data) flow
  - can have parameters (in, out and bidirectional)
  - is reusable in different contexts
  - can own an activity diagram
  - can have Precondition and Postcondition constraints

CONTROL NODES

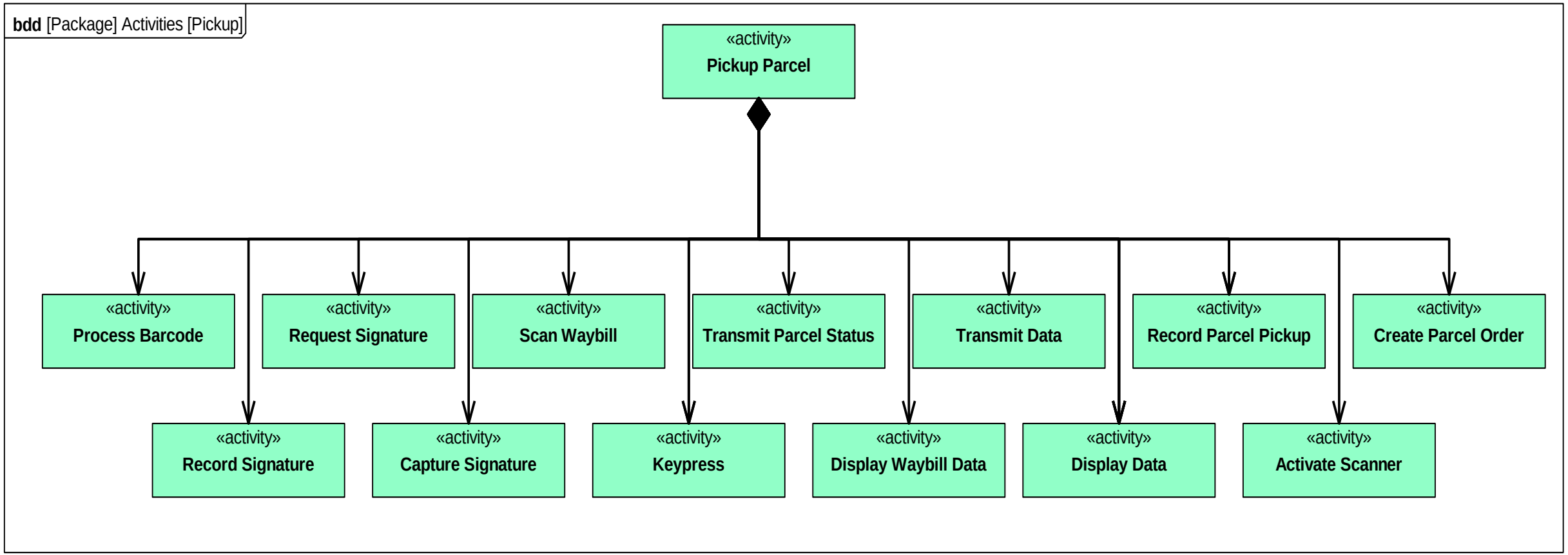


See notes for descriptions

HANDHELD LIFECYCLE ACTIVITIES

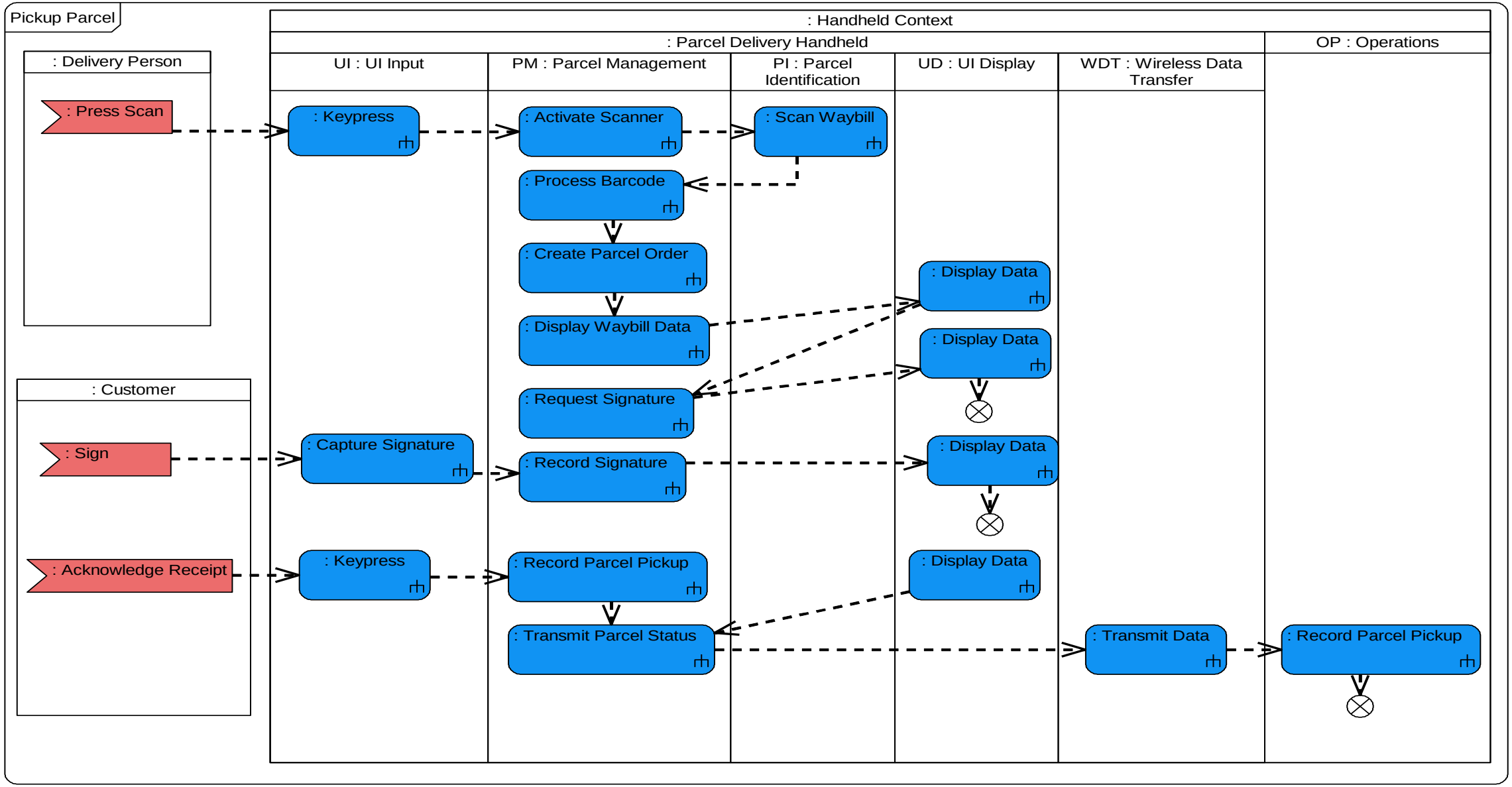


PICKUP PARCEL ACTIVITY STRUCTURE



Work in groups to define an activity diagram for the handheld device for parcel pickup

PICKUP PARCEL ACTIVITY DIAGRAM



STRUCTURE MODELING

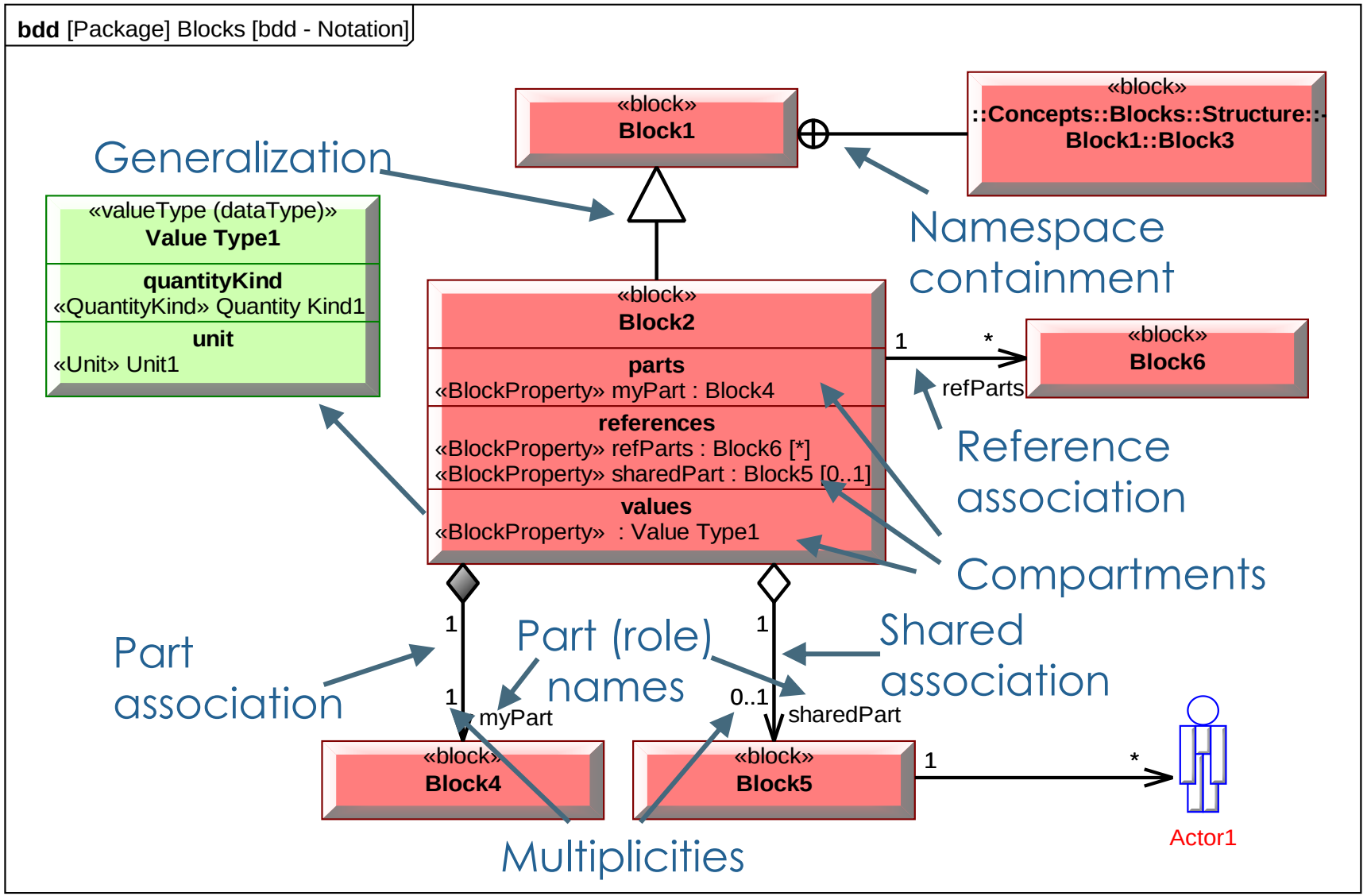
BLOCK DIAGRAMS - INTRODUCTION

- Capture information relating to structural aspects of systems
  - Physical component structures
  - Data/message structures
  - Organizational structures
  - Etc.

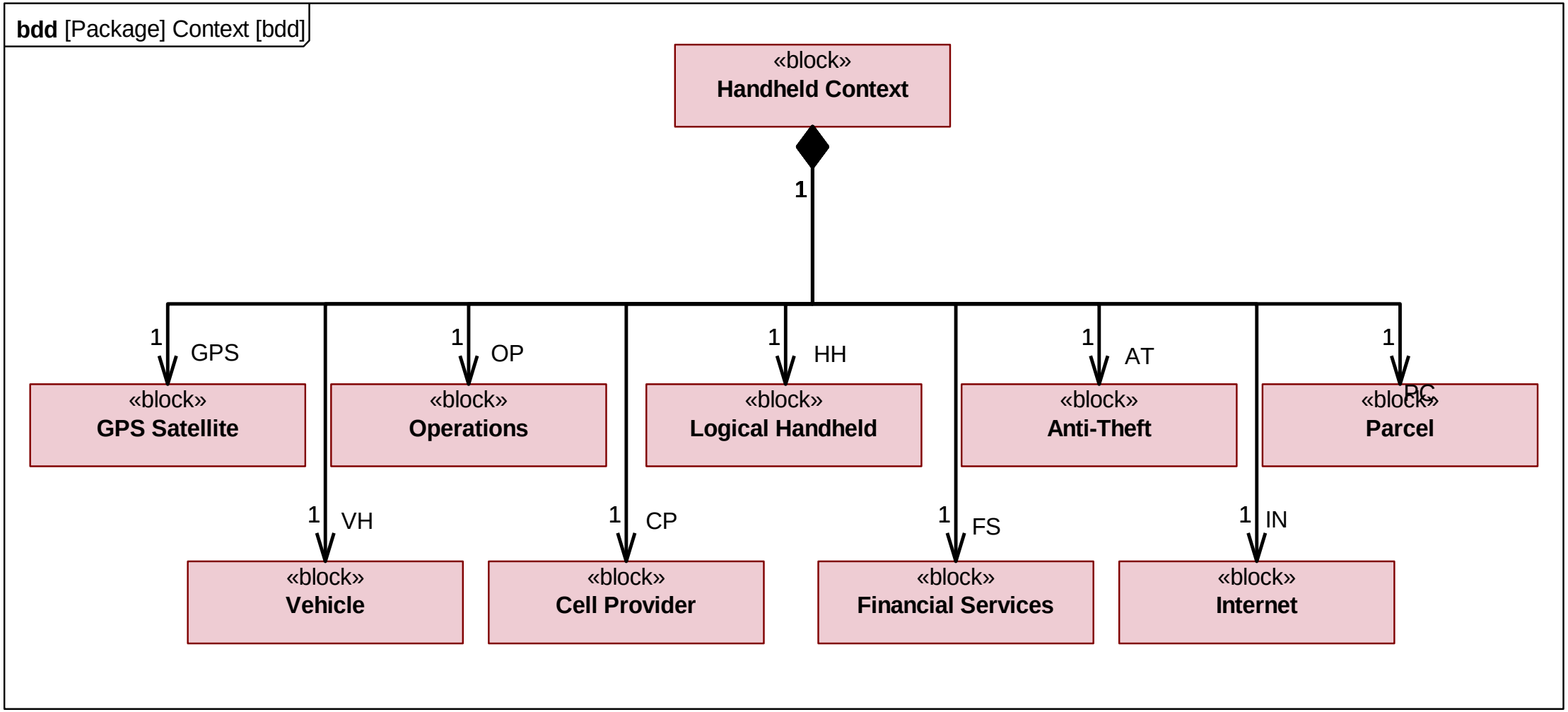
- Provide two different but complementary and consistent views
  - Block Definition Diagram (BDD)
    - shows block features and relationships
    - one diagram may include many blocks
    - can also be used to show certain other modeling items
      - e.g. Signals, ValueTypes, ConstraintBlocks, etc.
  - Internal Block Diagram (IBD)
    - shows internal composition of a single block
      - its parts and what flows between them

- The diagrams together support hierarchical decomposition of the system

BLOCK DEFINITION DIAGRAM (BDD) NOTATION

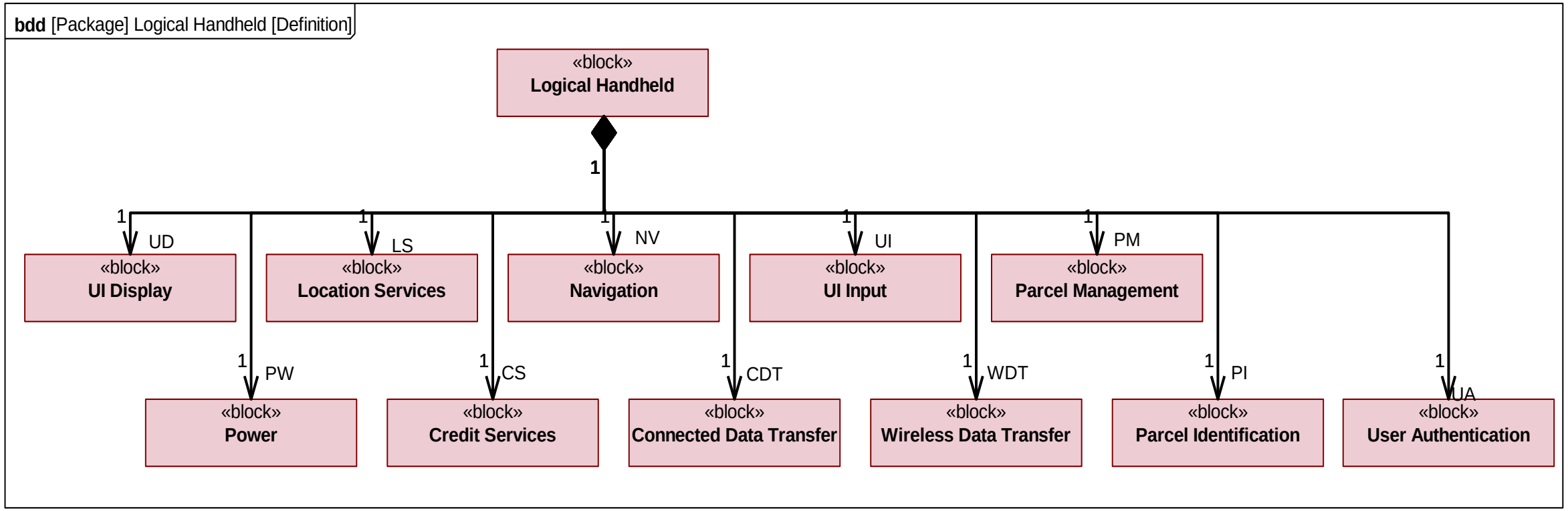


CONTEXT BLOCK DEFINITION DIAGRAM (BDD)

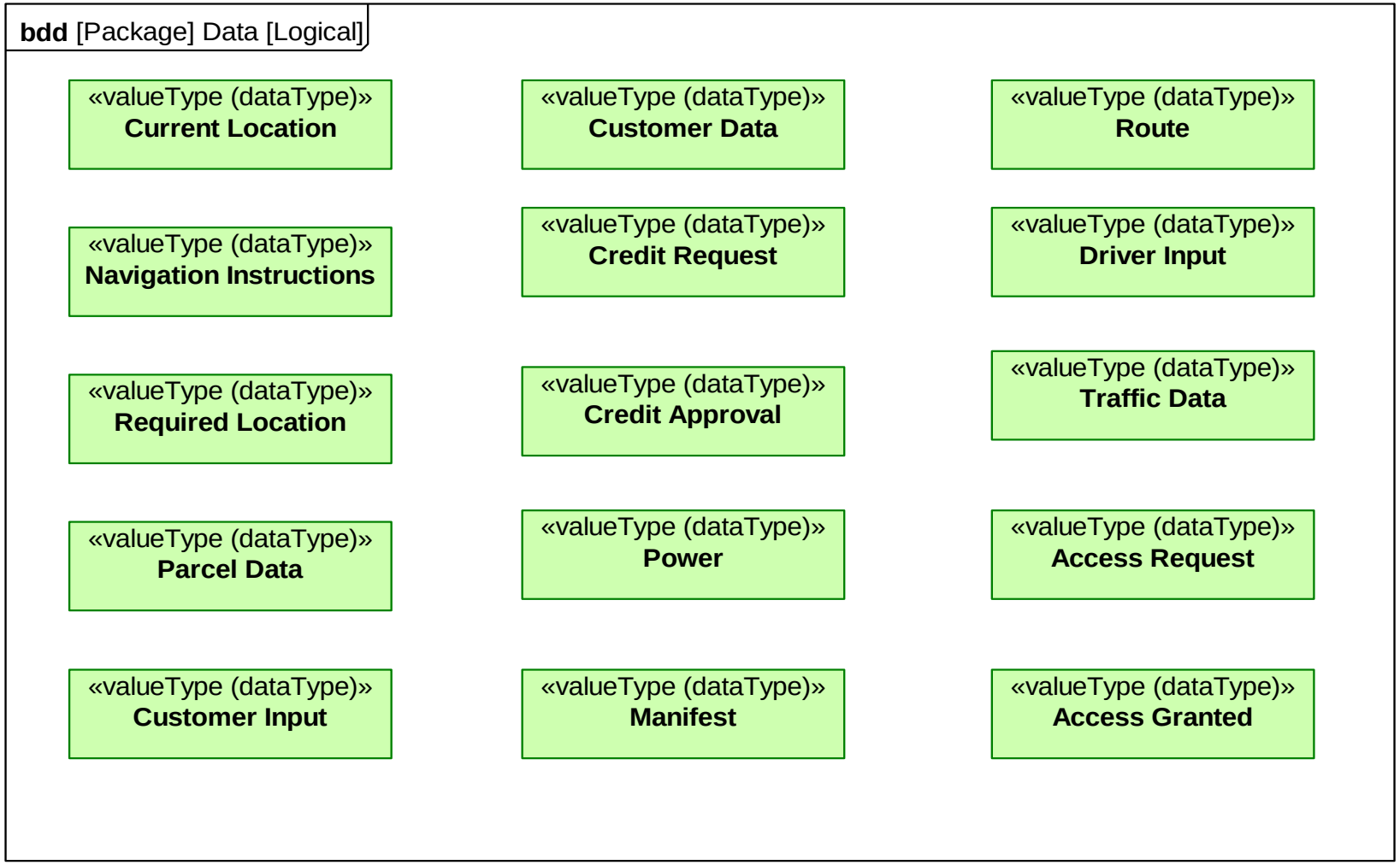


Work in groups to define the main elements for the handheld device

HANDHELD SYSTEMS



DATA TYPES FOR EXCHANGE



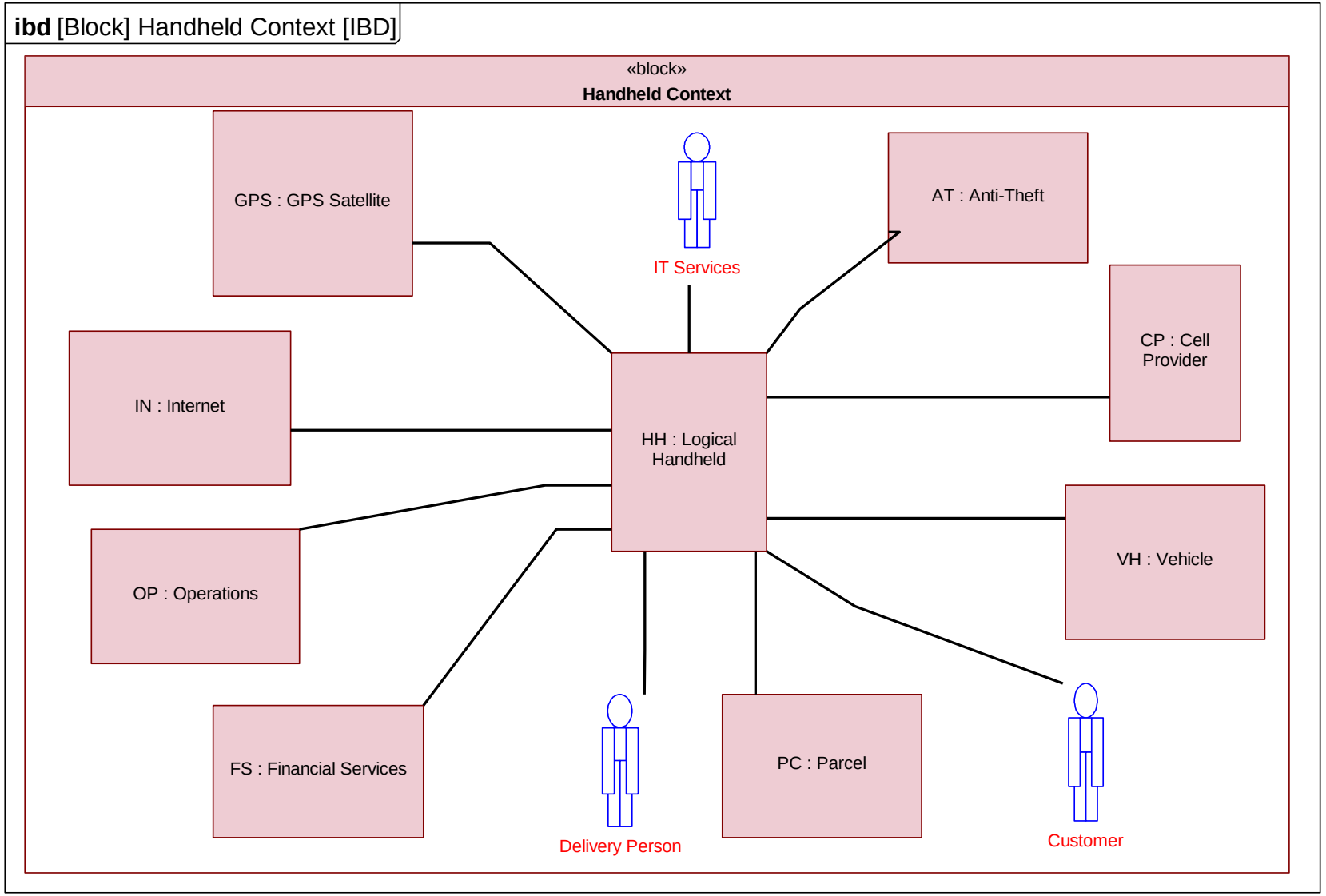
INTERNAL BLOCK DIAGRAM – USAGE

- Is owned by a block

- Shows internal structure of that block
  - Its parts (consistent with BDD)
    - to any level of nesting
  - Connections and flows between the parts
  - Associations between parts and actors

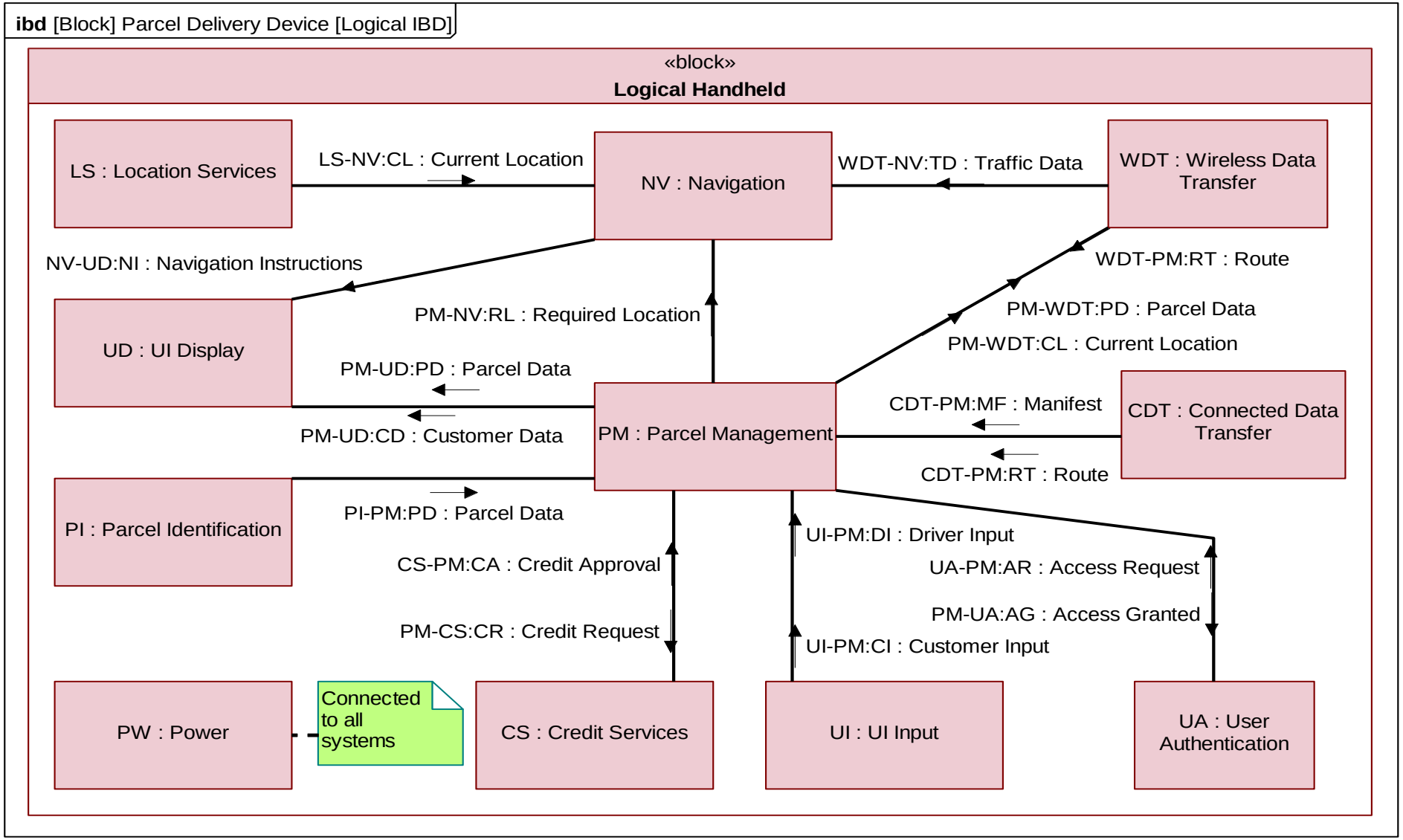
- Can specify connection interfaces (ports)
  - internal – on parts
  - external – on owning block

CONTROL CONTEXT



Work in groups to define the internal connections for the handheld device

HANDHELD LOGICAL SYSTEM

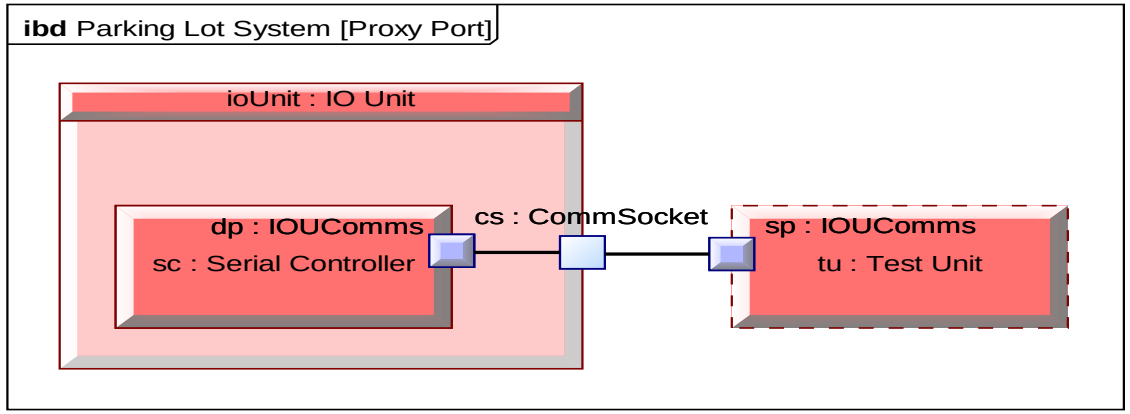


PROPOSED SOLUTIONS

PROXY PORTS

- Just like an 'electrical wire'
  - Do not represent system components
  - Do not contain behavior
  - Simply provide *'indirect'* access to data/features of the block and/or internally nested part or parts within the block.
  - Sometimes called a *'Delegation Port'*
  - Typed by an Interface Block (specifies the exposed data / features)
  - Can be conjugated (Conjugation covered with flow ports)

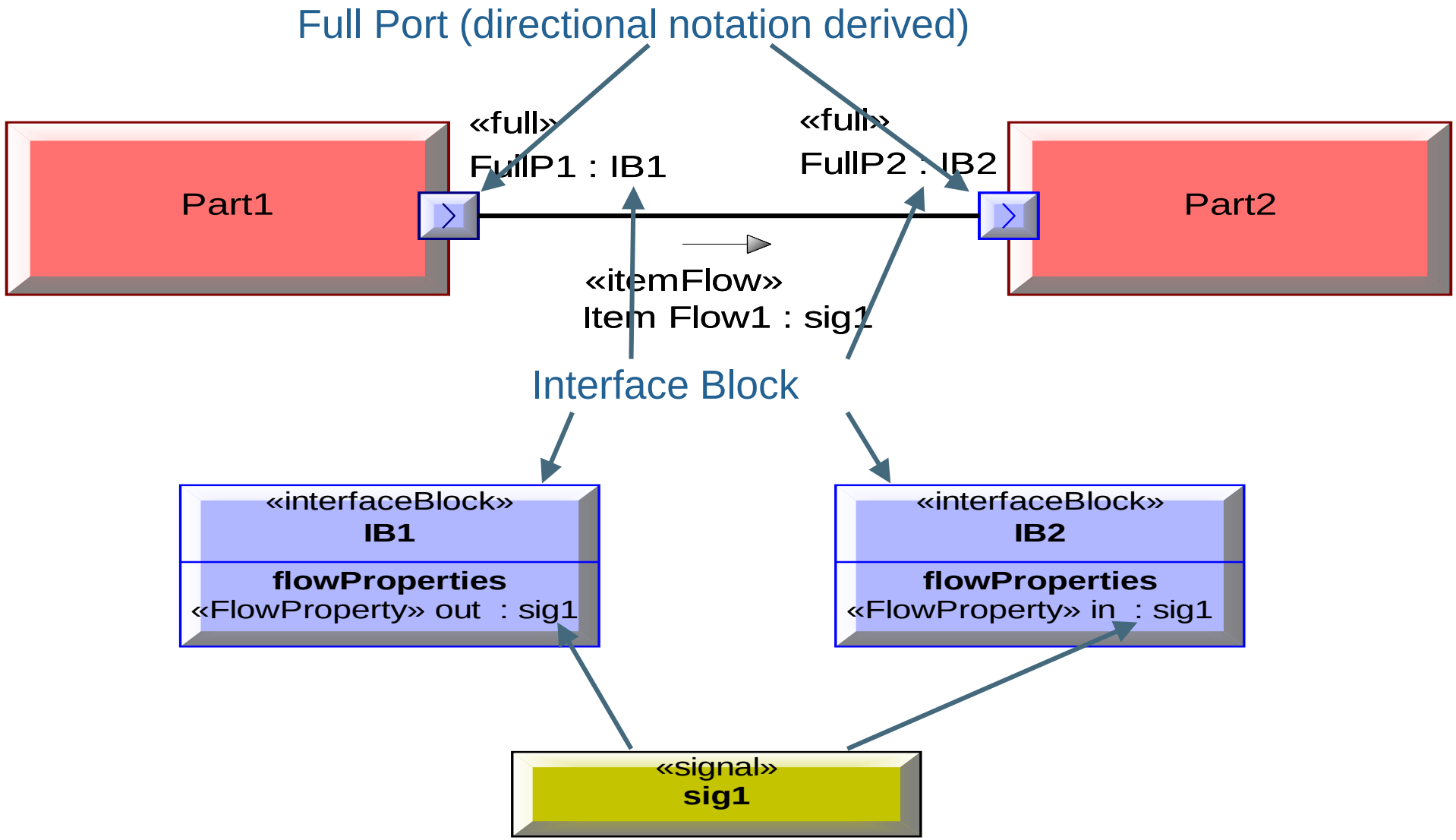
- Preserves the principle of 'Block Encapsulation'
  - Operate at the system boundary delegating data/access to and from the block and internal *'exposed'* Parts



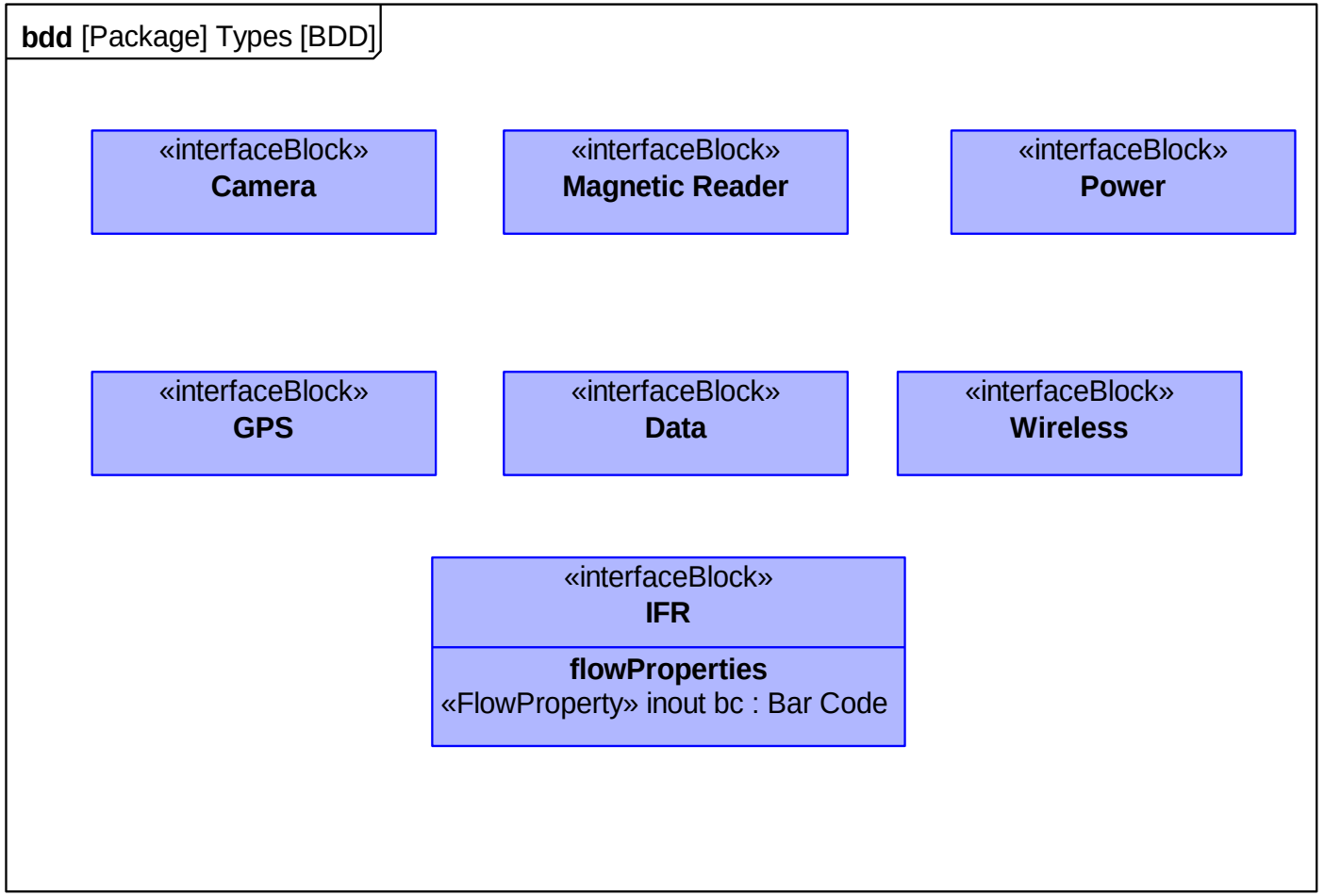
FULL PORTS

- Just like a SysML ‘Part‘
  - Can be used to model physical connections/mountings.
  - Represent actual system components (Generally typed by a block).
  - Can have behaviour and nested parts just like a normal block with block properties.
  - Can handle incoming items, operations, and signals themselves, and can send out items.
  - Can connect to other system components (Blocks) via Proxy Ports
- Operate at the System Boundary
  - Environment can directly access the exposed feature(s) via the system boundary.

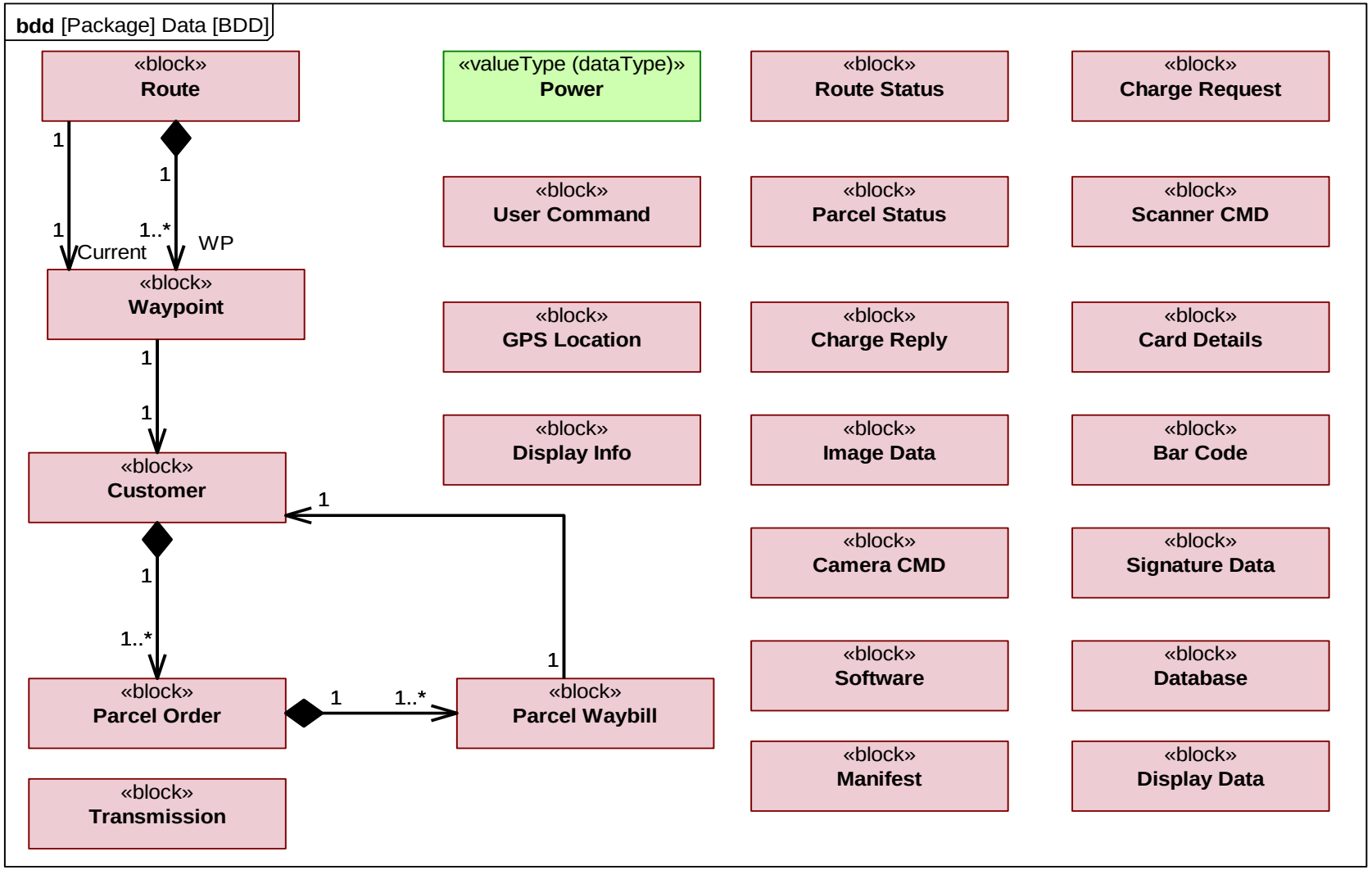
FULL PORT NOTATION



SYSTEM INTERFACE BLOCKS



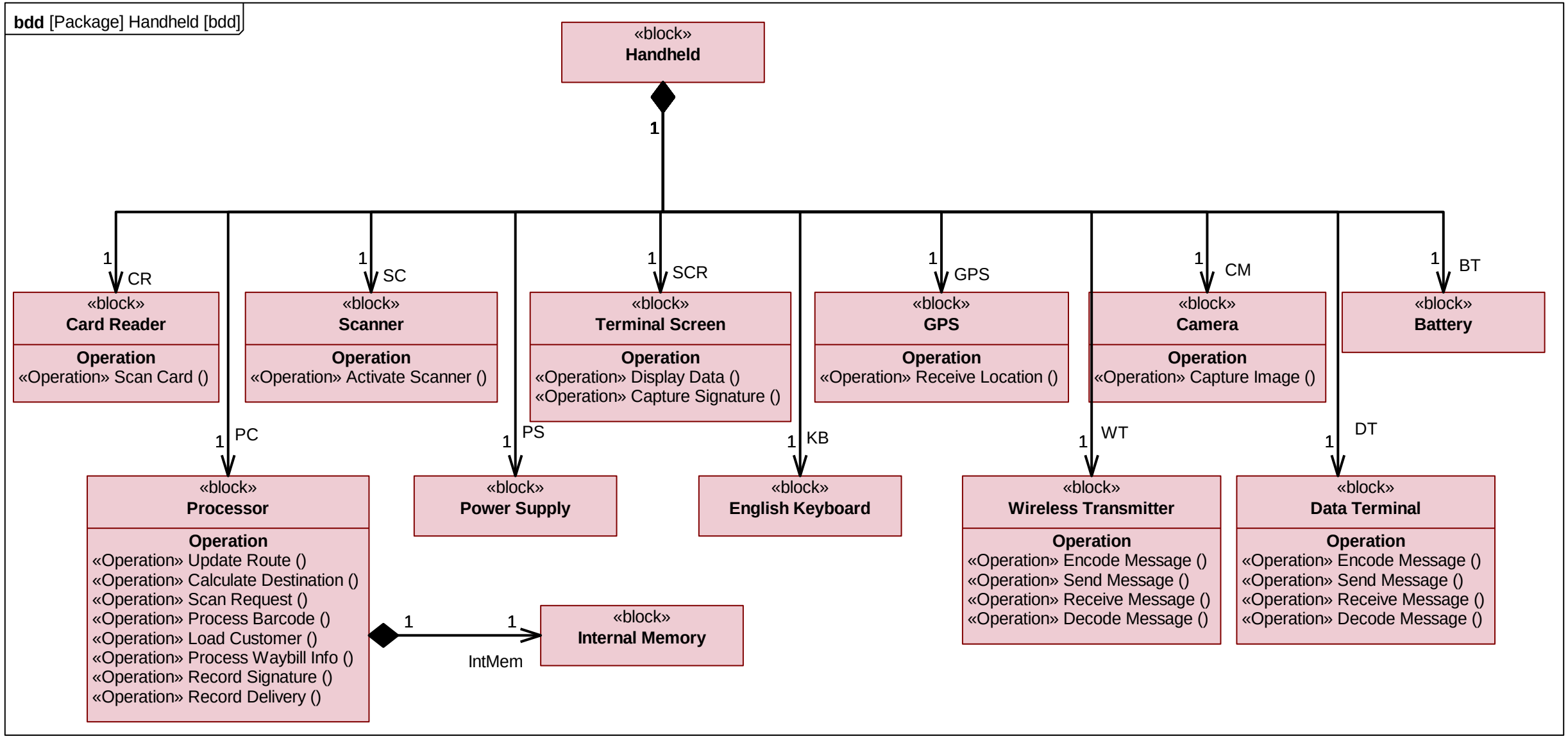
SYSTEM DATA MODEL



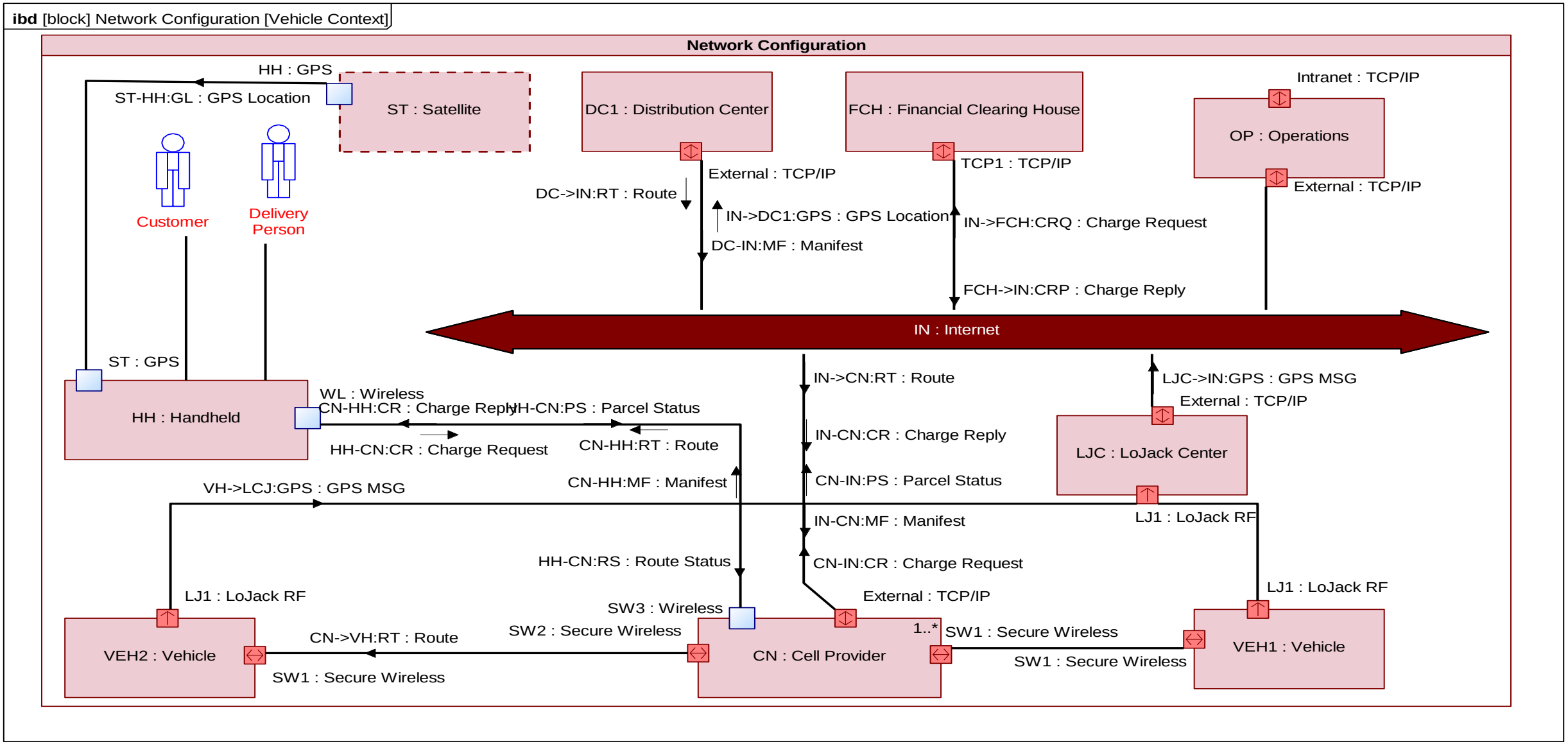
LOGICAL V. PHYSICAL MODELING WITH IBDS

- IBDs can be used to capture both a logical model of parts, connections and flows, and a physical model
- Logical model focuses on logical parts and flows and may not show ports or types (unless logical types defined)
  - Based on specification rather than implementation ('what' not 'how')
  - Abstract types (if any)
- Physical model focuses on physical parts and flows and normally shows ports and physical (implementation) types
  - Normally follows logical modeling
  - May be many physical models for one logical model
  - Real-world types
- May affect package structure
  - Logical package contains logical types
  - Physical package contains physical types
- Can link logical model items to physical model items via Allocation (later in course)

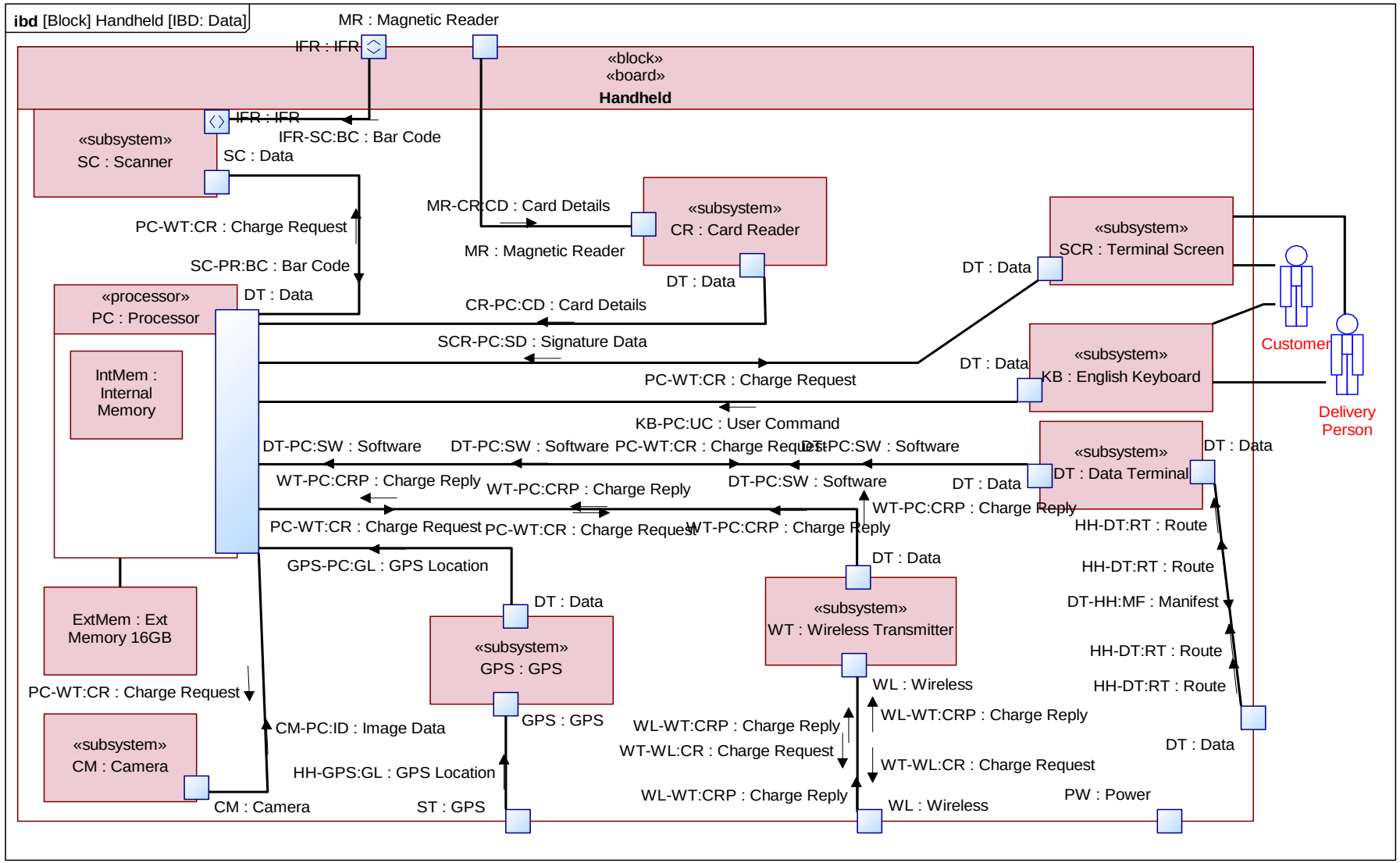
BLOCK DEFINITION DIAGRAM – SYSTEM LEVEL



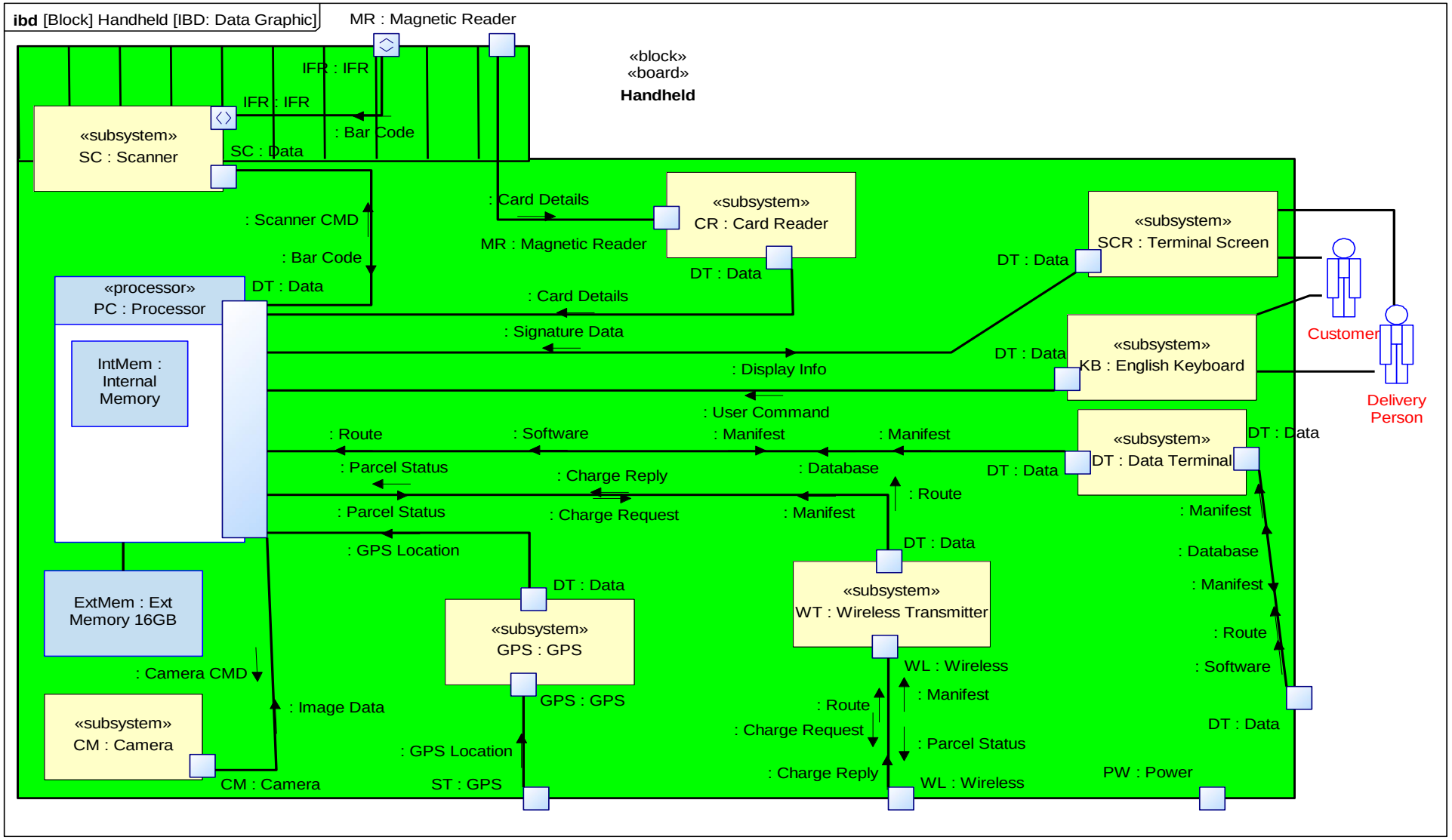
EXAMPLE IBD – PHYSICAL MODEL



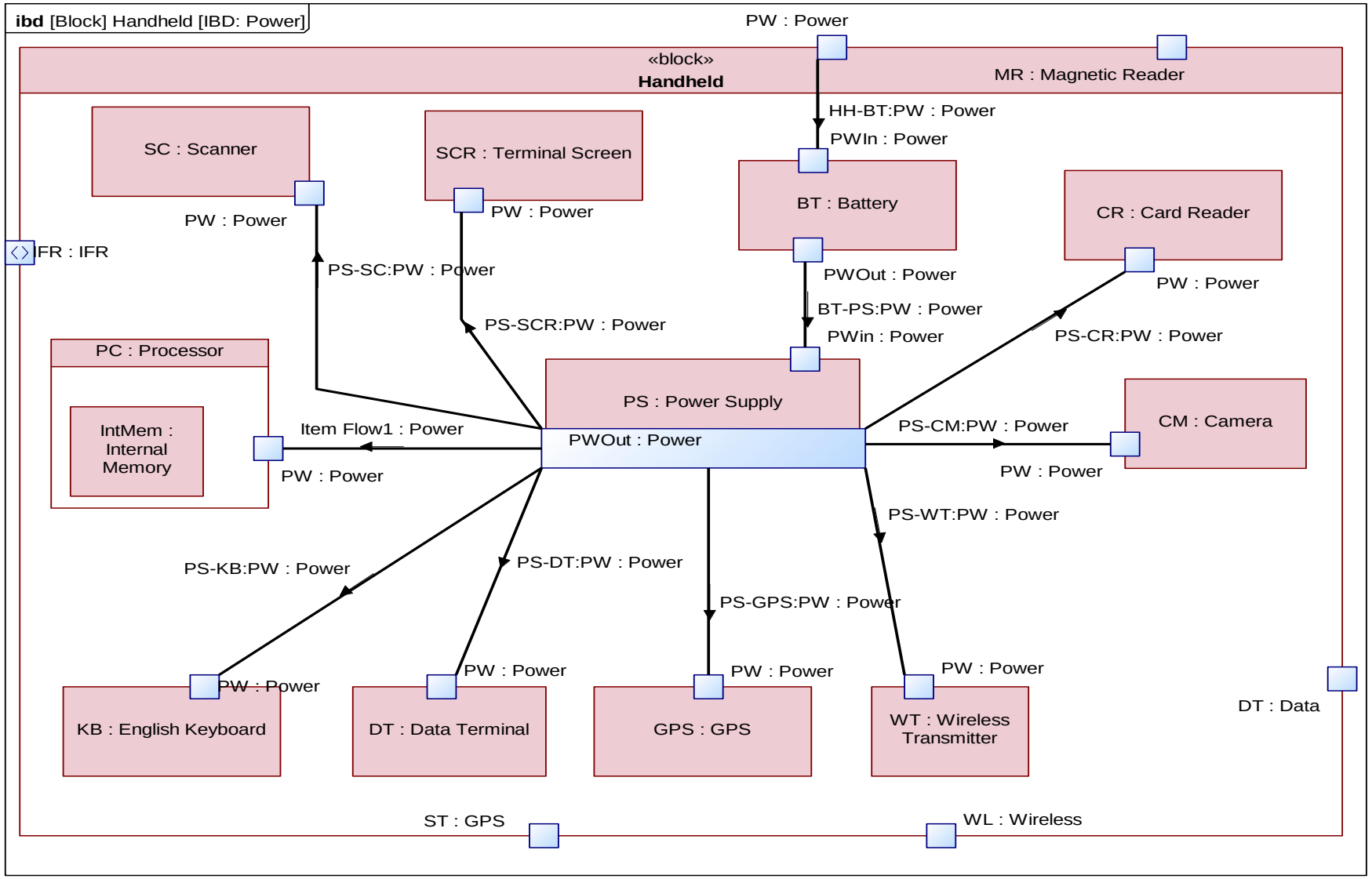
SYSTEM DATA INTERACTIONS



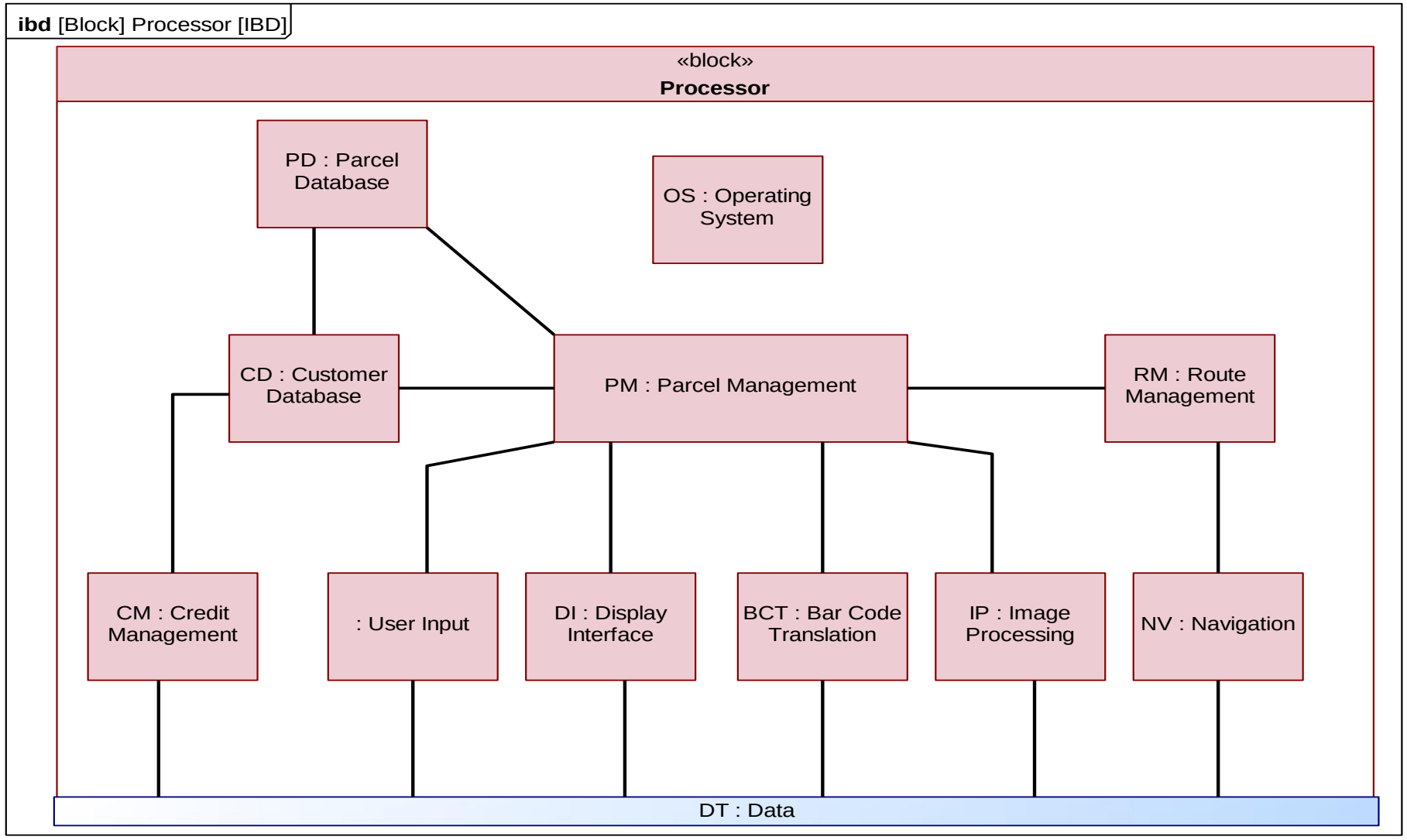
DATA INTERACTIONS W/GRAPHIC



POWER CONNECTIONS



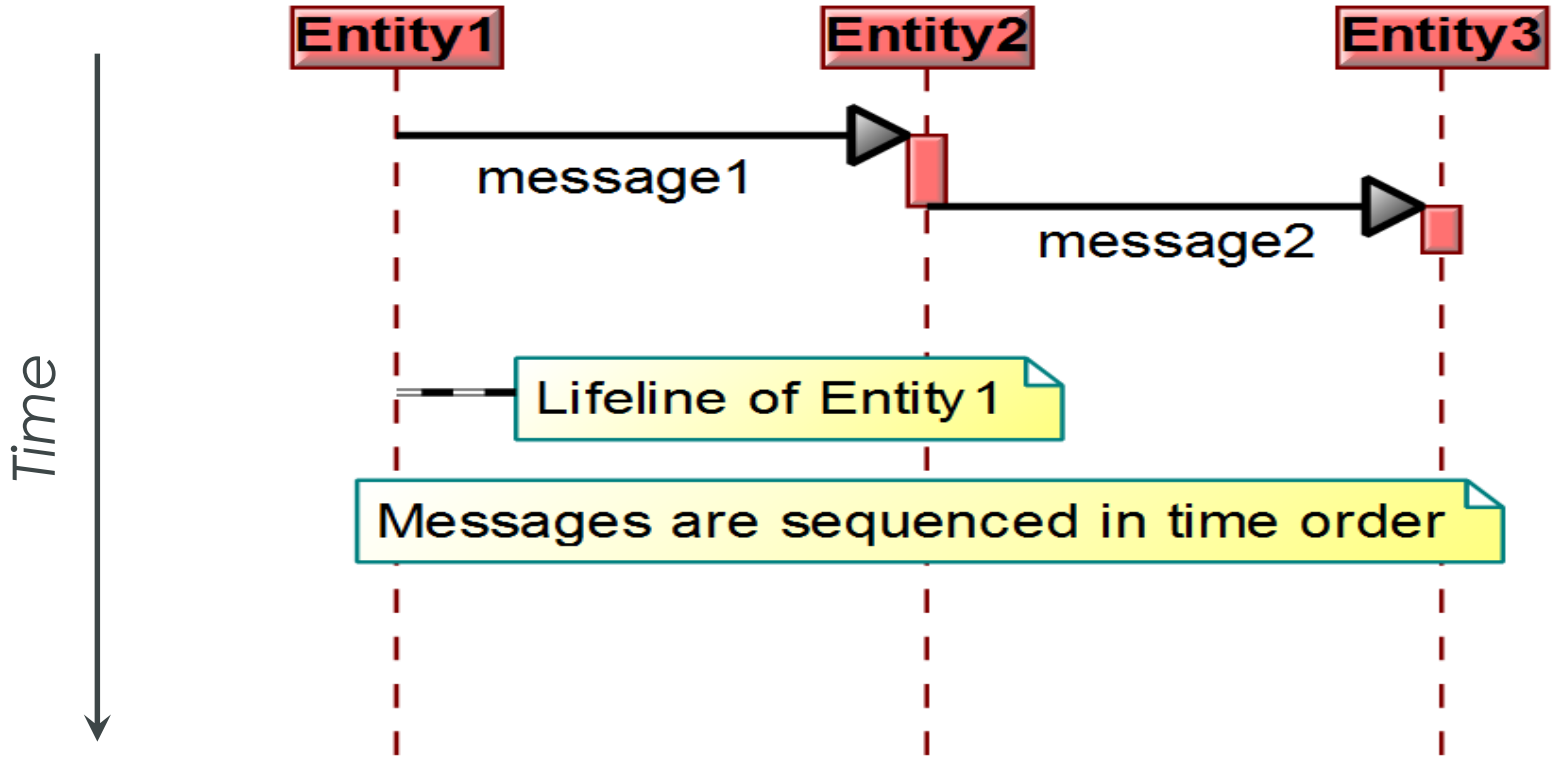
SOFTWARE SYSTEMS



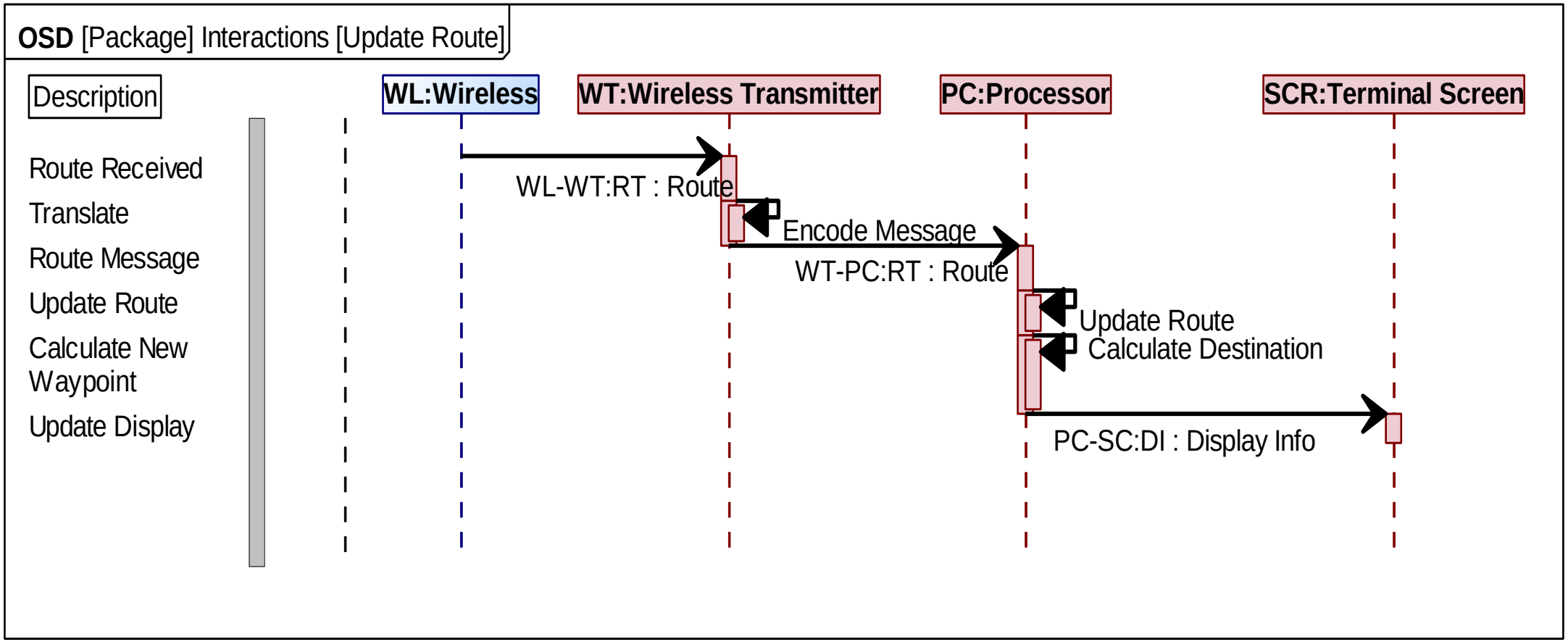
SEQUENCE MODELING

SEQUENCE DIAGRAM – OVERVIEW

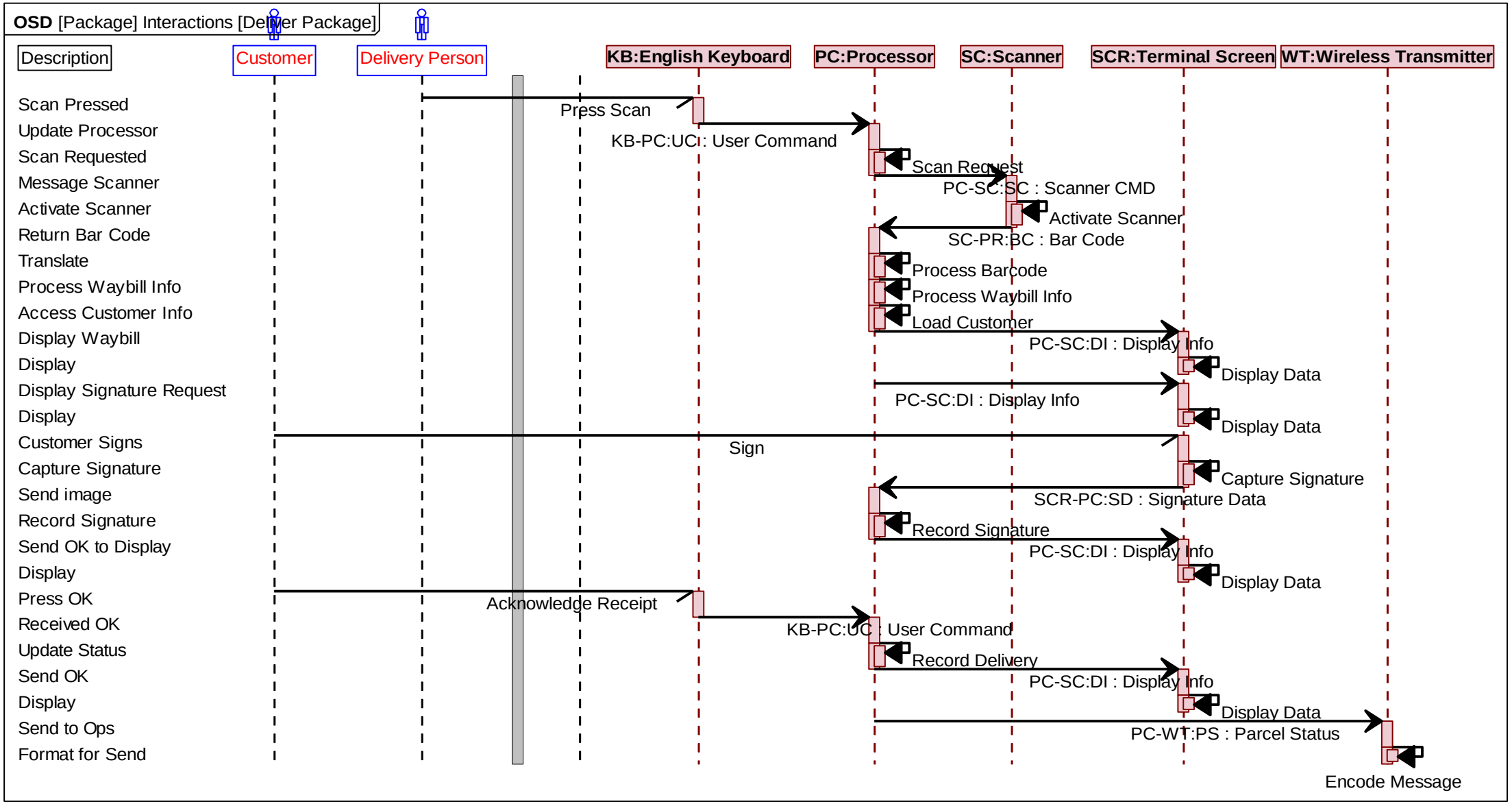
- Sequence diagrams depict sequence of message interchange (e.g. operations are shown below) between the lifelines of entities.
- Entities collaborate to realize a use case (or other behavior).



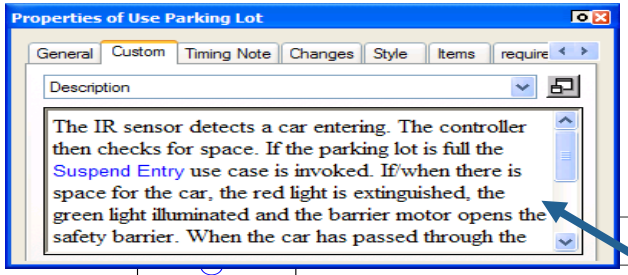
EXAMPLE – HIGH LEVEL DIAGRAM



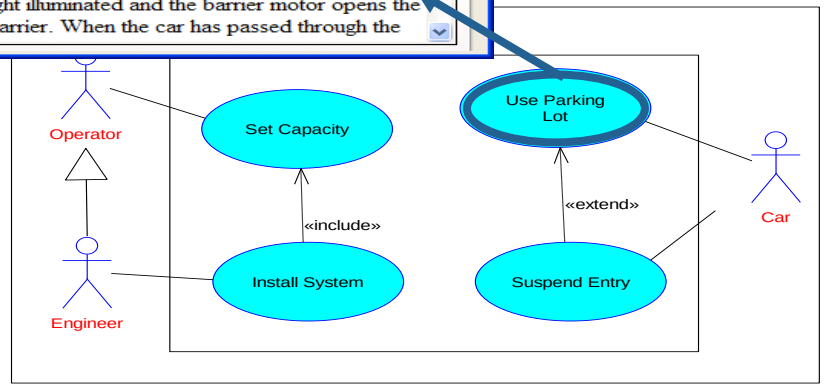
EXAMPLE


OSD [Package] Interactions [Deliver Package]
Description
Customer
Delivery Person
KB:English Keyboard
PC:Processor
SC:Scanner
SCR:Terminal Screen
WT:Wireless Transmitter
Scan Pressed
Update Processor
Scan Requested
Message Scanner
Activate Scanner
Return Bar Code
Translate
Process Waybill Info
Access Customer Info
Display Waybill
Display
Display Signature Request
Display
Customer Signs
Capture Signature
Send image
Record Signature
Send OK to Display
Display
Press OK
Received OK
Update Status
Send OK
Display
Send to Ops
Format for Send
Press Scan
KB-PC:UC : User Command
Scan Request
PC-SC:SC : Scanner CMD
Activate Scanner
SC-PR:BC : Bar Code
Process Barcode
Process Waybill Info
Load Customer
PC-SC:DI : Display Info
Display Data
PC-SC:DI : Display Info
Display Data
Sign
Capture Signature
SCR-PC:SD : Signature Data
Record Signature
PC-SC:DI : Display Info
Display Data
Acknowledge Receipt
KB-PC:UC : User Command
Record Delivery
PC-SC:DI : Display Info
Display Data
PC-WT:PS : Parcel Status
Encode Message

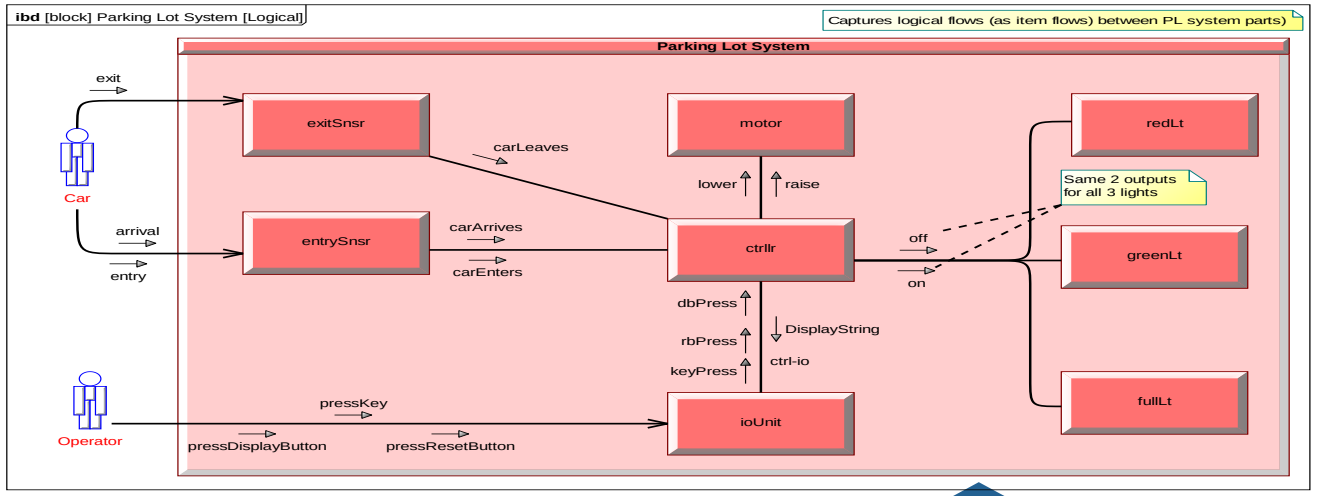
USE CASE REFINEMENT THROUGH SEQUENCE DIAGRAM SCENARIOS



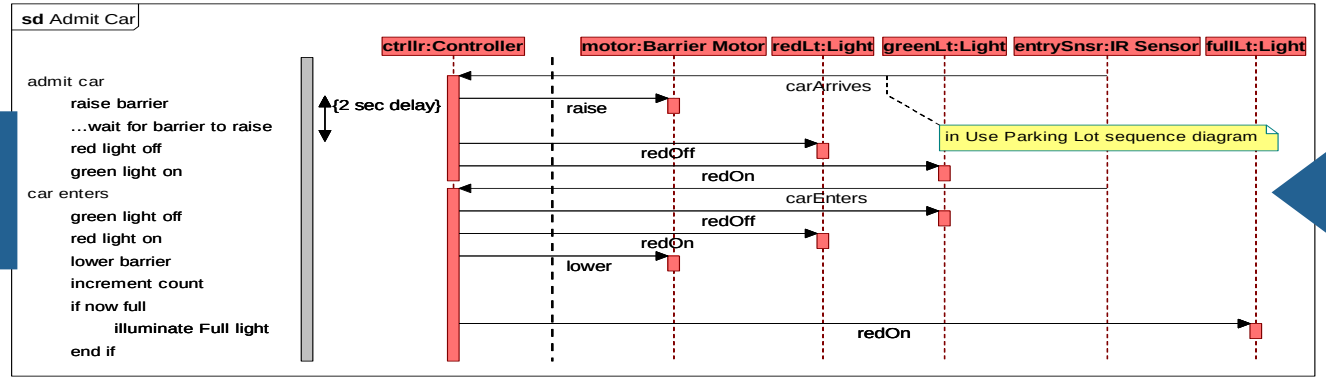
Use Case Description



Candidate Entities



Sequence Diagram



Iterate

STATE MODELING

STATE MACHINES - OVERVIEW

- Most embedded systems are state-based in behavior:
  - Even the simplest system will exhibit states of:
    - Initialization
    - '*Doing Something*'
    - Shutting Down
  - More complex embedded systems may have:
    - Multiple sequential states (the system does multiple things in a specific order).
    - Multiple concurrent states (the system does multiple things all at the same time).
    - A mixture of the above…

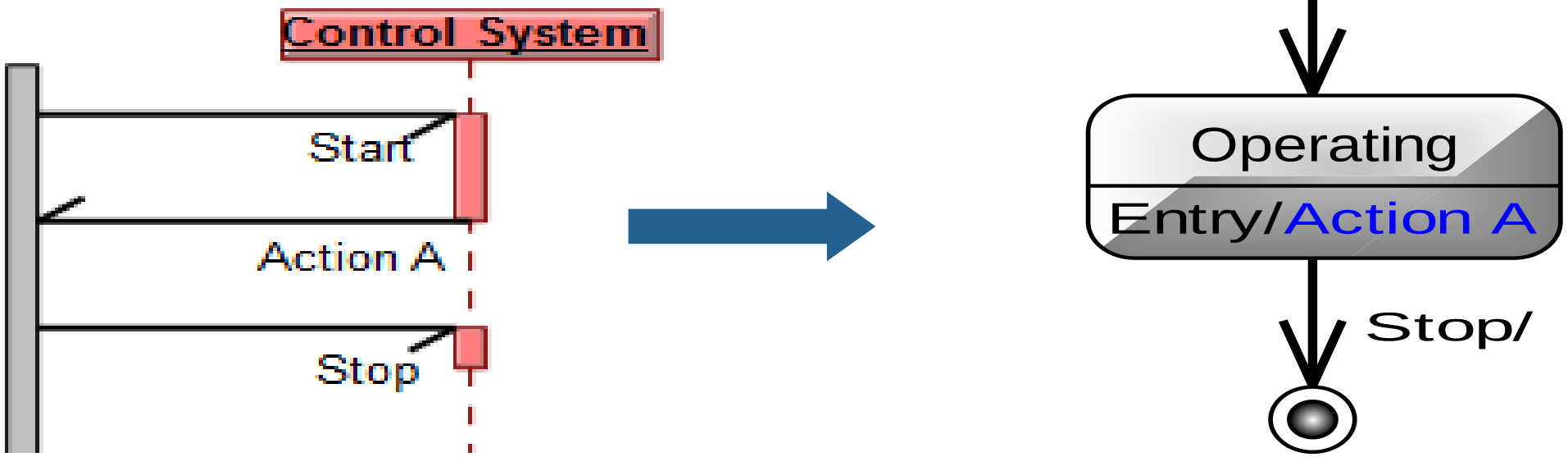
- How do you document the required states of a system or one of its components?
  - Use a State Machine with a State Machine Diagram.
  - Identical to UML state machine semantics and notation.
  - State Machine is a child of a parent block
  - State Machine Diagram is a child of State Machine

STATE MACHINE DIAGRAM MODELING ELEMENTS

- States
- Initial Pseudostates and Final States
- Transitions
- Events
- Guards
- Effects

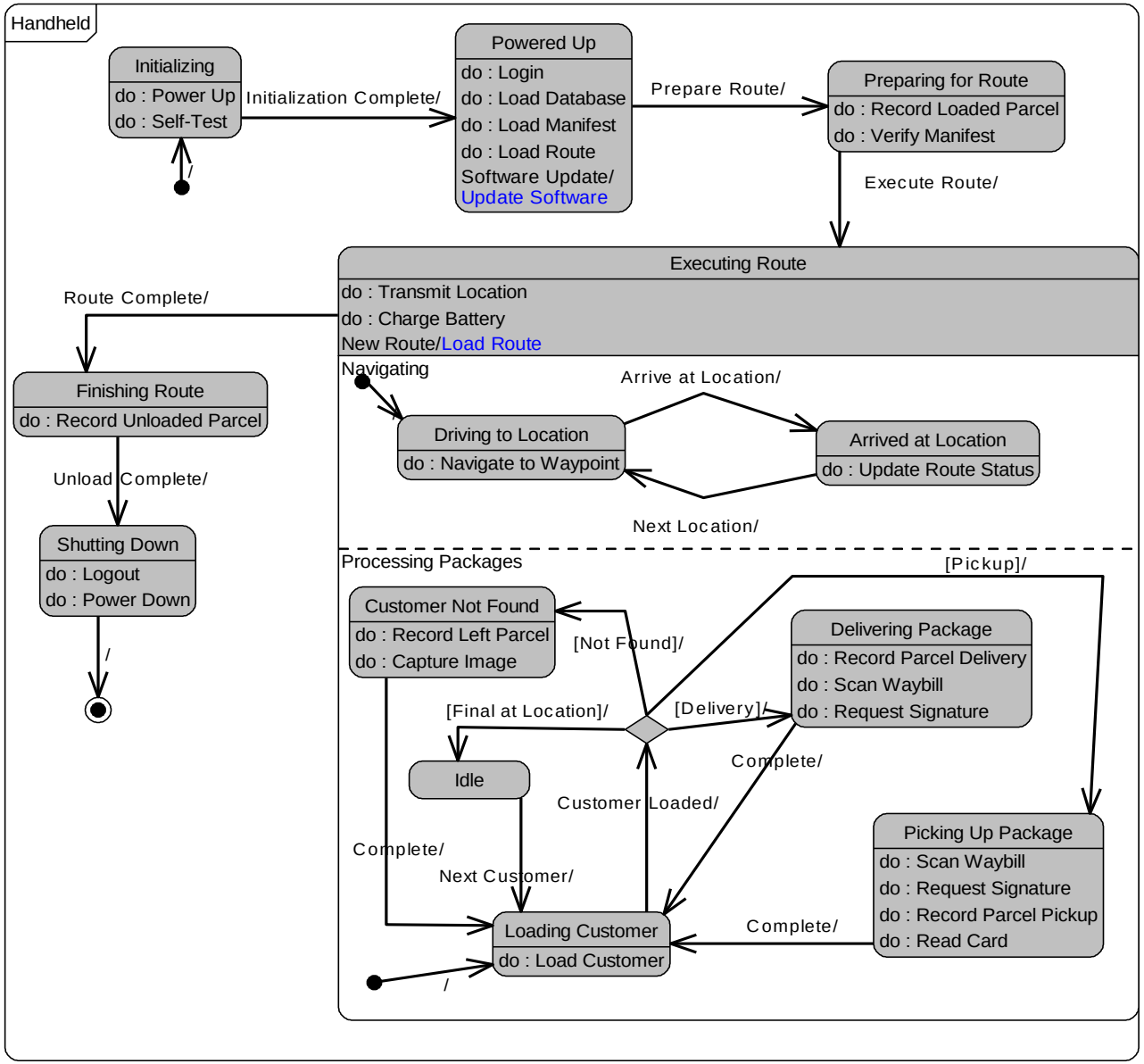
STATES SPECIFY RESPONSES TO EVENTS

- Represent a 'between events' condition of their parent object (block or part)
  - events may be external or internal
- Have duration of a finite interval of time
- May involve processing activity



Work in groups to define the operational states for the handheld device

STATE MACHINES



PARAMETRIC MODELING

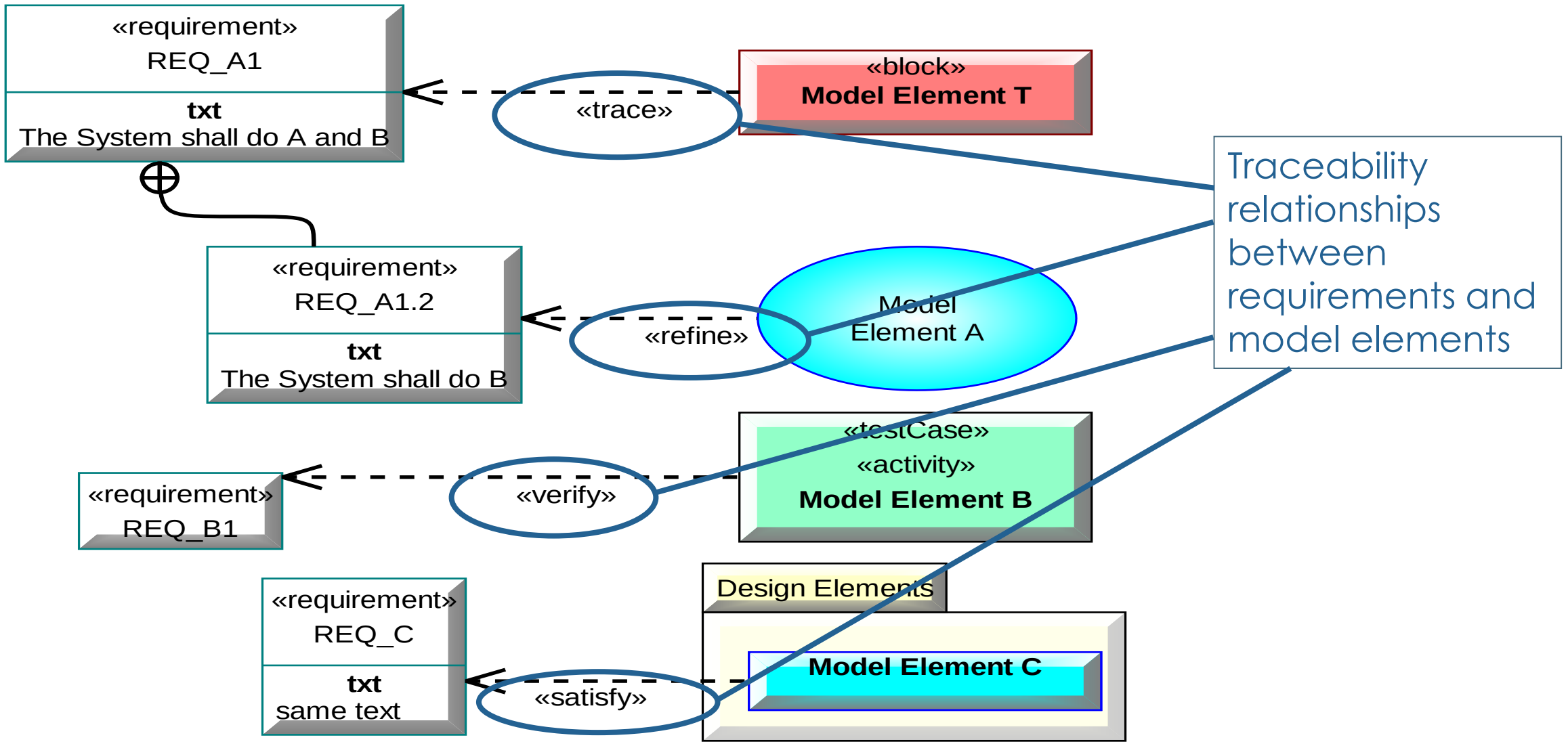
PARAMETRIC DIAGRAMS

- Provide support for engineering trade-off analysis (e.g. vehicle performance v. economy)
- Used to show relationships between (formal) constraint parameters and (actual) block value properties
- Child diagram of either Block or Constraint Block
- Constraint block defines constraint equation and its parameters (on BDD)
- Parametric diagram represents the usage of the constraint equation parameters in a specific analysis context
  - Binding of constraint parameters to value properties of blocks

Parametrics Enable Integration of Engineering Analysis with Design Models

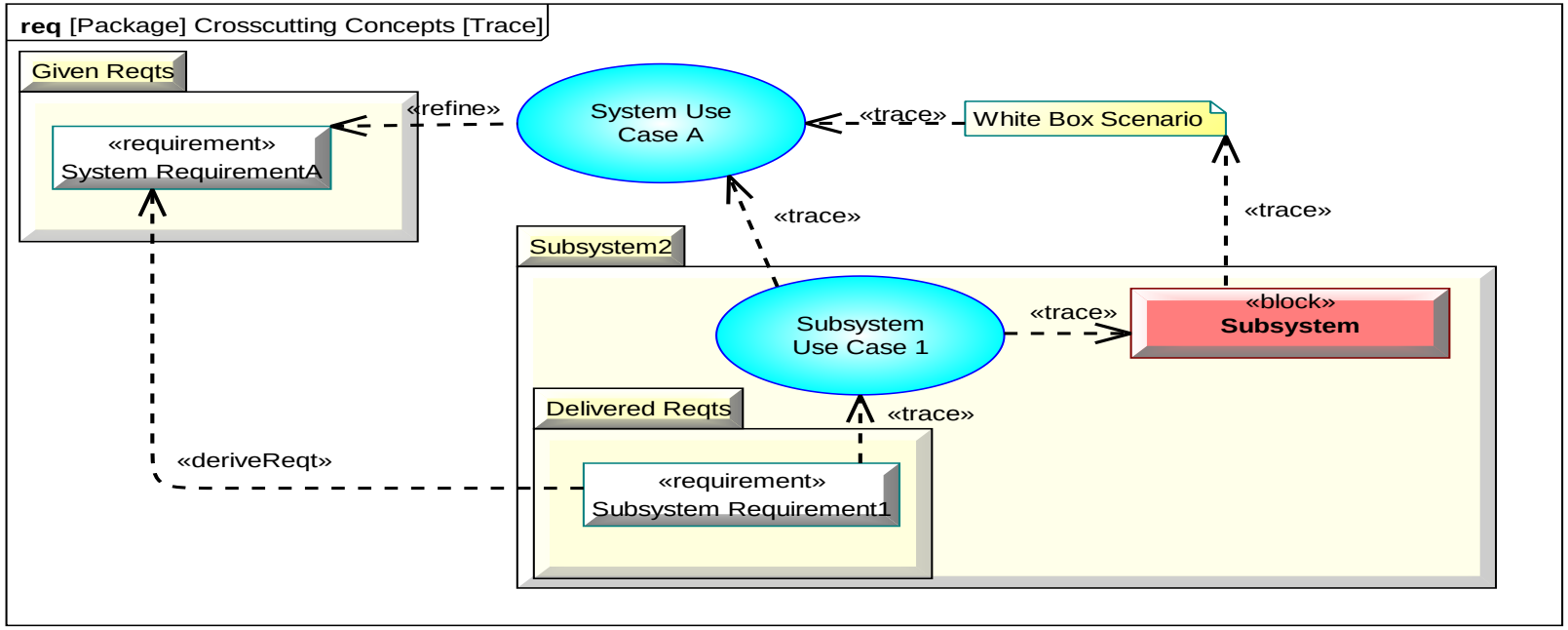
CROSS CUTTING CONCERNS

SYSML REQUIREMENTS TRACEABILITY

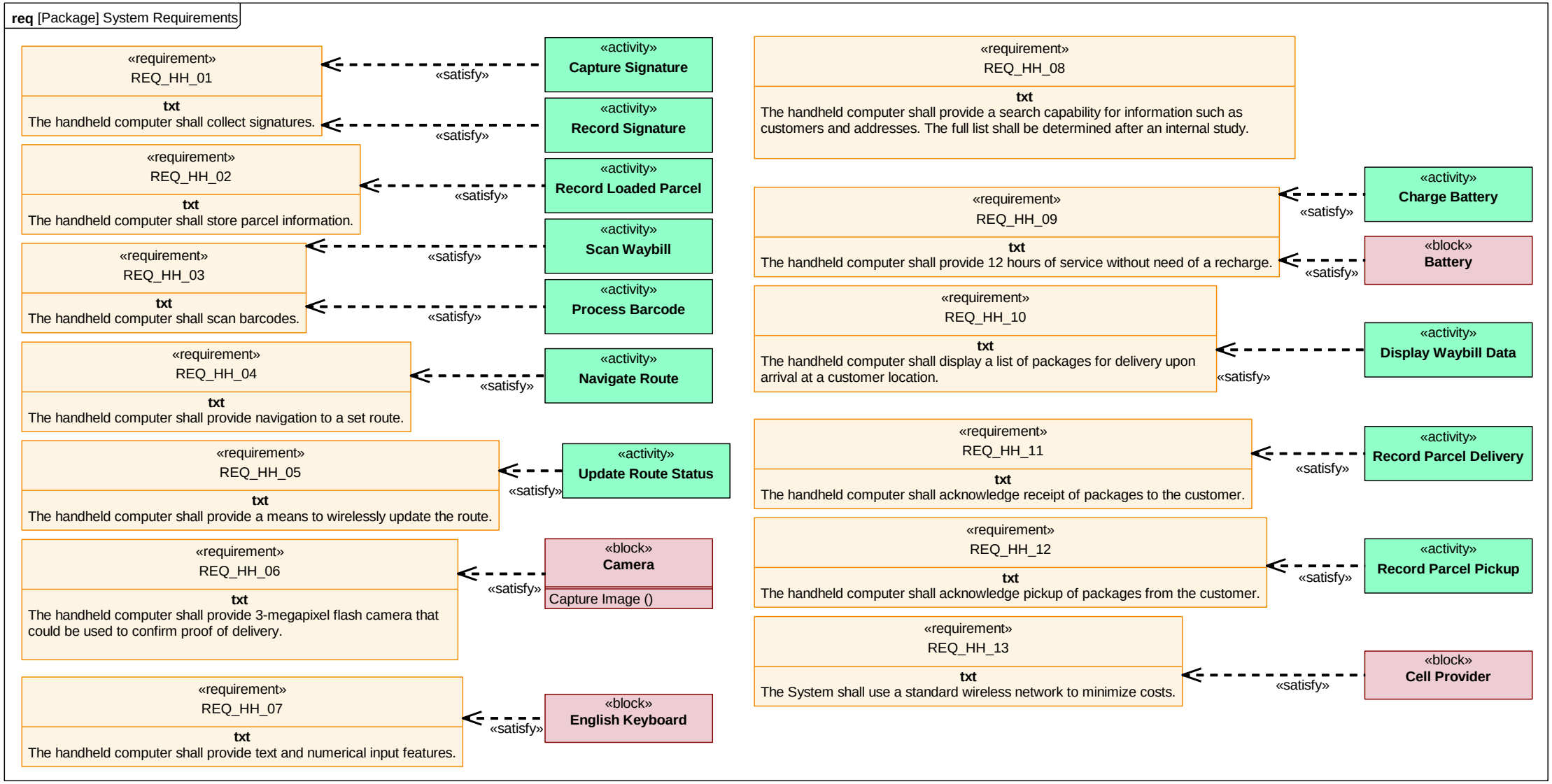


EXPLICIT TRACEABILITY WITHIN MODEL

- <<trace>> not limited to pointing towards requirements
  - Pretty much any model item types at either end
  - Diagram shows path from higher to lower level requirement
    - Explicitly via other model items – more usually some hops left implicit
  - Derive Requirement link added last
    - Inspection of explicit (and usually implicit too) links triggers this



SYSML REQUIREMENTS TRACEABILITY EXAMPLE



SYSML REQUIREMENTS TRACEABILITY TABLE

[Package] System Requirements [1]

Type: Requirement Table (rt)

Description:

Name	txt	rationale	satisfiedBy
REQ_HH_01	The handheld computer shall collect signatures.		«Activity» Capture Signature (Handheld System::Behavior::Activities) «Activity» Record Signature (Handheld System::Behavior::Activities)
REQ_HH_02	The handheld computer shall store parcel information.		«Activity» Record Loaded Parcel (Handheld System::Behavior::Activities)
REQ_HH_03	The handheld computer shall scan barcodes.		«Activity» Process Barcode (Handheld System::Behavior::Activities) «Activity» Scan Waybill (Handheld System::Behavior::Activities)
REQ_HH_04	The handheld computer shall provide navigation to a set route.		«Activity» Navigate Route (Handheld System::Behavior::Activities)
REQ_HH_05	The handheld computer shall provide a means to wirelessly update the route.		«Activity» Update Route Status (Handheld System::Behavior::Activities)
REQ_HH_06	The handheld computer shall provide 3-megapixel flash camera that could be used to confirm proof of delivery.		«Block» Camera (Handheld System::Structure::Handheld)
REQ_HH_07	The handheld computer shall provide text and numerical input features.		«Block» English Keyboard (Handheld System::Structure::Handheld)
REQ_HH_08	The handheld computer shall provide a search capability for information such as customers and addresses. The full list shall be determined after an internal study.		
REQ_HH_09	The handheld computer shall provide 12 hours of service without need of a recharge.		«Block» Battery (Handheld System::Structure::Handheld) «Activity» Charge Battery (Handheld System::Behavior::Activities)
REQ_HH_10	The handheld computer shall display a list of packages for delivery upon arrival at a customer location.		«Activity» Display Waybill Data (Handheld System::Behavior::Activities)
REQ_HH_11	The handheld computer shall acknowledge receipt of packages to the customer.		«Activity» Record Parcel Delivery (Handheld System::Behavior::Activities)
REQ_HH_12	The handheld computer shall acknowledge pickup of packages from the customer.		«Activity» Record Parcel Pickup (Handheld System::Behavior::Activities)
REQ_HH_13	The System shall use a standard wireless network to minimize costs.		«Block» Cell Provider (System Network Model::Structure::Components) «Block» Cell Provider (Handheld System::Structure::Logical::Context)

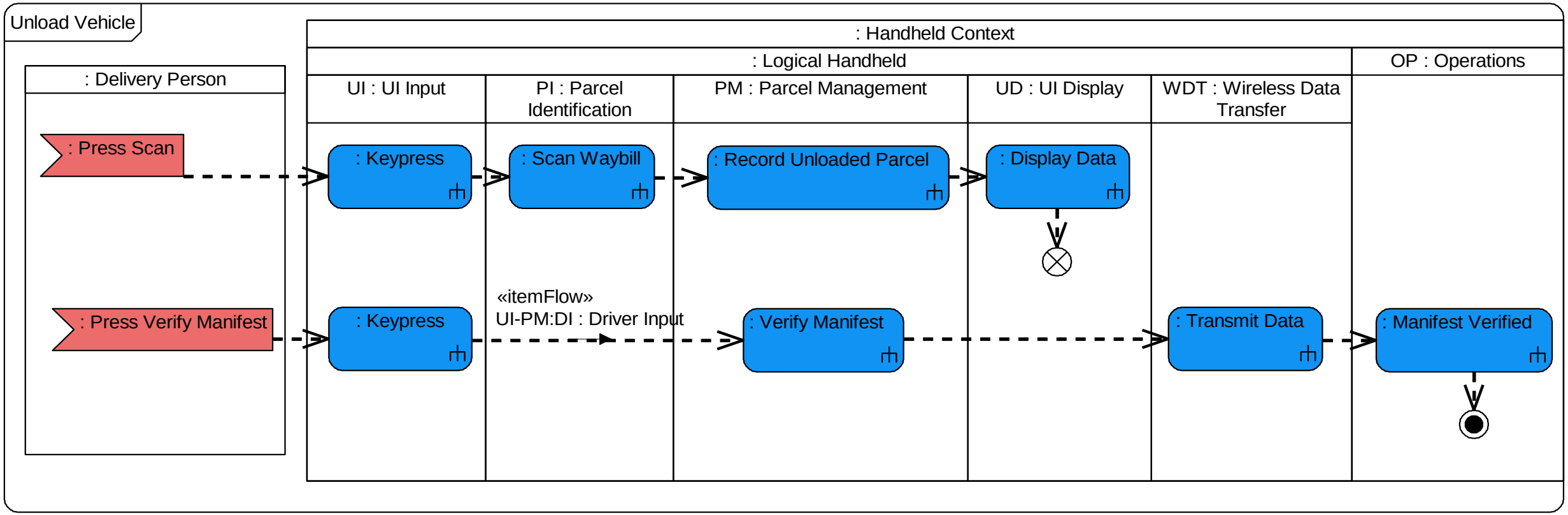
Automatic creation of table with configurable columns

ALLOCATIONS

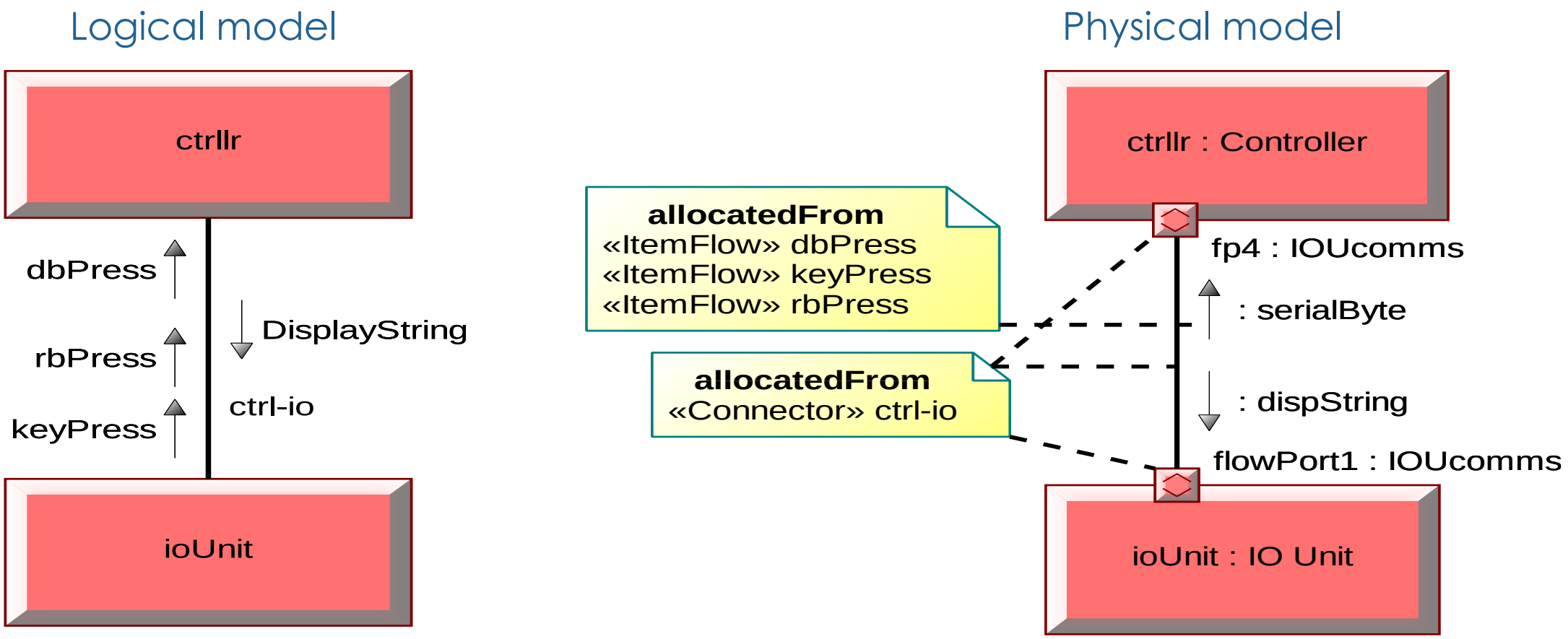
- Flexible, directed relationships that link model elements
  - Abstract dependency
  - Often precursor to later, more concrete relationship
- Different types of allocation including:
  - Behavioral (Behavior to Structure, e.g. activity to block, action to part)
  - Flow (sub-type of Behavioral, e.g. IBD connector to activity edge)
  - Structural (Abstract to Concrete or Logical to Physical)
  - Software to Hardware
  - etc.
- Both diagrammatic and tabular representations are specified
- Facilitates model navigation

ALLOCATION EXAMPLE – BEHAVIORAL

- Placement of Activities within partition automatically allocates activity to part
- Flow from IBD allocated to control flow



ALLOCATION EXAMPLE - STRUCTURAL



SYSML ALLOCATIONS TRACEABILITY TABLE

[Package] Handheld System [Allocation Table]

Allocated From	Relation	Allocated To
«Accept Event Action» Access (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Acknowledge Receipt (Handheld System::Behavior::Activities::Pickup Parcel)	Allocate	«Actor» Customer (Handheld System::Behavior::Actors)
«Accept Event Action» Acknowledge Receipt (Handheld System::Behavior::Activities::Deliver Parcel)	Allocate	«Actor» Customer (Handheld System::Behavior::Actors)
«Call Behavior Action» KP1 (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«BlockProperty» UI (Handheld System::Structure::Logical::Systems::Logical Handheld)
«Call Behavior Action» KP2 (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«BlockProperty» UI (Handheld System::Structure::Logical::Systems::Logical Handheld)
«Call Behavior Action» KP3 (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«BlockProperty» UI (Handheld System::Structure::Logical::Systems::Logical Handheld)
«Accept Event Action» Press Verify Manifest (Handheld System::Behavior::Activities::Unload Vehicle)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Press Verify Manifest (Handheld System::Behavior::Activities::Load Vehicle)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Request Database (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Request Manifest (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Request Route (Handheld System::Behavior::Activities::Access Handheld)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Scan Press (Handheld System::Behavior::Activities::Load Vehicle)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Scan Press (Handheld System::Behavior::Activities::Unload Vehicle)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Scan Request (Handheld System::Behavior::Activities::Pickup Parcel)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Scan Request (Handheld System::Behavior::Activities::Deliver Parcel)	Allocate	«Actor» Delivery Person (Handheld System::Behavior::Actors)
«Accept Event Action» Sign (Handheld System::Behavior::Activities::Deliver Parcel)	Allocate	«Actor» Customer (Handheld System::Behavior::Actors)
«Accept Event Action» Sign (Handheld System::Behavior::Activities::Pickup Parcel)	Allocate	«Actor» Customer (Handheld System::Behavior::Actors)

SYSML ALLOCATIONS TRACEABILITY MATRIX (FRAGMENT)

[Package] Handheld System [Allocation Matrix]

Allocated To

	«Accept Event Action» Access Handheld	System::Behavior::Actions::AcceptEventAction::Acknowledge Receipt Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld	System::Behavior::Actions::AcceptEventAction::Deliver Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld	System::Behavior::Actions::AcceptEventAction::Unload Handheld	System::Behavior::Actions::AcceptEventAction::Access Handheld	System::Behavior::Actions::AcceptEventAction::Load Handheld	System::Behavior::Actions::AcceptEventAction::Deliver Handheld	System::Behavior::Actions::AcceptEventAction::Deliver Handheld	System::Behavior::Actions::AcceptEventAction::Access Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld	System::Behavior::Actions::AcceptEventAction::Access Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld	System::Behavior::Actions::AcceptEventAction::Deliver Handheld	System::Behavior::Actions::AcceptEventAction::Deliver Handheld	System::Behavior::Actions::AcceptEventAction::Access Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld	System::Behavior::Actions::AcceptEventAction::Access Handheld	System::Behavior::Actions::AcceptEventAction::Pickup Handheld
«Actor» Customer (Handheld System::Behavior::Actors)		X	X																	
«Actor» Delivery Person (Handheld System::Behavior::Actors)	X																			
OP (Handheld System::Structure::Logical::Context::Handhel																				
CDT (Handheld System::Structure::Logical::Systems::Logical										X		X								
PI (Handheld System::Structure::Logical::Systems::Logical																				
PM (Handheld System::Structure::Logical::Systems::Logical						X			X		X				X	X			X	X
UD (Handheld System::Structure::Logical::Systems::Logical																	X	X		
UI (Handheld System::Structure::Logical::Systems::Logical				X	X		X	X												
WDT (Handheld System::Structure::Logical::Systems::Logical													X	X						

SYSML AND PRODUCT LINE ENGINEERING

ENHANCING MBSE WITH PRODUCT LINE ENGINEERING (PLE)

- The Goal of PLE is to Reduce the Time, Cost and Effort required to Create, Deploy and Maintain Similar Products
  - By Leveraging Product & System Commonality
  - And Designed-in Variation (More than just Asset Reuse)

- To achieve this goal, the solution must
  - Minimize duplicate effort
  - Maximize commonality
  - Optimize reuse across similar products
  - Manage product line variations and complexity

- Model Based PLE offers a fundamental shift in approach
  - "A broader perspective that views product line engineering as designing a single system rather than as creating a multitude of products"

- Designing your products as a single system can deliver considerable development cost savings (Dr Jerry Krasner, EMF 2013)

1 - VARIANT MODELING

- Variant Diagram
  - Variation on all Diagrams
  - Simple Notation

Variation Point

Variant

Variability Dependency

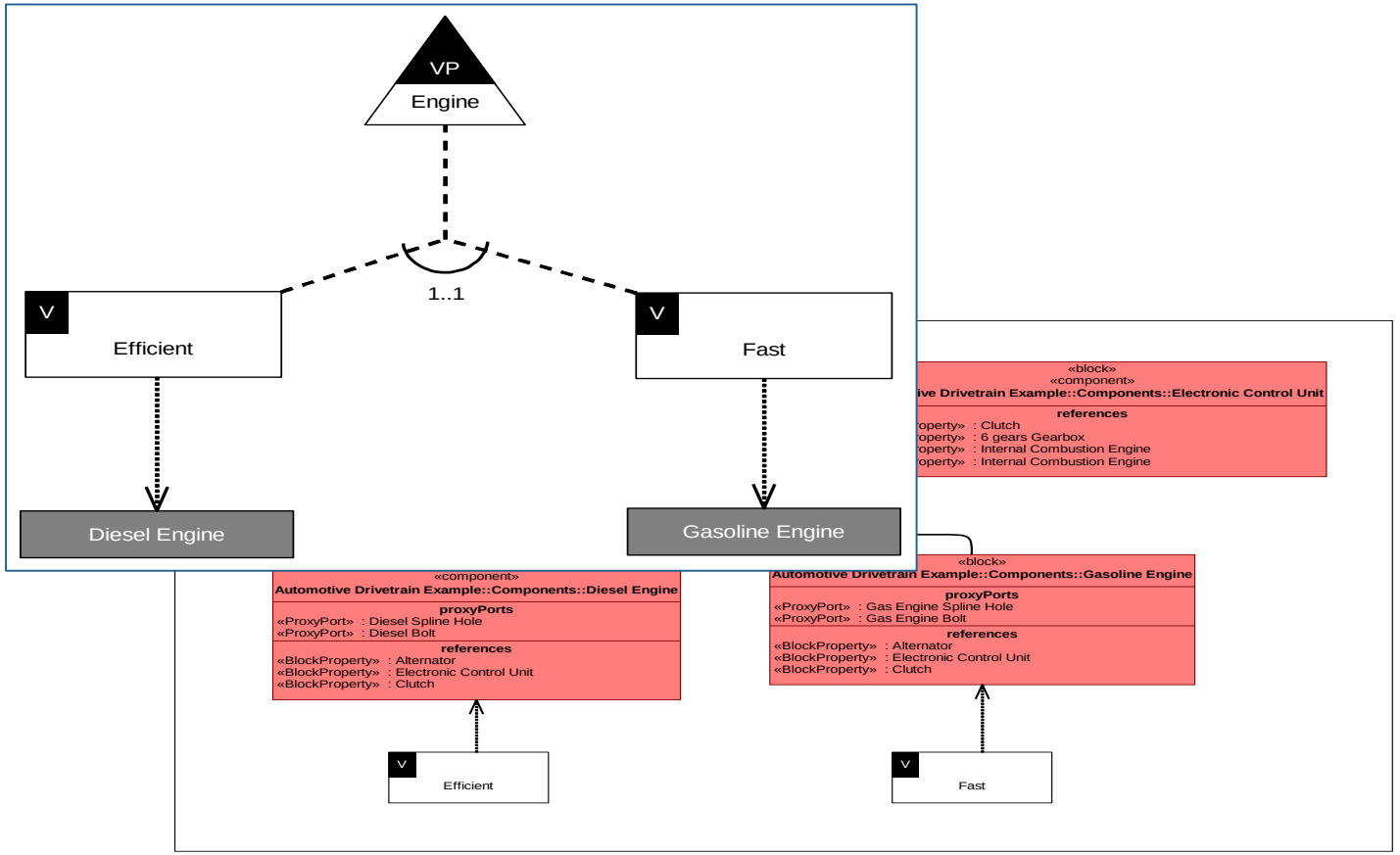
Mandatory/Optional

Requires Dependency

Excludes Dependency

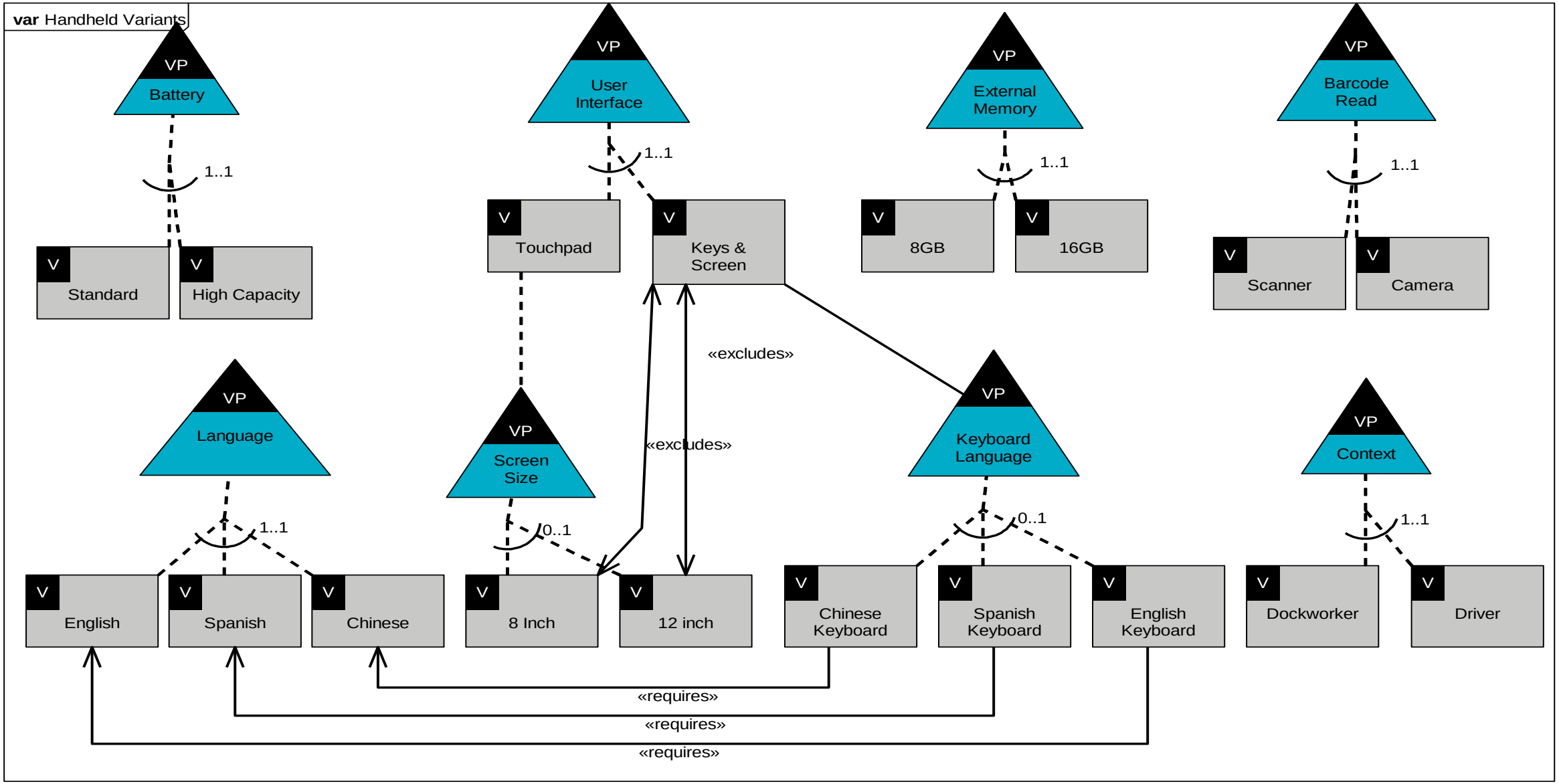
Artifact Dependency

Alternate Choice

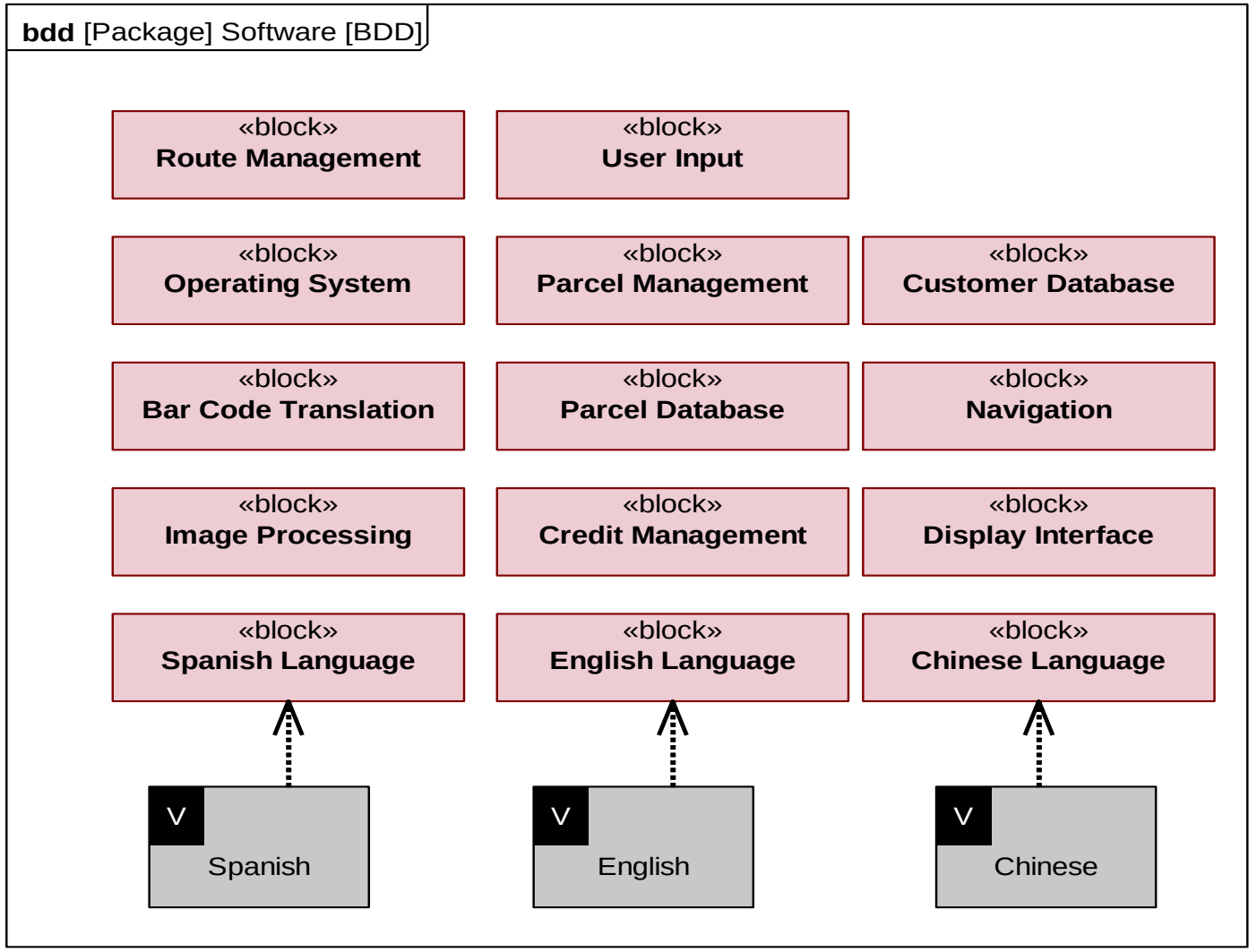


- OVM
  - PALUNO, The Ruhr Institute of Software Technology
  - Software Product Line Engineering (Pohl et al - Springer 2005)

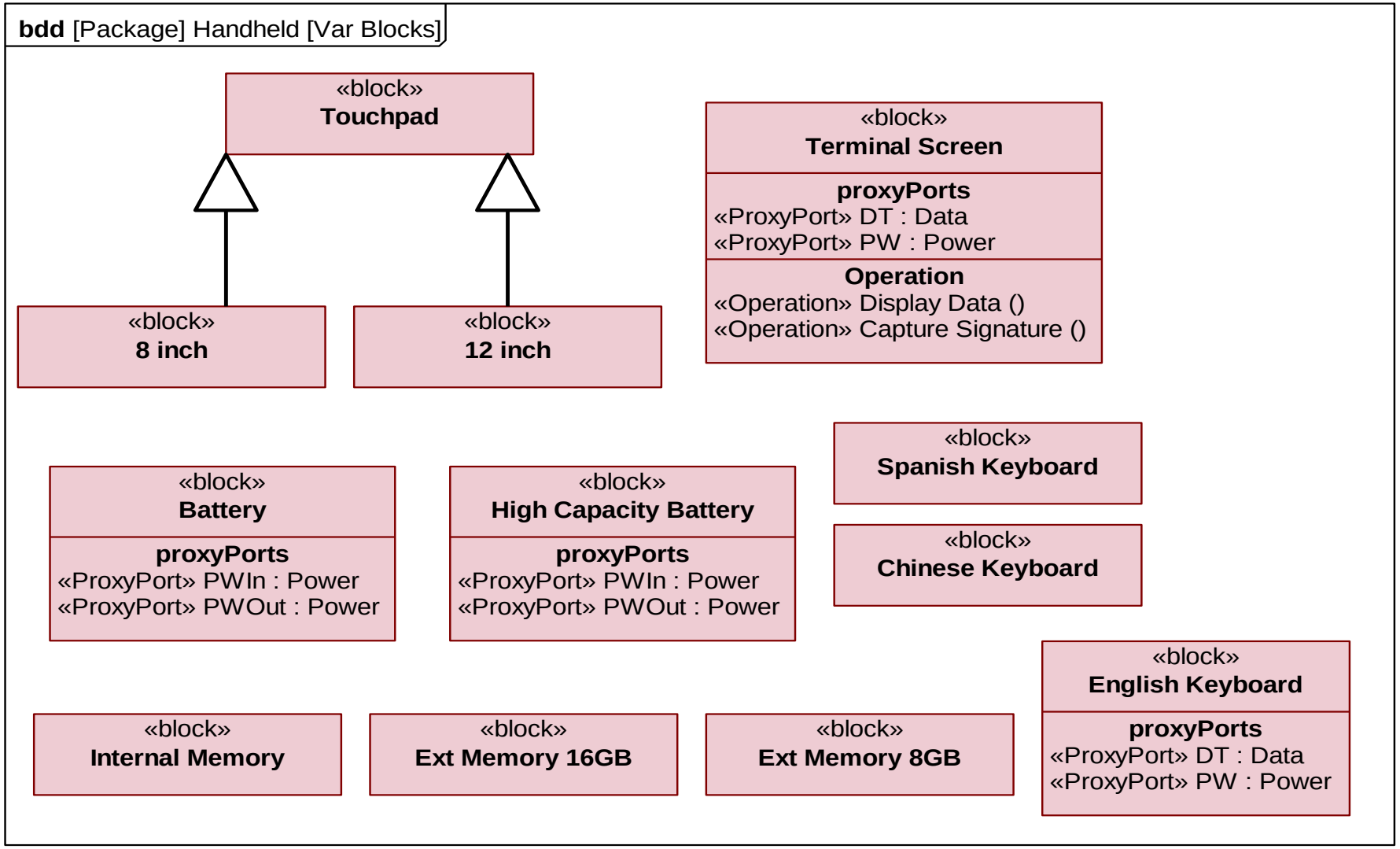
HANDHELD VARIANTS



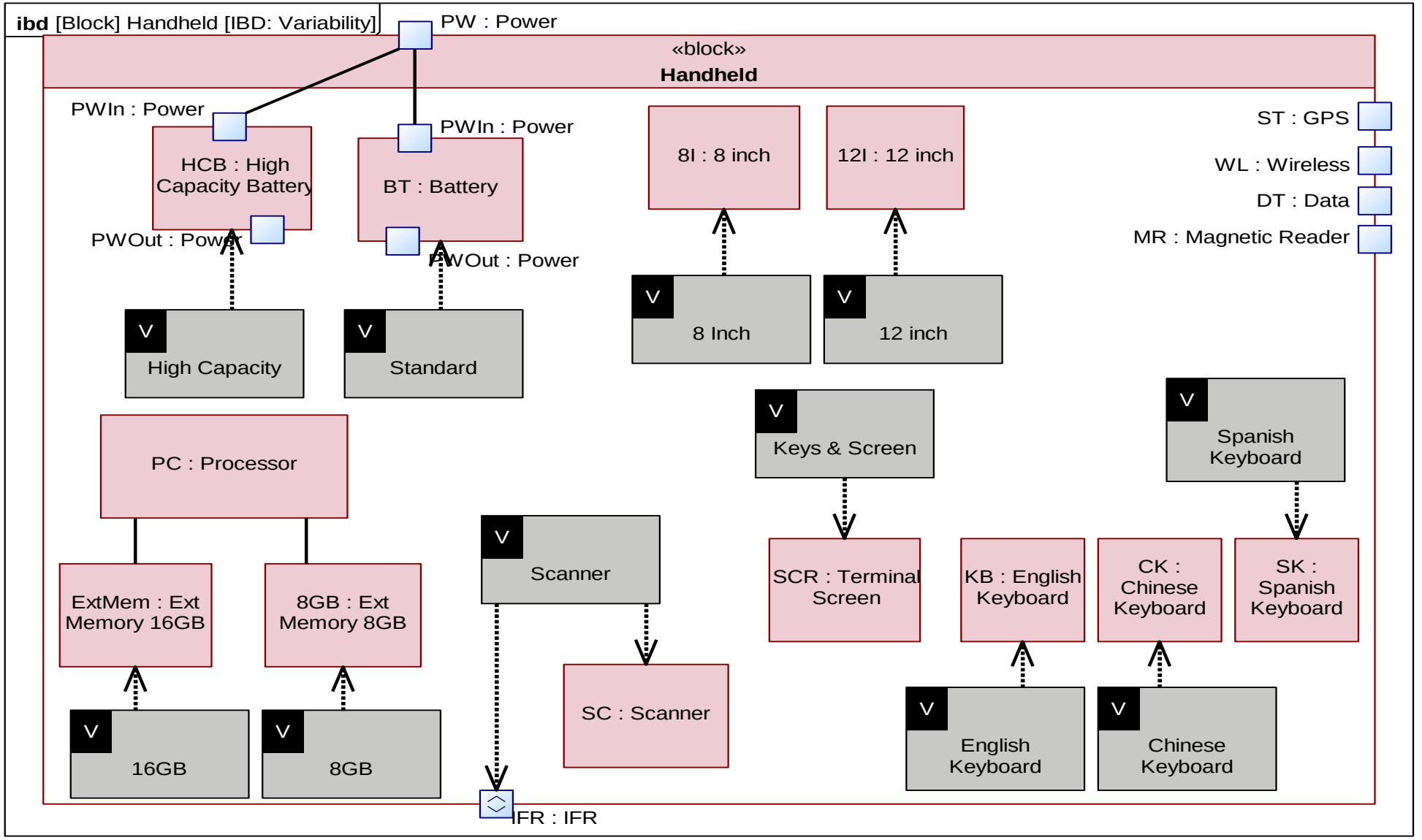
LANGUAGE VARIANTS



STRUCTURE


bdd [Package] Handheld [Var Blocks]
«block»
Touchpad
«block»
8 inch
«block»
12 inch
«block»
Terminal Screen
proxyPorts
«ProxyPort» DT : Data
«ProxyPort» PW : Power
Operation
«Operation» Display Data ()
«Operation» Capture Signature ()
«block»
Battery
proxyPorts
«ProxyPort» PWIn : Power
«ProxyPort» PWOut : Power
«block»
High Capacity Battery
proxyPorts
«ProxyPort» PWIn : Power
«ProxyPort» PWOut : Power
«block»
Spanish Keyboard
«block»
Chinese Keyboard
«block»
English Keyboard
proxyPorts
«ProxyPort» DT : Data
«ProxyPort» PW : Power
«block»
Internal Memory
«block»
Ext Memory 16GB
«block»
Ext Memory 8GB

VARIANTS WITH MODEL ELEMENTS

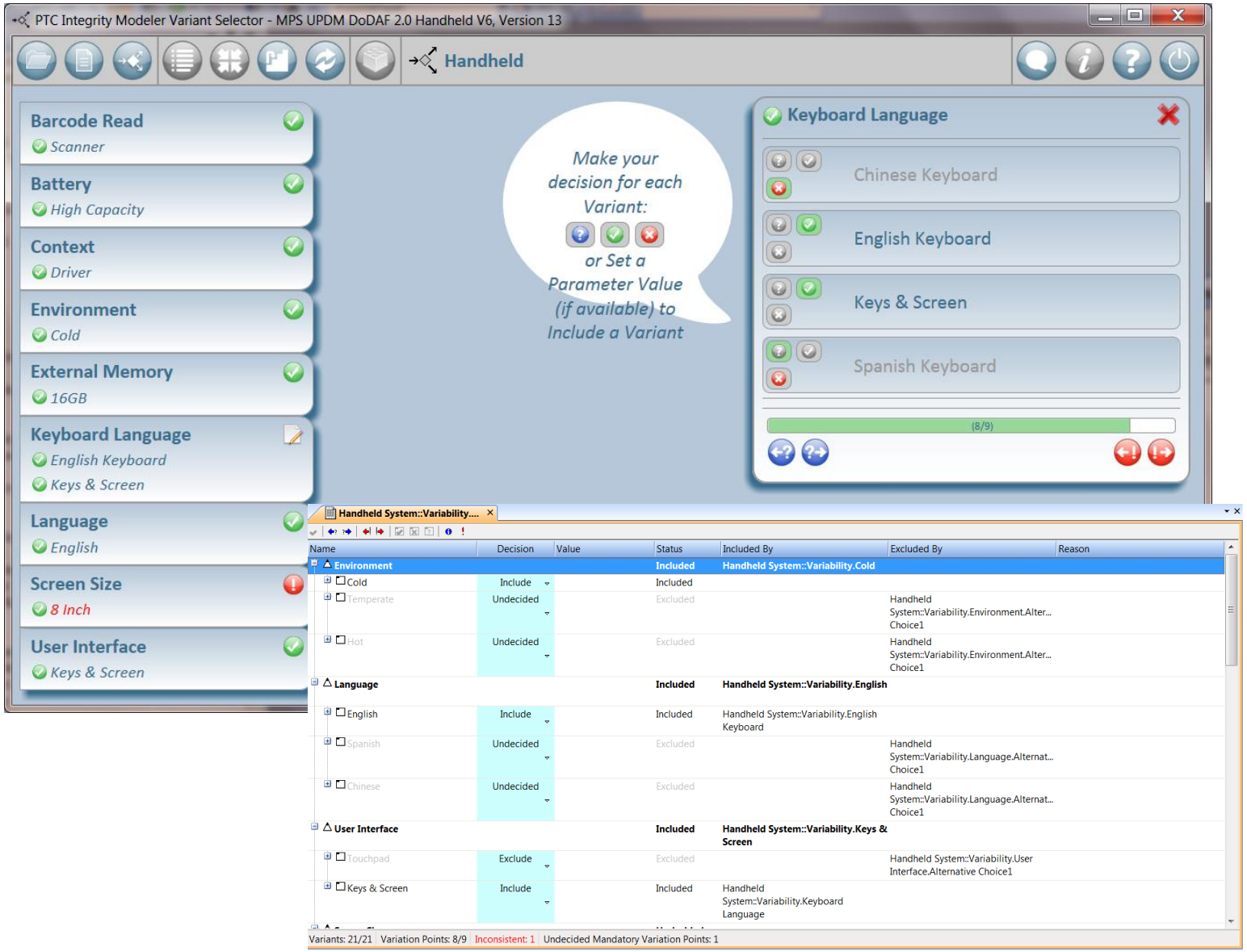


2 - VARIANT SELECTION

- Variant Selector
  - Browser User Interface
  - External Variation Points Only
  - Jump to Next Decision/Problem
  - Progress Bar

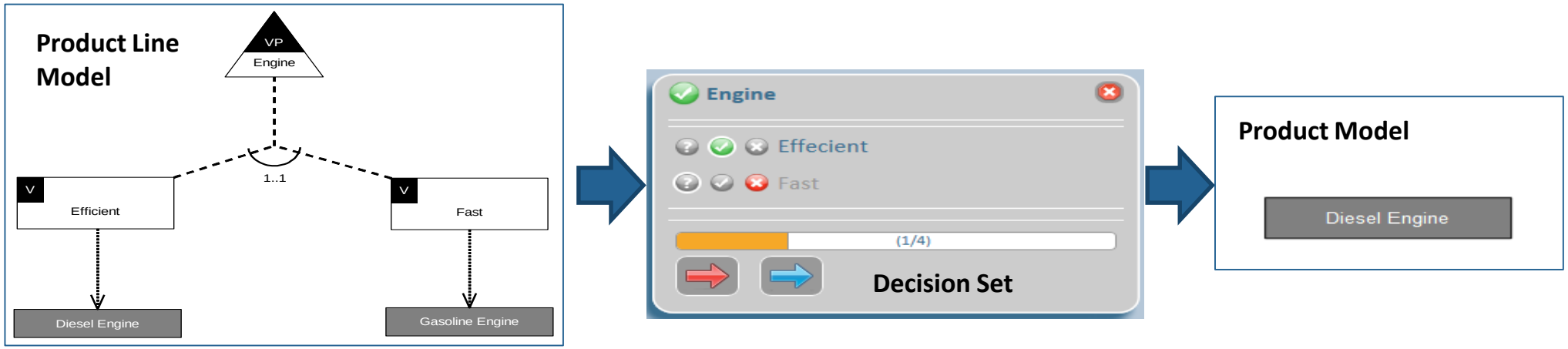
- Decision Set Editor
  - Variant Debug
  - External & Internal Variation Points
  - Jump to Next Decision/Problem

■ Both Edit the Same Decision Sets



3 - PRODUCT MODEL CREATION

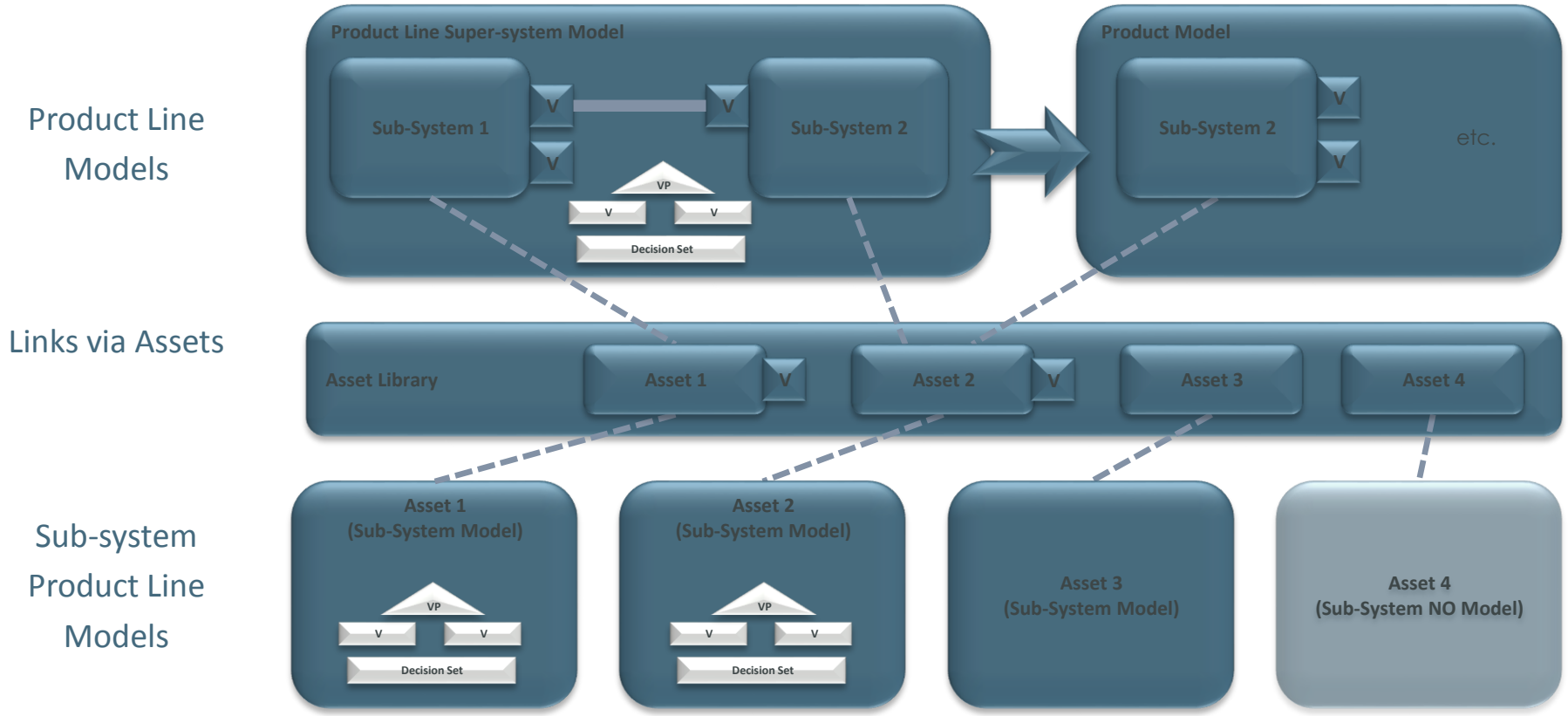
- Auto-Create Product Models
  - Applies Variability Decisions
  - Unnecessary Variation Points, Variants & Base Model Artefacts Removed



- Creates New Product Model Branch, Original Product Line Model Retained
- Product Model suitable for Trade Studies, Simulation, Documentation & Generation

MODEL- BASED PRODUCT LINE ENGINEERING

- Integrated MBSE, Modular Design & Variability Modeling = Model-based Product Line Engineering

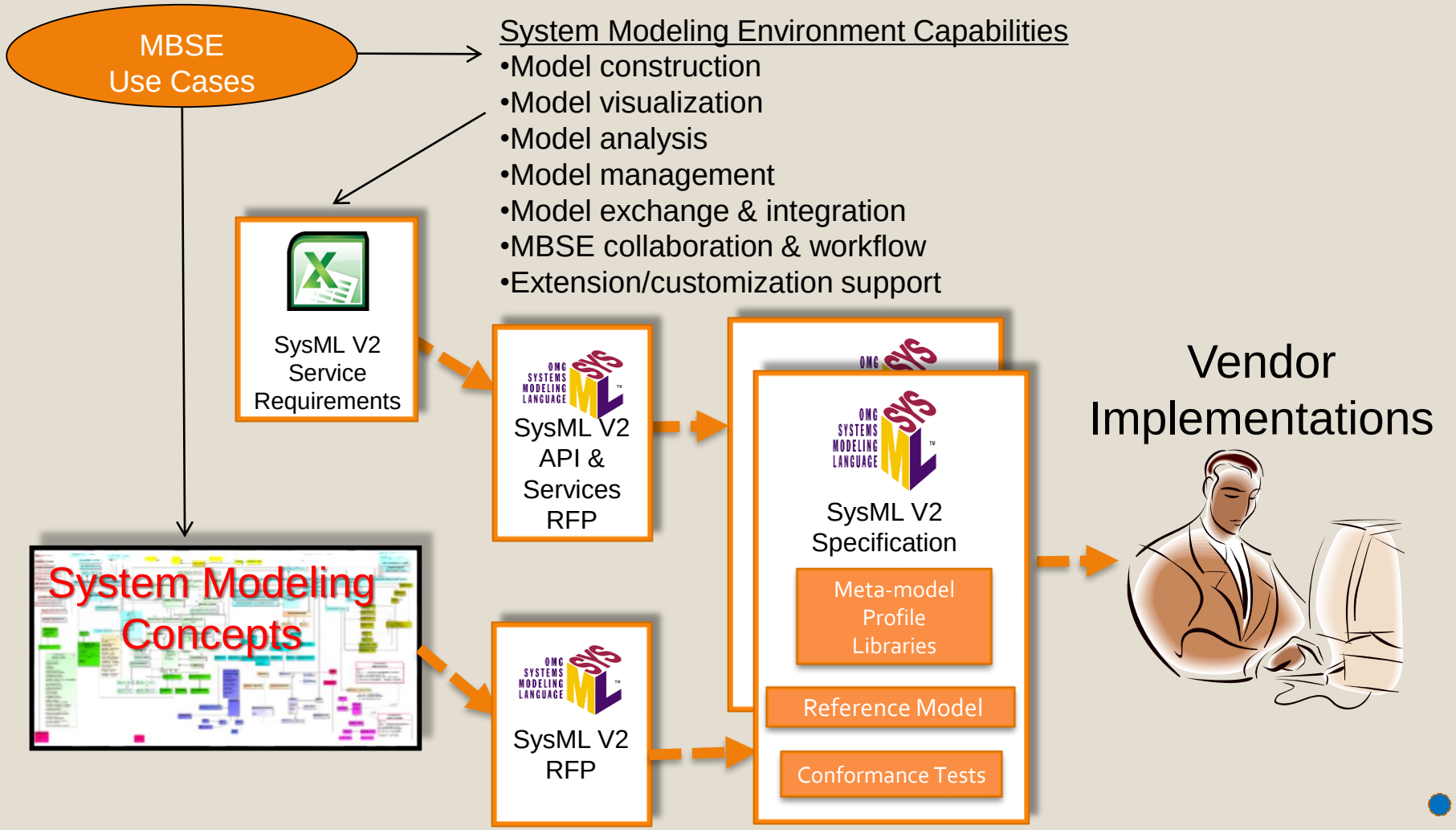


SYSML V2

SysML v2 Objectives

- Facilitate increased adoption and effectiveness of MBSE over SysML v1 through enhanced:
  - Precision & expressiveness
  - Integration among the language concepts
  - Interoperability with other engineering models and tools
  - Usability by model developers and consumers

SysML v2 Specification Development



SysML v2 RFP (Draft)
Requirements in Section 6.5

OMG Document #: syseng-2017-09-04

6.5 Mandatory Requirements

6.5.1 Language Architecture

6.5.2 Data Model

6.5.2.1 Cross-cutting

6.5.2.2 Properties, Values, and Expressions

6.5.2.3 Structure

6.5.2.4 Interfaces

6.5.2.5 Behavior

6.5.2.6 Requirements

6.5.2.7 Verification & Validation

6.5.2.8 Analysis

6.5.3 Reference Model & Model Libraries

6.5.4 Language Conformance

SysML v2 API and Services RFP (Draft)
Requirements in Sections 6.5 and 6.6

OMG Document #: syseng-2017-09-05

6.5 API Mandatory Requirements

6.5.1 API Requirements

6.5.2 Services Requirements (query, extension services)

6.5.3 API and Service Conformance

6.6 Non Mandatory Features

6.6.1 Model Construction Services

6.6.2 Model Visualization Services

6.6.3 Model Analysis Services

6.6.4 Model Management Services

6.6.5 Workflow and Collaboration Services

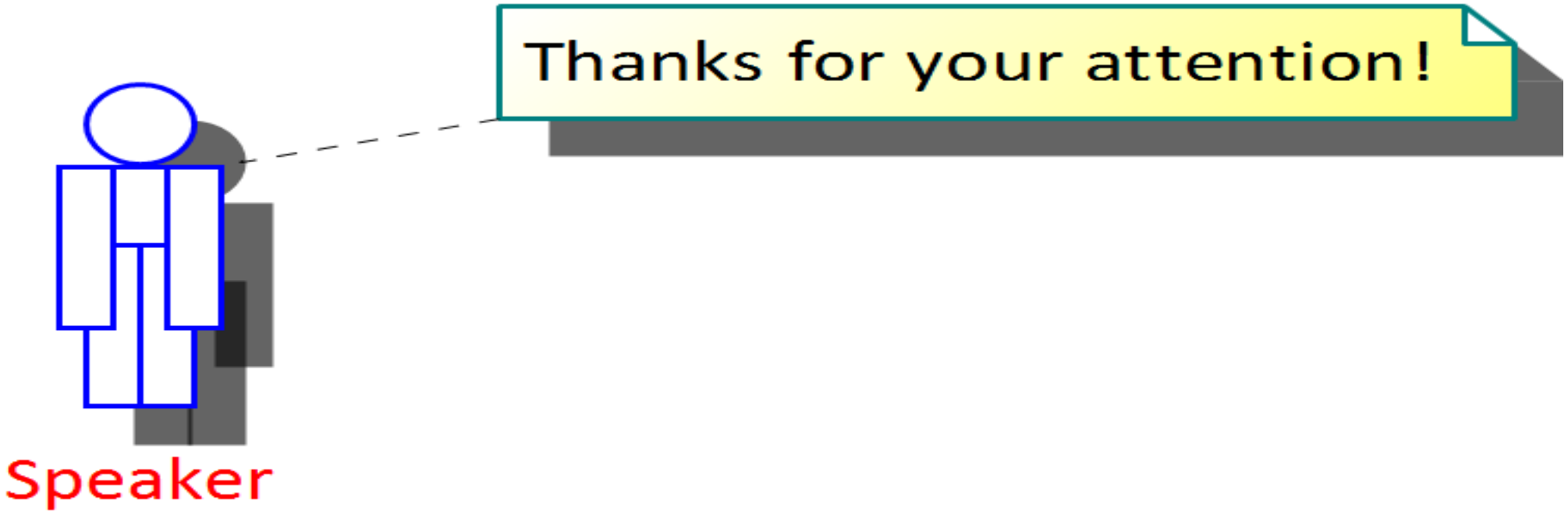
6.6.6 Interoperability Services

SYSTEM DEVELOPMENT WITH SYSML

- SysML is process-independent
  - Can be used and adapted for wide variety of systems engineering processes
- You need to develop 'good practice' guidelines for your process
  - PTC can help with this
- Process/model considerations
  - Give early consideration to the model (package) structure
    - Layered approach? (system, sub-systems, …)
    - Logical and physical models?
    - Structural and behavioral models?
    - Access permissions
  - Co-develop the structural and behavioral model, and their links
    - for a 'layer' in a layered model
  - Review and baseline one layer before moving on to the next layer
    - Consistency, correctness, completeness, etc.

HOWEVER, A FOOL WITH A TOOL, IS STILL A FOOL!

QUESTIONS AND ANSWERS




ptc