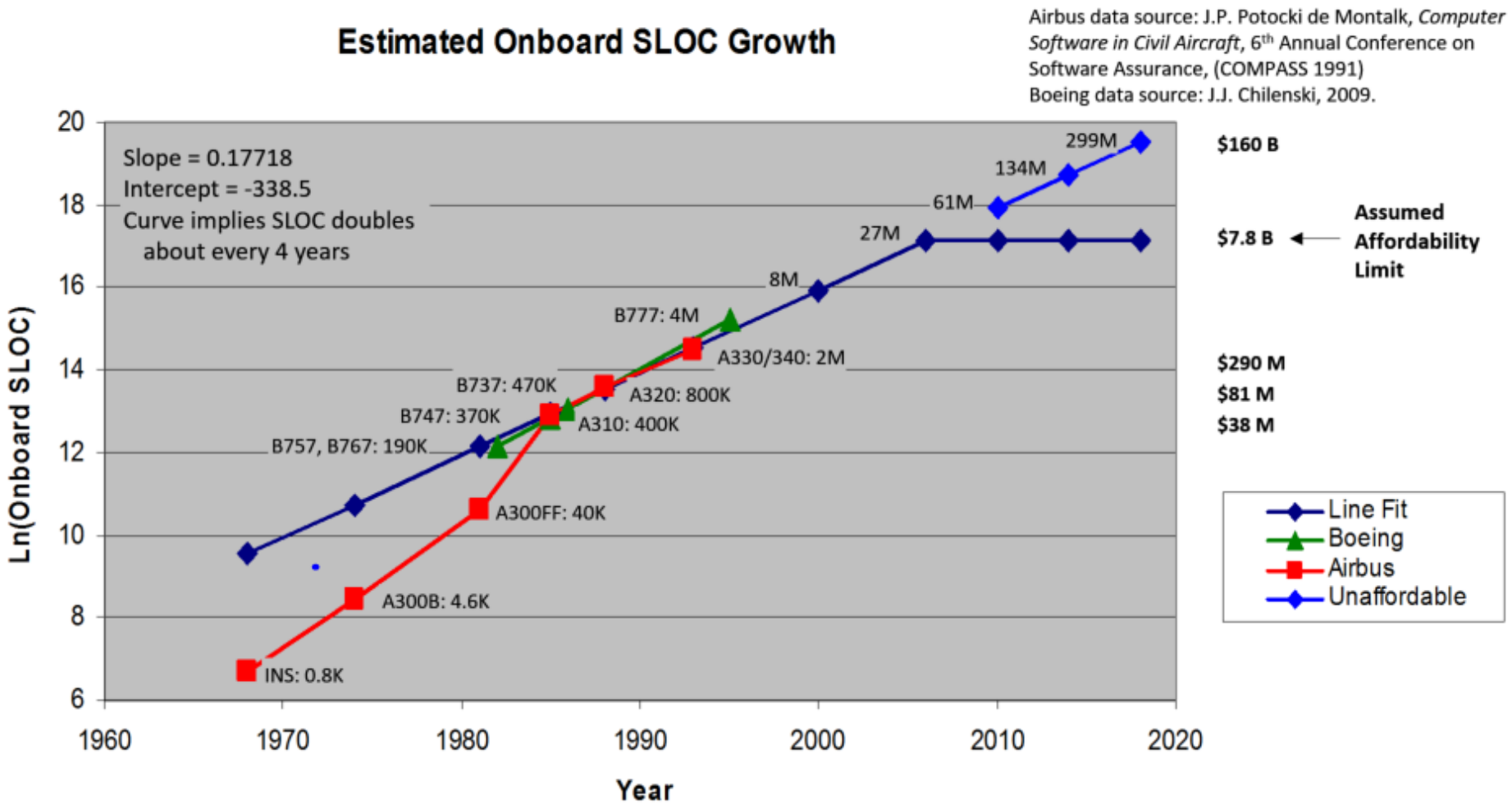
Architecture-Led Requirements Engineering for Cyber-Physical Systems with the Architecture Analysis and Design Language

Dr. Dominique Blouin
Telecom ParisTech, France
dominique.blouin@telecom-paristech.fr

Outline

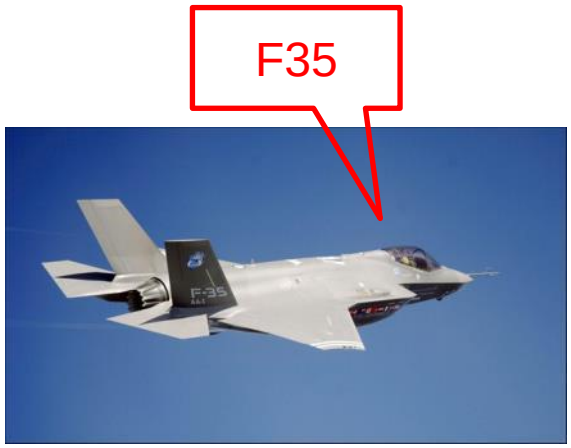
- **Introduction**
- **Model-Based Engineering with AADL**
- **Overview of Line Follower Robot Example**
- **Modeling the Line Follower Robot Example**
- **Conclusion and Perspectives**

Increasing System Complexity: Avionics Domain Example



From Hansson, Helton and Feiler, *ROI Analysis of the System Architecture Virtual Integration Initiative*, SEI Technical Report, 2018

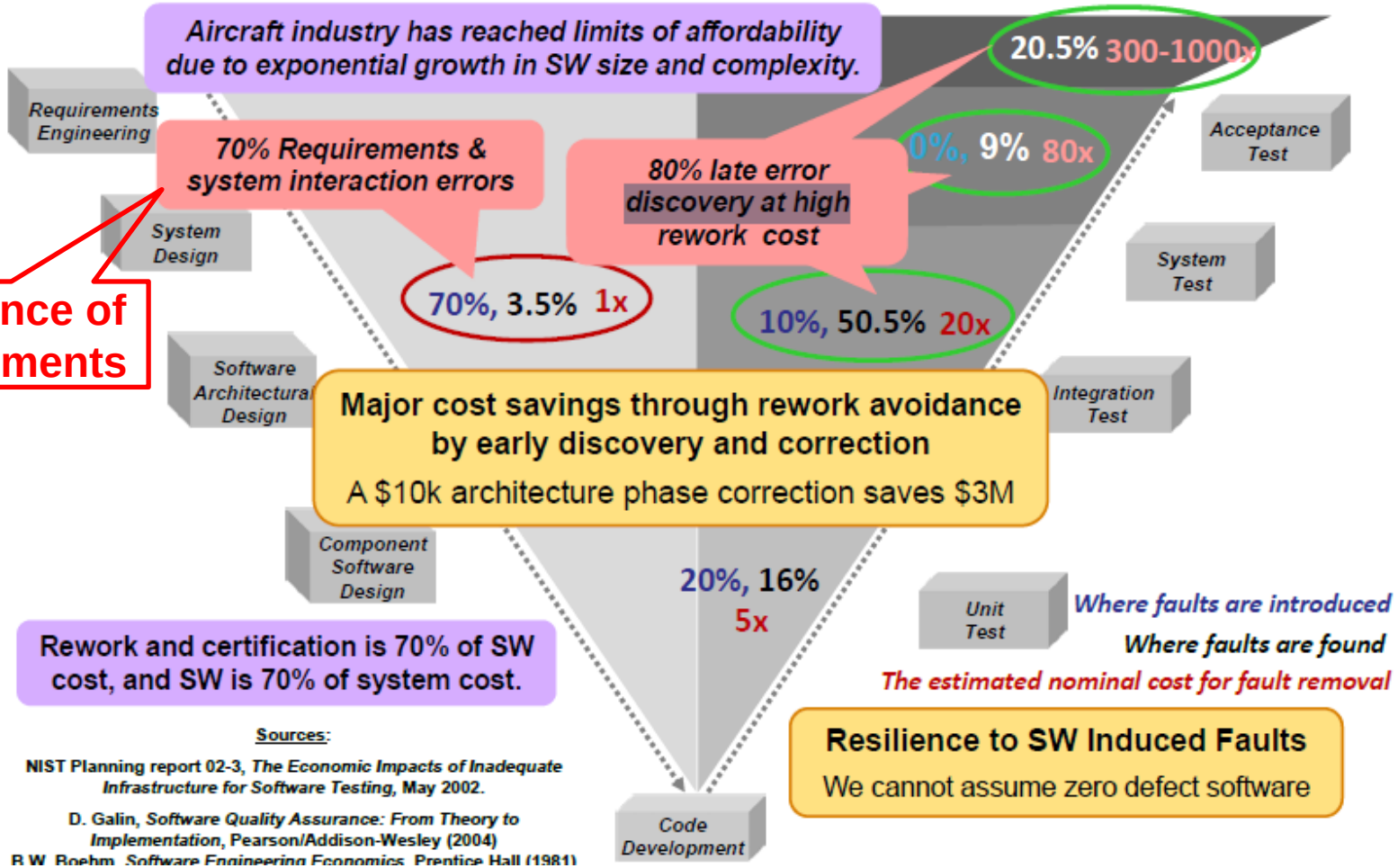
Non-Linear Development Effort Increase: F35 versus F16 Example



- **F35 SLOC / F16 SLOC ~ 175**
- **F35 Effort / F16 Effort ~ 300**
  - Source: SAVI Project (https://savi.avsi.aero/)

- **A400M:**
  - 4 years delayed
  - 6.2 billion euros over budget (30% overrun)
  - Source: https://www.rt.com/business/airbus-a400m-france-delays-561/

Why this Non-Linearity?
Where are Faults Introduced?



Importance of Requirements

From Feiler and Delange, *Design and Analysis of Cyber-Physical Systems: AADL and Avionics Systems*, 2013.

Source of Problems

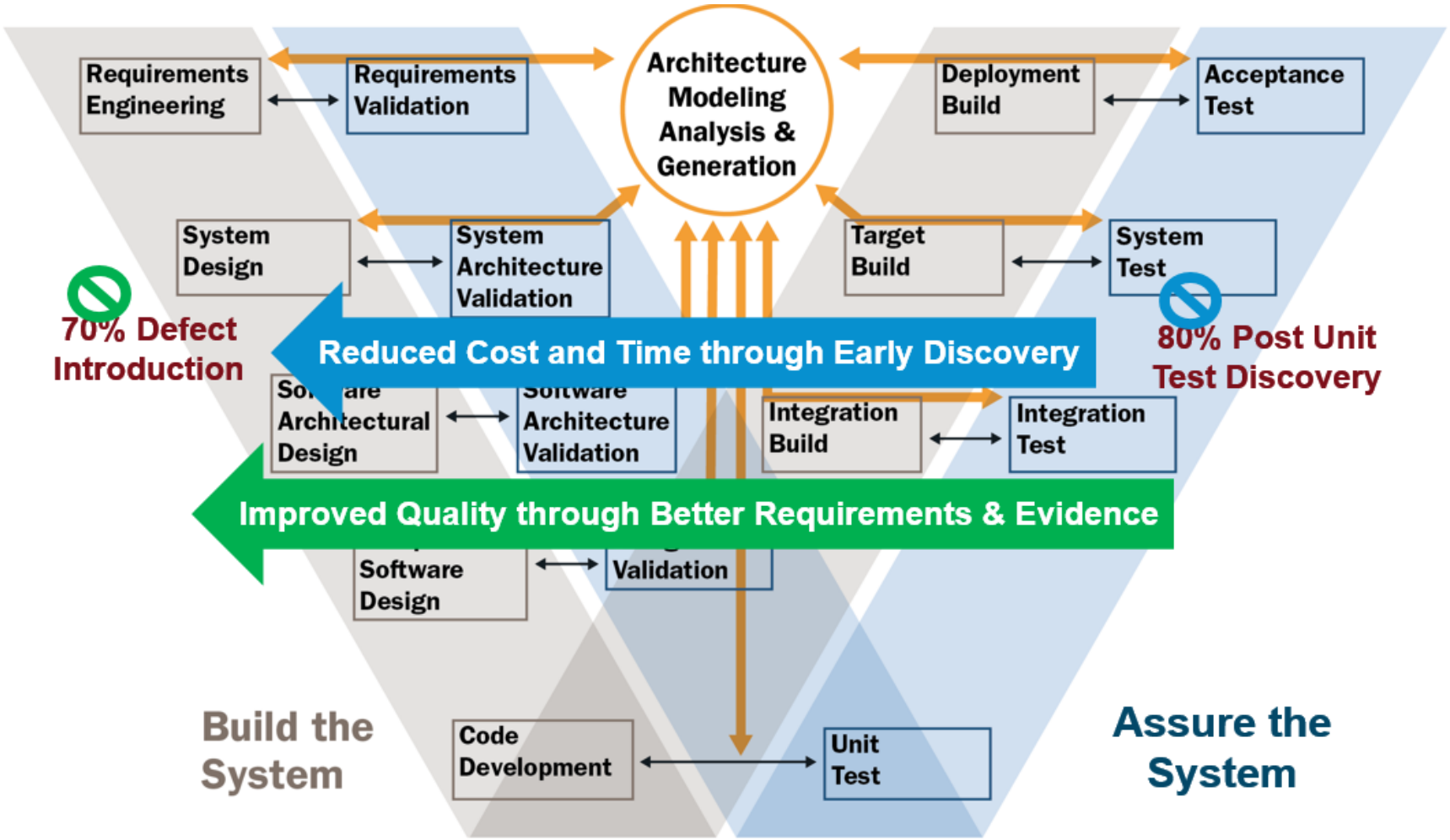
- **Growth of software: Software count increases while the physical parts count remains stable,**
- **Software increase from 66% in 2010 to 88% of the total system-development cost by 2024**
- System integration
- **Ambiguous, missing, incomplete, and inconsistent requirements**
  - Requirements-related rework cost: 79%!
  - Design-related rework cost 16%

- **Source: ROI (Return on Investment) Analysis of the System Architecture Virtual Integration Initiative**
  - SAVI Project (https://savi.avsi.aero/)
  - SEI technical report available at https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=517157

Outline

- Introduction
- Model-Based Engineering with AADL
- Overview of Line Follower Robot Example
- Modeling the Line Follower Robot Example
- Conclusion and Perspectives

Model-Based Engineering as a Solution



From McGregor, Gluch, and Feiler, *Analysis and Design of Safety-critical, Cyber-physical Systems*, 2017.

AADL
(Architecture Analysis & Design Language)

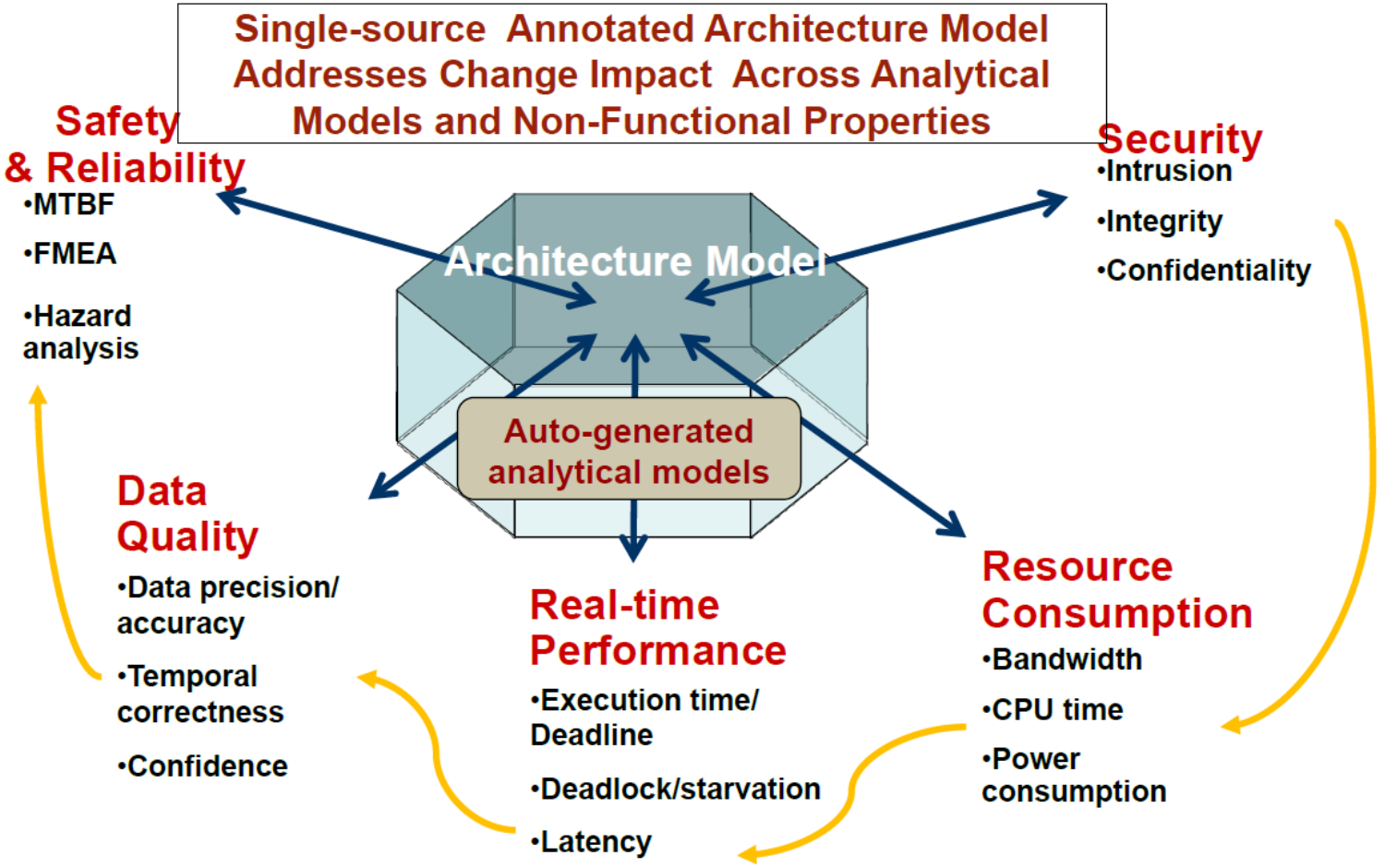
- **Modeling Language for Safety-Critical Systems**
- **Analysis of properties such as:**
  - Timing
  - Safety
  - Schedulability
  - Fault tolerance
  - Security
  - Functional simulation
  - Etc.
- **SAE standard AS-5506**
- **Textual and graphical notations**

AADL for Cyber-Physical Systems

- **Abstract components**
  - System-level modeling
- **Software components**
  - Cyber part
- **Hardware components**
  - Physical part
- **Deployment**
  - Complete CPS

- **Extensible via annexes:**
  - Behavioral (developed at Telecom ParisTech)
  - Error Model (fault tolerance)
  - Code generation
  - ARINC 653
  - Network annex (under development)
  - **Assurance annex** (ALISA)

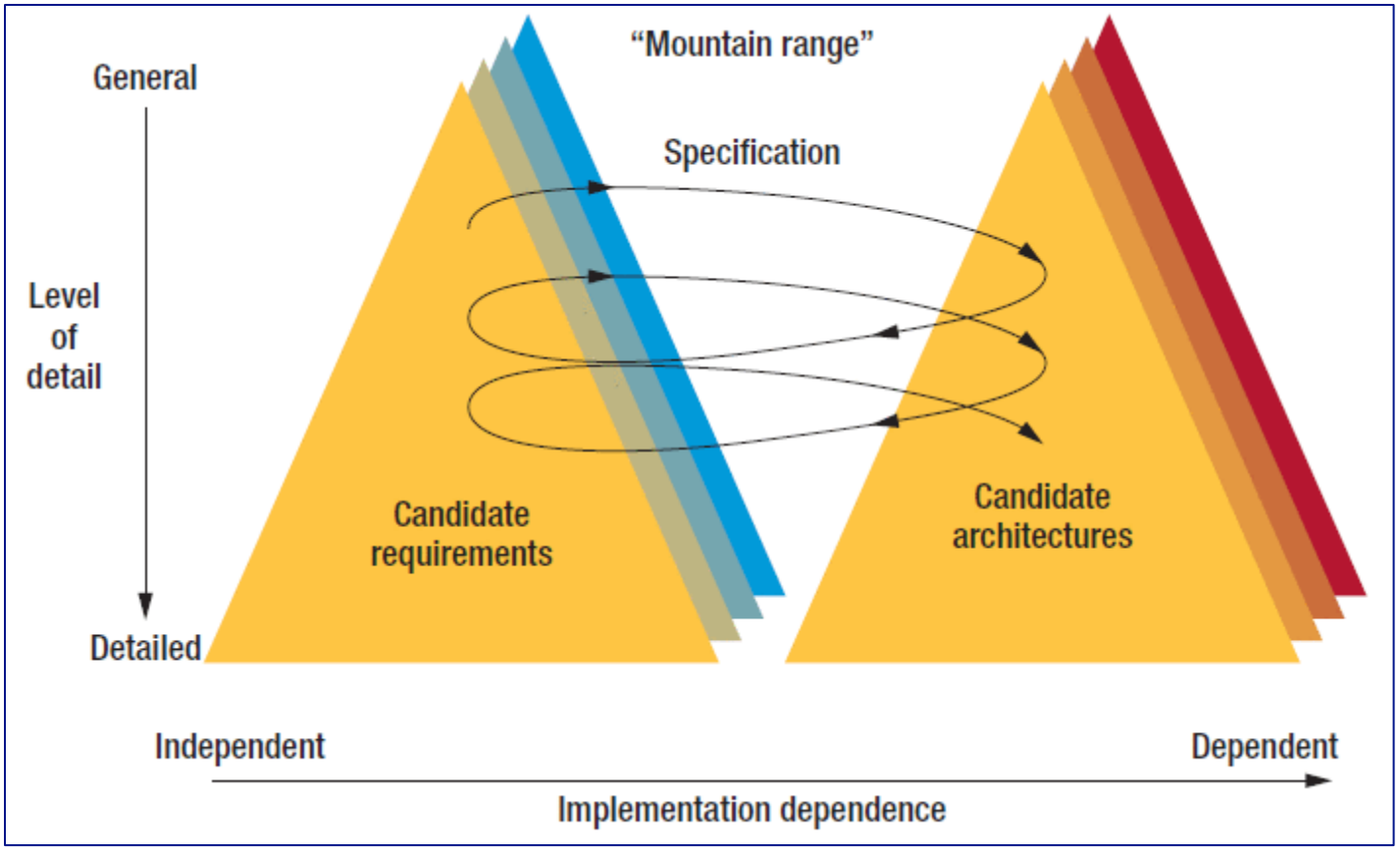
ACVIP (Architecture-Centric Virtual Integration Process)



From Feiler, Hansson, de Niz and Wrage. *System Architecture Virtual Integration: An Industrial Case Study*, 2009.

ALISA (Architecture-Led Incremental System Assurance)

- **Requirements and architecture "Twin Peaks" model**



From Cleland-Huang et al., *The Twin Peaks of Requirements and Architecture*, IEEE Software, 2013

ALISA in a Nutshell (Architecture-Led Incremental System Assurance)

- **Initiated from RDAL (Requirements Definition and Analysis Language)**
  - **Fragment language** that can be coupled with an architecture language
  - Inspired from FAA Requirements Engineering Management Handbook and other RE approaches (SysML, KAOS, i*)

- **Re-implemented and extended as the ALISA set of textual notations**
  - **ReqSpec**: Stakeholder goals and system requirements.
  - **Verify**: verification methods, activities and verification plans with claims that requirements are satisfied by the results of verification activities
  - **Alisa**: Assurance cases (consist of one or more assurance plan)
  - **Assure**: Assurance case result instance, i.e., the evidence as the result of executing verification plans on one or more system instance models.
  - **Organization**: Defines the stakeholders of a project

- **Provides**
  - Stakeholder goals transformed into verifiable requirements
  - Executable verification activities
  - Assurance case modeling
  - Optimization goal (conflicts)
  - Etc.

Outline

- Introduction
- Model-Based Engineering with AADL
- Overview of Line Follower Robot Example
- Modeling the Line Follower Robot Example
- Conclusion and Perspectives

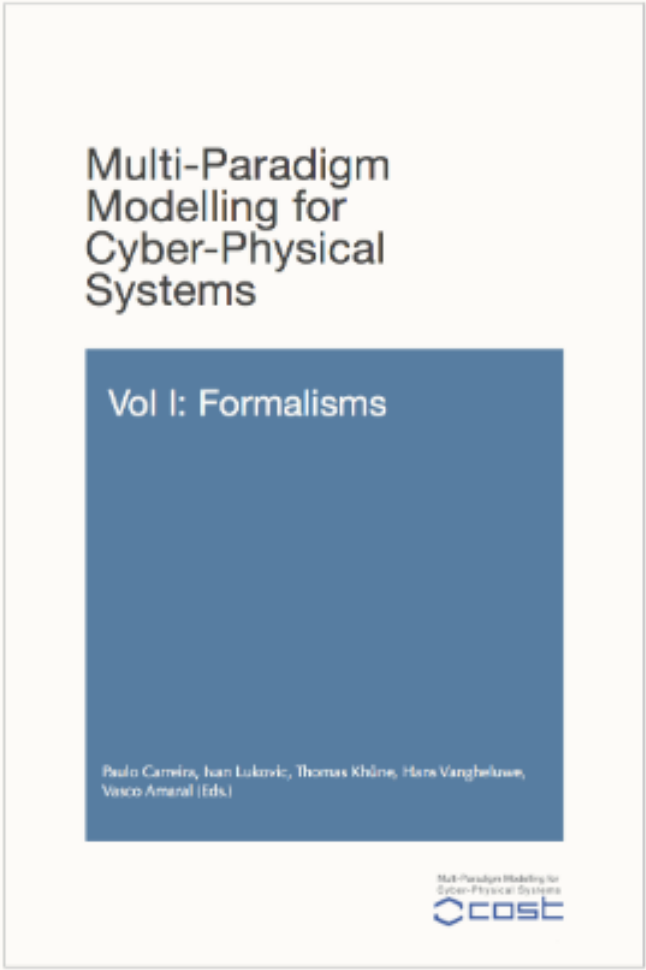
Example developed during MPM4CPS COST Action IC1404

- **Multi-Paradigm Modeling for Cyber-Physical Systems**
- **COST: European Cooperation in Science and Technology**
- **Provides funding for the creation of research networks, called “COST Actions”**

More info on this Example

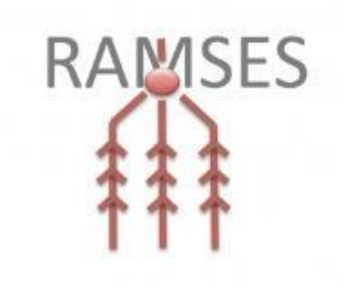
- **Book on formalisms for CPS**
  - Chapter on AADL for architecture illustrated with example

- **Training school Pisa November 18-21**
  - http://mpm4cps.eu/WGs/WG1/foundations/trainingSchools/pisa2018
  - 3 hours on architecture with AADL
    - Analysis and code generation with RAMSES
  - https://mem4csd.telecom-paristech.fr/blog/index.php/mpm4cps-training-school/



Line Follower Robot Overview

- **NXT Mindstorm Lego Robot**
- **Well documented**
  - Hardware developer kit
  - NXT OSEK (http://lejos-osek.sourceforge.net/)
  - Example line follower application on the web
- **Automatic C code generation from AADL models with RAMSES**
  - NXT OSEK middleware

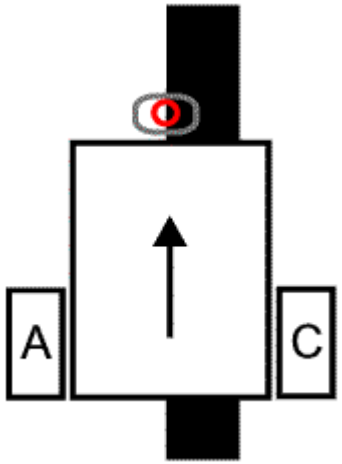


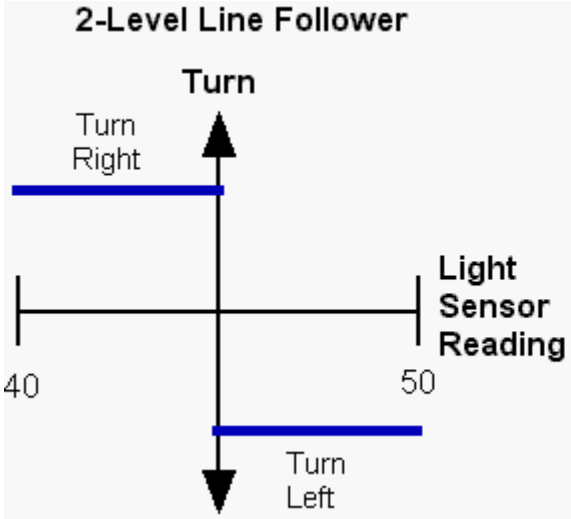
Functional Overview

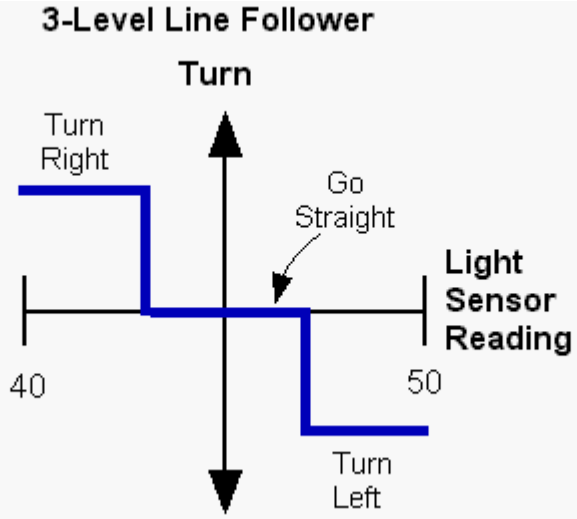
- **Goal: Carry objects in a warehouse**
  - Pick-up object
  - Follow a line on the floor
  - Drop-off object
- **Perform obstacles detection (e.g. other robots on crossing lines)**
- **Log events periodically to another computer (Raspberry Pi 3)**

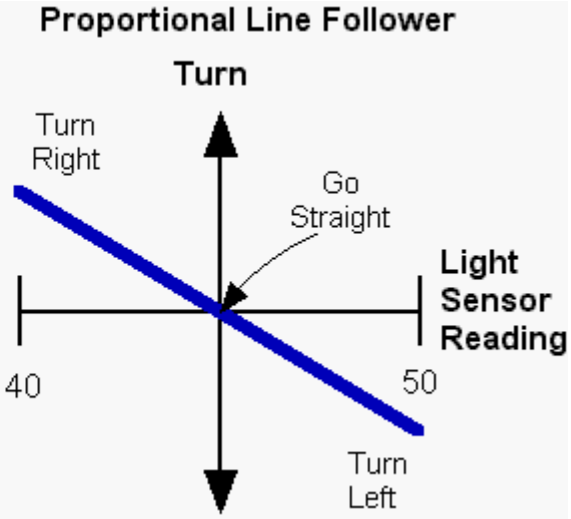
Line Following

- **PID controller**
  - http://www.nxtprograms.com/line_follower/steps.html
  - http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html





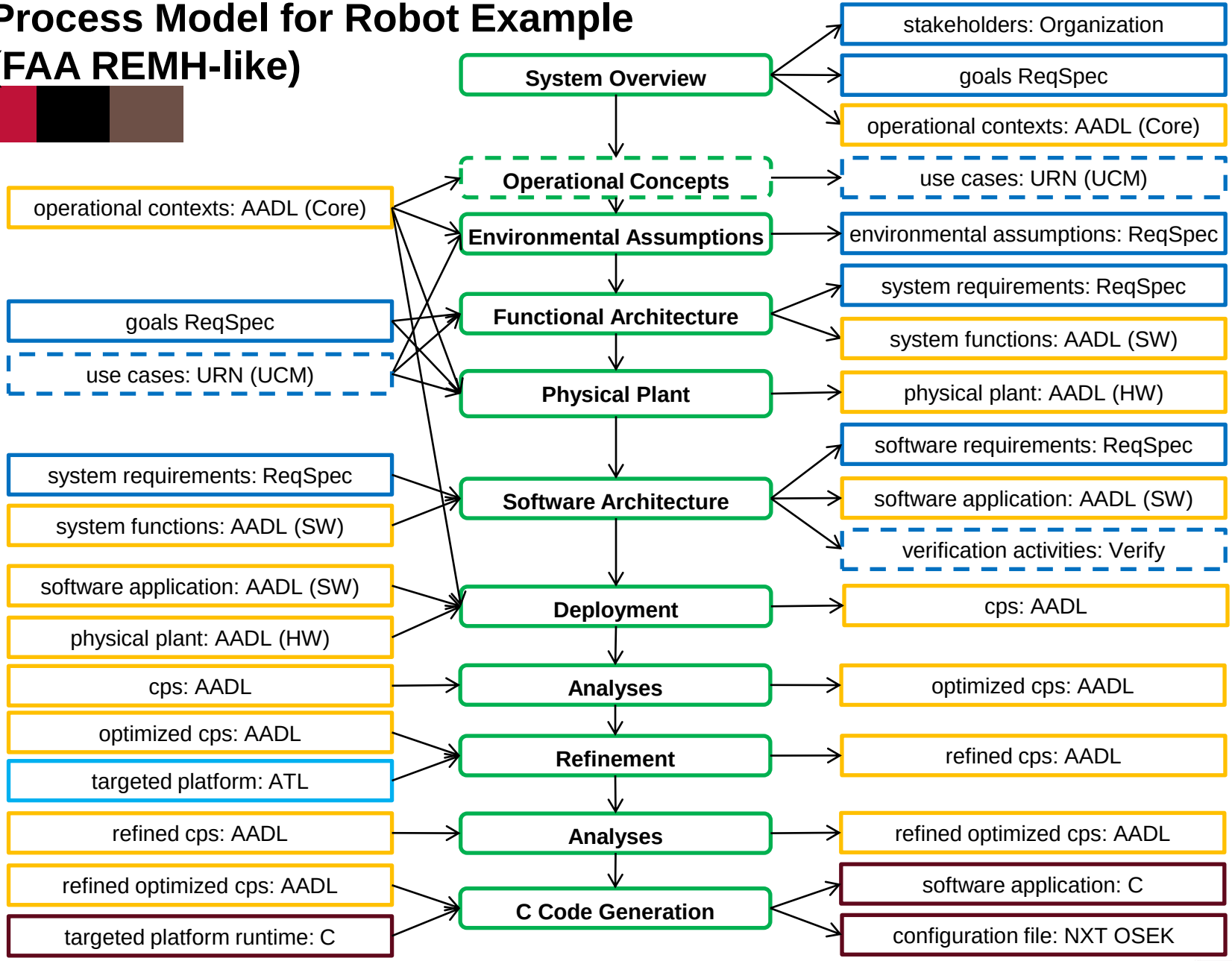




Outline

- Introduction
- Model-Based Engineering with AADL
- Overview of Line Follower Robot Example
- Modeling the Line Follower Robot Example
- Conclusion and Perspectives

Process Model for Robot Example (FAA REMH-like)



System Overview: Stakeholder Goals

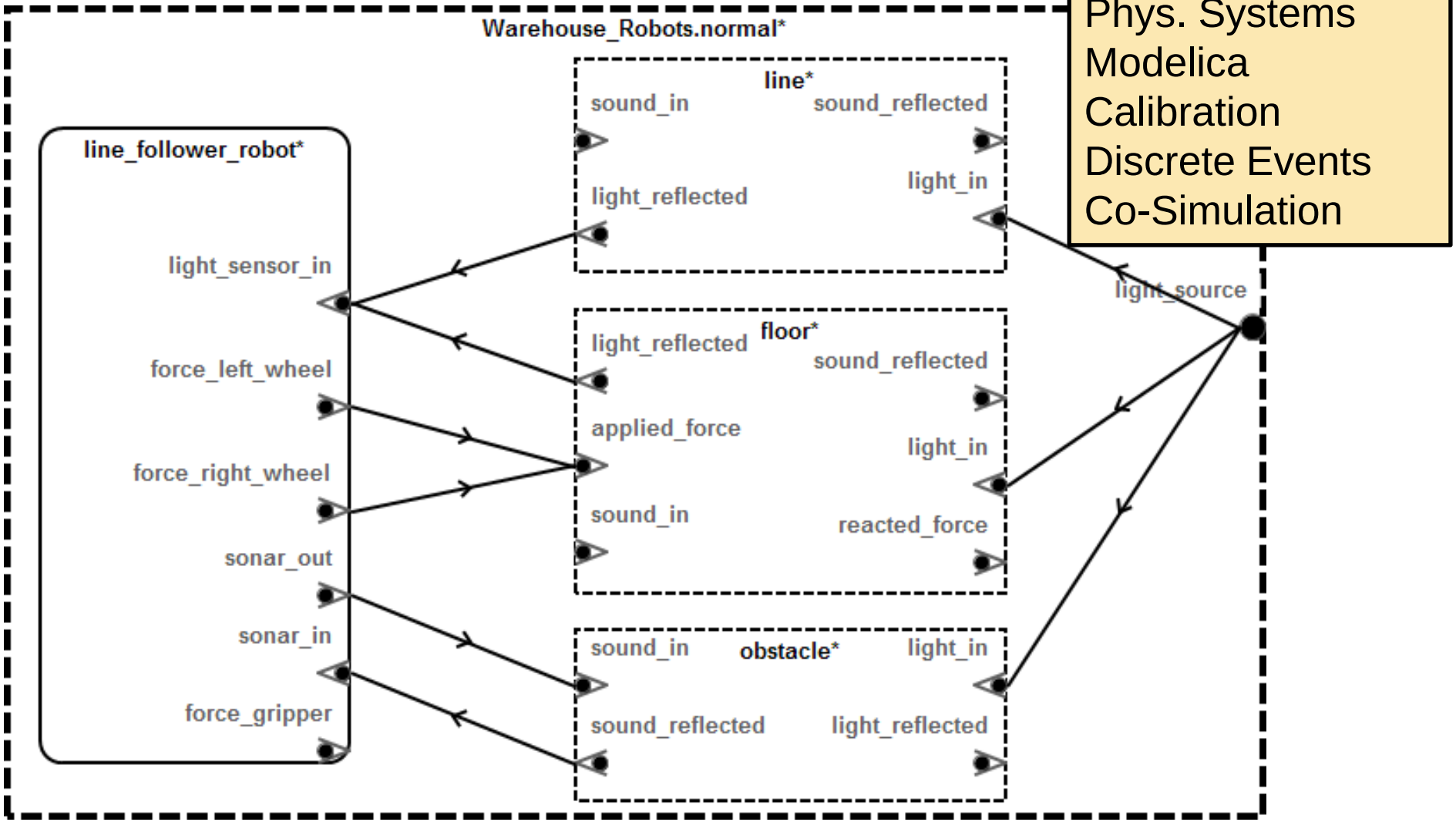
Architecture

```
stakeholder goals Line_Follower_Robot_Behavior for Line_Follower_App::Cary_Object [

    goal G_Behav_1 : "Objects_Transportation" [
        description
                "The robot should be able to carry an object between two specified points by following a predefined trajectory in the wa
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer
        rationale "This fulfills the main need of customers."
        category Quality.Behavior
    ]

    goal G_Behav_2 : "Obstacle_Avoidance" [
        description "The robot should be able to avoid obstacles along the path."
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer
        rationale
                "There may be several robots working on the warehouses and therefore, it is important to avoid damaging the robots and t
        category Quality.Behavior
    ]
]
```

```
stakeholder goals Line_Follower_Robot_Perf for Line_Follower_Robot_Cps::Line_Follower_Robot_Cps [

    goal G_Perf_1 : "Minimal Cost" [
        description "The cost of producing the robot should be minimal."
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer Tartempion_Warehouse_Equipments_Ltd.Marketing
        rationale "The robot should be cheap so that it is competitive on the market."
        category Quality.Cost
    ]

    goal G_Perf_2 : "Minimal_Transportation_Time" [
        description "The time taken to carry objects should be minimal."
        stakeholder Tartempion_Warehouse_Equipments_Ltd.Customer Tartempion_Warehouse_Equipments_Ltd.Customer
        rationale "The robot should be fast to meet the needs of customers."
        category Quality.Performance
    ]
]
```

System and Environment: Normal Operational Context

Architecture
Differential Equations
Phys. Systems Modelica
Calibration
Discrete Events
Co-Simulation



Environmental Assumptions

```
system requirements Line_Follower_Robot_Env_Assumptions for Line_Follower_Robot::Warehouse_Robots.normal [

    requirement EA_1: "Minimum Warehouse Luminosity" for light_source [
        description "The power of the light source shall not be less than the Minimum Illuminance value"
        rationale "Otherwise the light sensor of the robot will not be able to give proper readings given its sensitivity and its calibra
        category Kind.Assumption
        val Minimum_Illuminance = 100.0 lx
        value predicate #Physics_Properties::Illuminance >= Minimum_Illuminance
    ]

    requirement EA_2: "Minimum Curvature Radius" for line [
        description "The curvature radius of the line to be followed by the robot shall not be lower than TODO"
        rationale "Otherwise the robot given its speed, mass and response time will not be able to follow the line."
        category Kind.Assumption
        val Minimum_Curvature_Radius = 100.0 mm
        value predicate #Physics_Properties::Curvature_Radius >= Minimum_Curvature_Radius
    ]
]
```

Functional Architecture: High Level Requirements


```
requirement R_Behav_1 : "Carry_Object_Function" [
    description
            "The robot shall carry an object between two specified points by following a predefined trajectory in the warehouse."
    see goal Line_Follower_Robot_Behavior.G_Behav_1
    category Quality.Behavior
]

    requirement R_Behav_1_1 : "Pick_Up_Object_Function" for pick_up_object [
        description "At the beginning of the path, the robot shall pick up an object on the floor."
        category Quality.Behavior
        decomposes R_Behav_1
    ]

    requirement R_Behav_1_2 : "Follow_Line_Function" for follow_line [
        description "The robot shall follow a line on the floor of the warehouse."
        category Quality.Behavior
        decomposes R_Behav_1
    ]

    requirement R_Behav_1_3 : "Drop_Off_Object_Function" for drop_off_object [
        description "At the end of the path, the robot shall drop off the carried object on the floor."
        category Quality.Behavior
        decomposes R_Behav_1
    ]
```

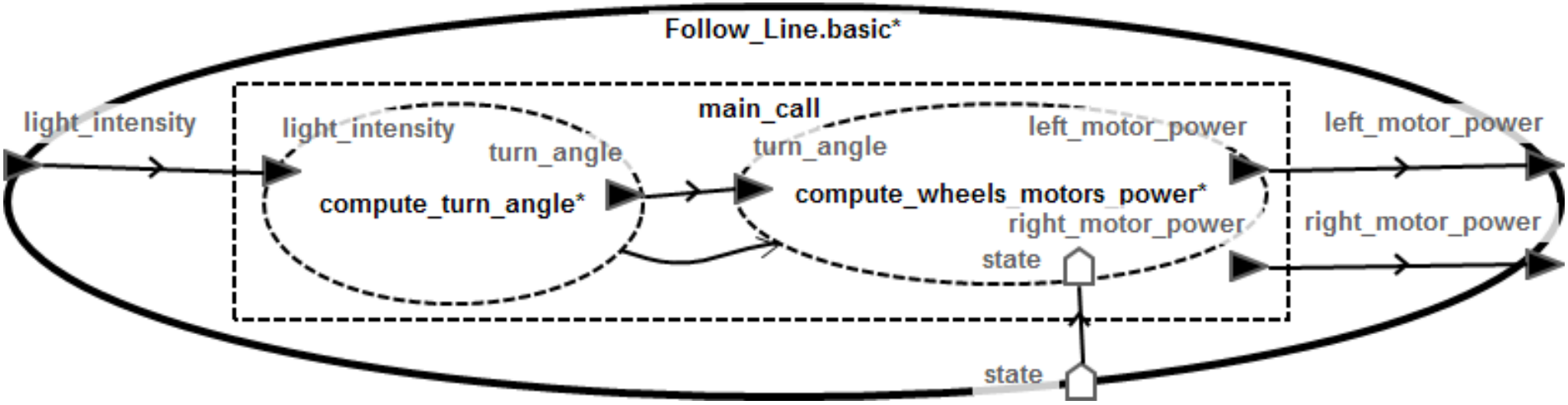
Functional Architecture: Line Following Requirements

Model Checking

```
system requirements Follower_Line_Behavior for Line_Follower_App::Follow_Line.basic [

    requirement R_Behav_1_2 : "Follow_Line_Function" [
        description "The robot shall follow a line on the floor of the warehouse."
        category Quality.Behavior
        decomposes Line_Follower_Robot_Behavior.R_Behav_1
    ]

        requirement R_Behav_1_2_1 : "Set_Turn_Angle_Function" for compute_turn_angle [
            description "The controller shall set the value of the turn angle variable."
            category Quality.Behavior
            decomposes R_Behav_1_2
        ]

        requirement R_Behav_1_2_2 : "Set_Left_Wheel_Power_Function" for compute_wheels_motors_power [
            description "The left wheel controller shall set the value of the left wheel power variable."
            category Quality.Behavior
            decomposes R_Behav_1_2
        ]

        requirement R_Behav_1_2_3 : "Set_Right_Wheel_Power_Function" for compute_wheels_motors_power [
            description "The right wheel controller shall set value of the right wheel power variable."
            category Quality.Behavior
            decomposes R_Behav_1_2
        ]
]
```

Functional Architecture: Line Following Function

Architecture



Software Architecture: Follow Line Subprogram

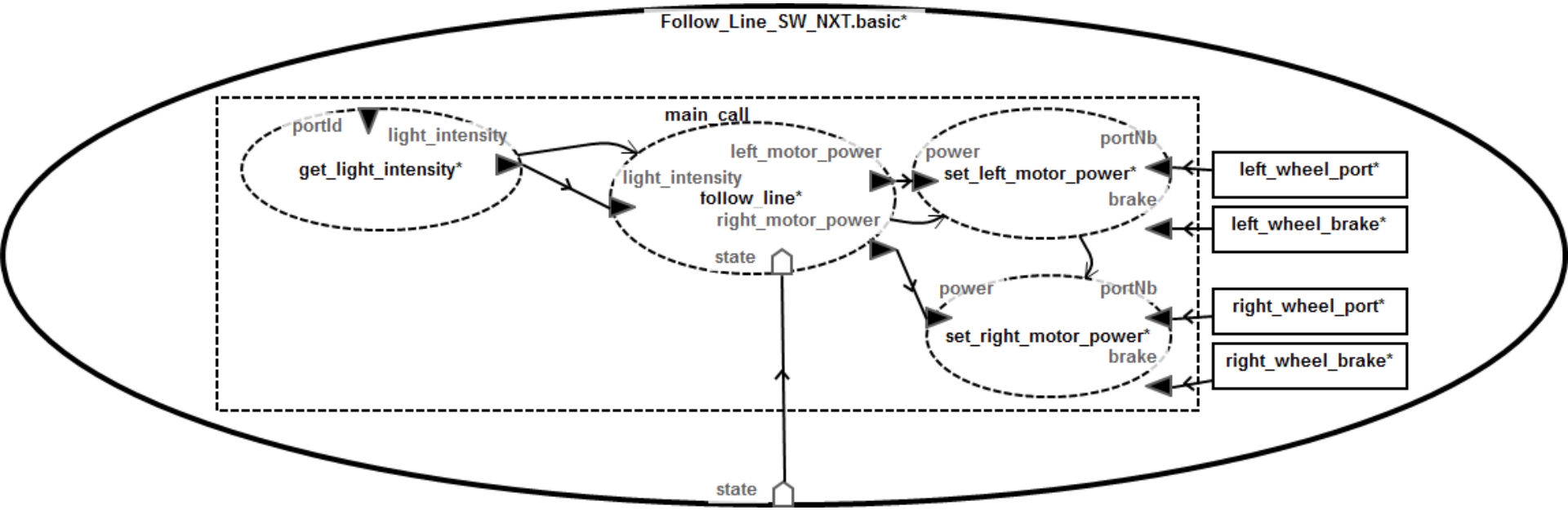

```
subprogram Follow_Line_SW extends Line_Follower_Functions::Follow_Line
    features
        light_intensity: refined to in parameter Light_Intensity_SW;
        left_motor_power: refined to out parameter Power_SW;
        right_motor_power: refined to out parameter Power_SW;
        state: refined to requires data access Robot_State_SW;
    properties
        Classifier_Substitution_Rule => Type_Extension;
end Follow_Line_SW;

subprogram implementation Follow_Line_SW.basic extends Line_Follower_Functions::Follow_Line.basic
    subcomponents
        compute_turn_angle: refined to subprogram Compute_Turn_Angle_SW.pid;
        compute_wheels_motors_power: refined to subprogram Compute_Wheels_Motors_Power_SW.basic;
end Follow_Line_SW.basic;
```

Software Architecture: Software Variables

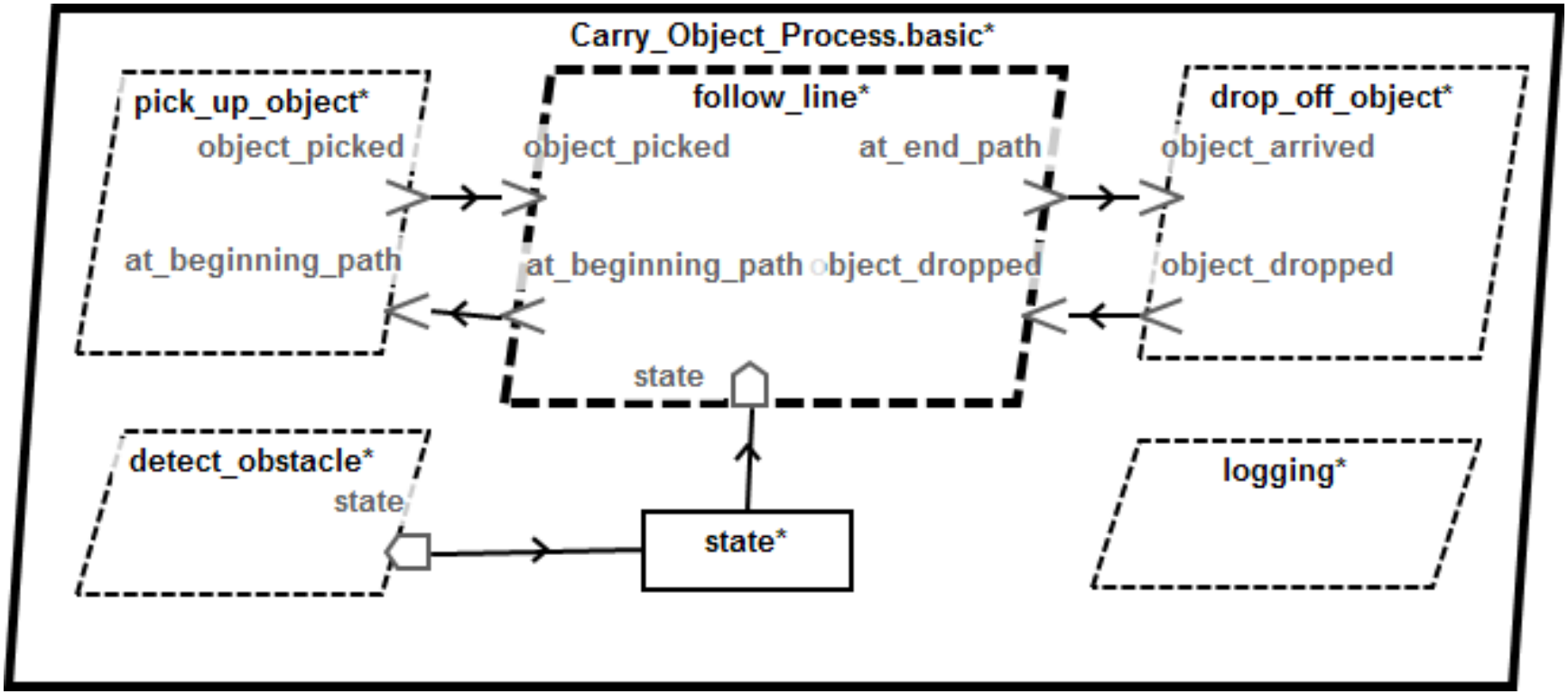

```
data Light_Intensity_SW extends Line_Follower_Functions::Light_Intensity
    properties
        Data_Model::Data_Representation => integer;
end Light_Intensity_SW;

data Turn_Angle_SW extends Line_Follower_Functions::Turn_Angle
    properties
        Data_Model::Data_Representation => integer;
end Turn_Angle_SW;

data Power_SW extends Line_Follower_Functions::Power
    properties
        Data_Model::Data_Representation => integer;
end Power_SW;

data Robot_State_SW extends Line_Follower_Functions::Robot_State
    properties
        Data_Model::Data_Representation => Enum;
        Data_Model::Enumerators => ( "FORWARD", "STOP" );
        Source_Name => "Robot_state";
        Source_Text => ("data_types.h");
end Robot_State_SW;
```

Software Architecture: NXT Follow Line Subprogram

Architecture DSE

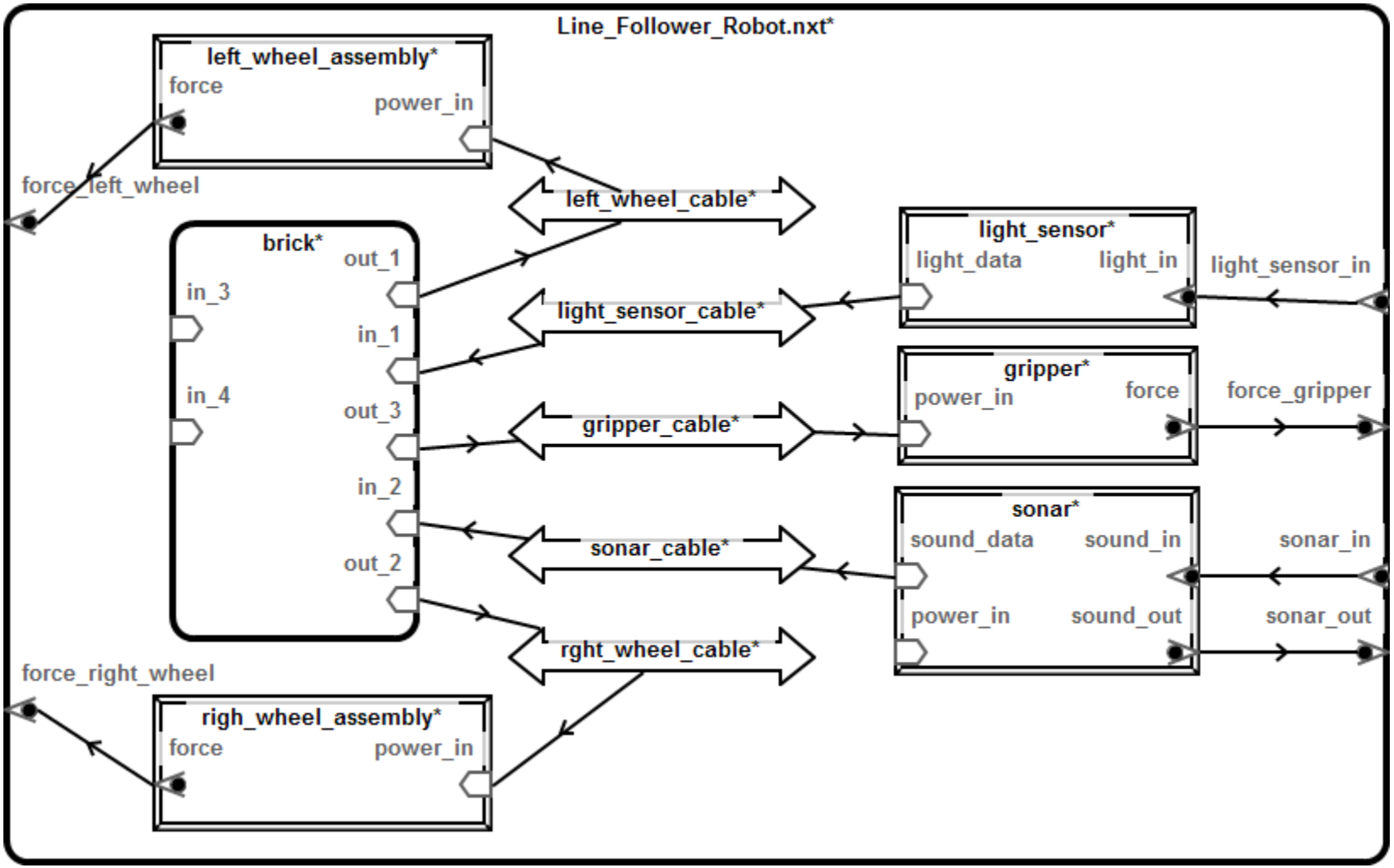


Software Architecture: Carry Objects Threads

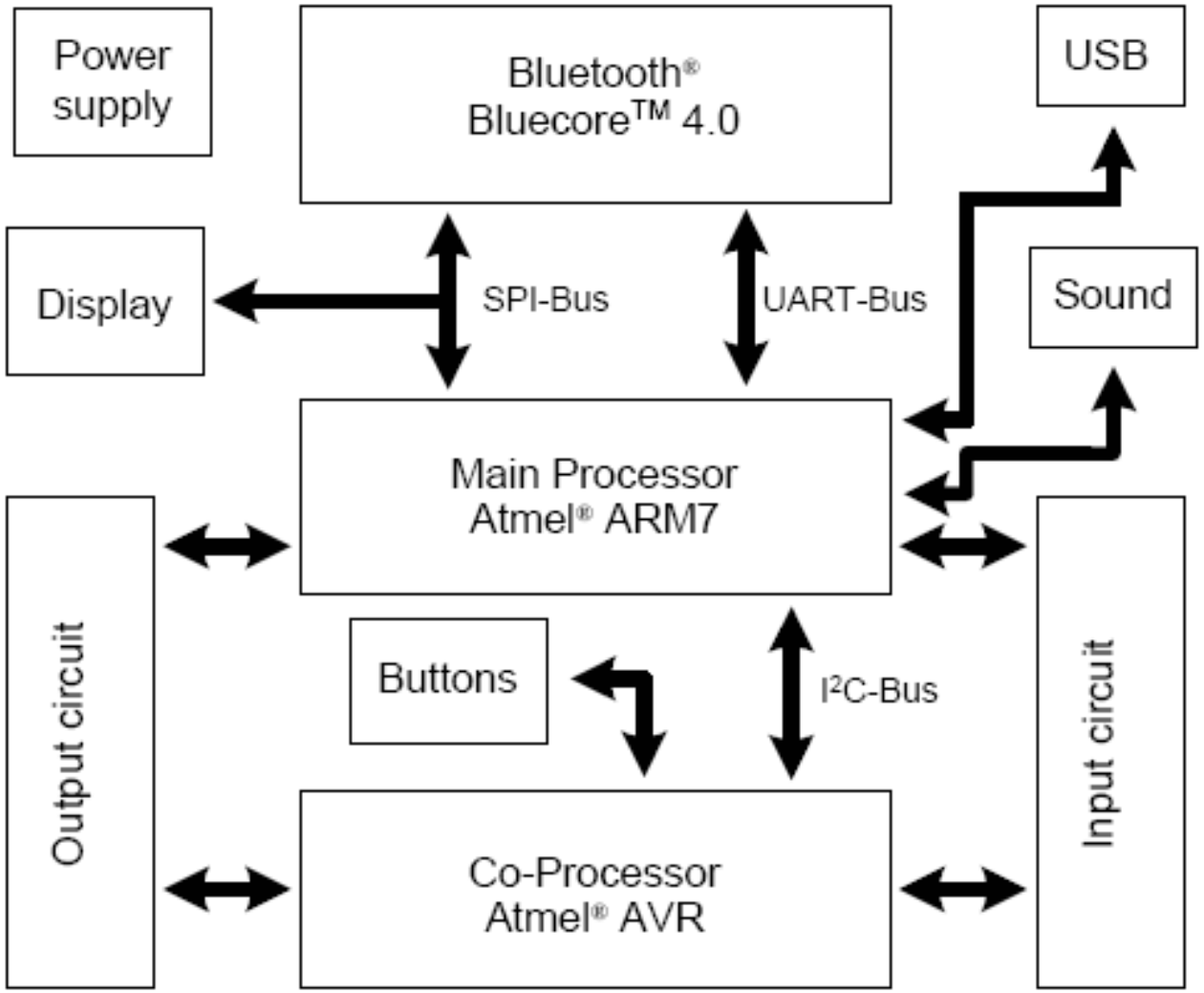


Physical Plant Model: Line Follower NXT Robot

Control
Model Checking

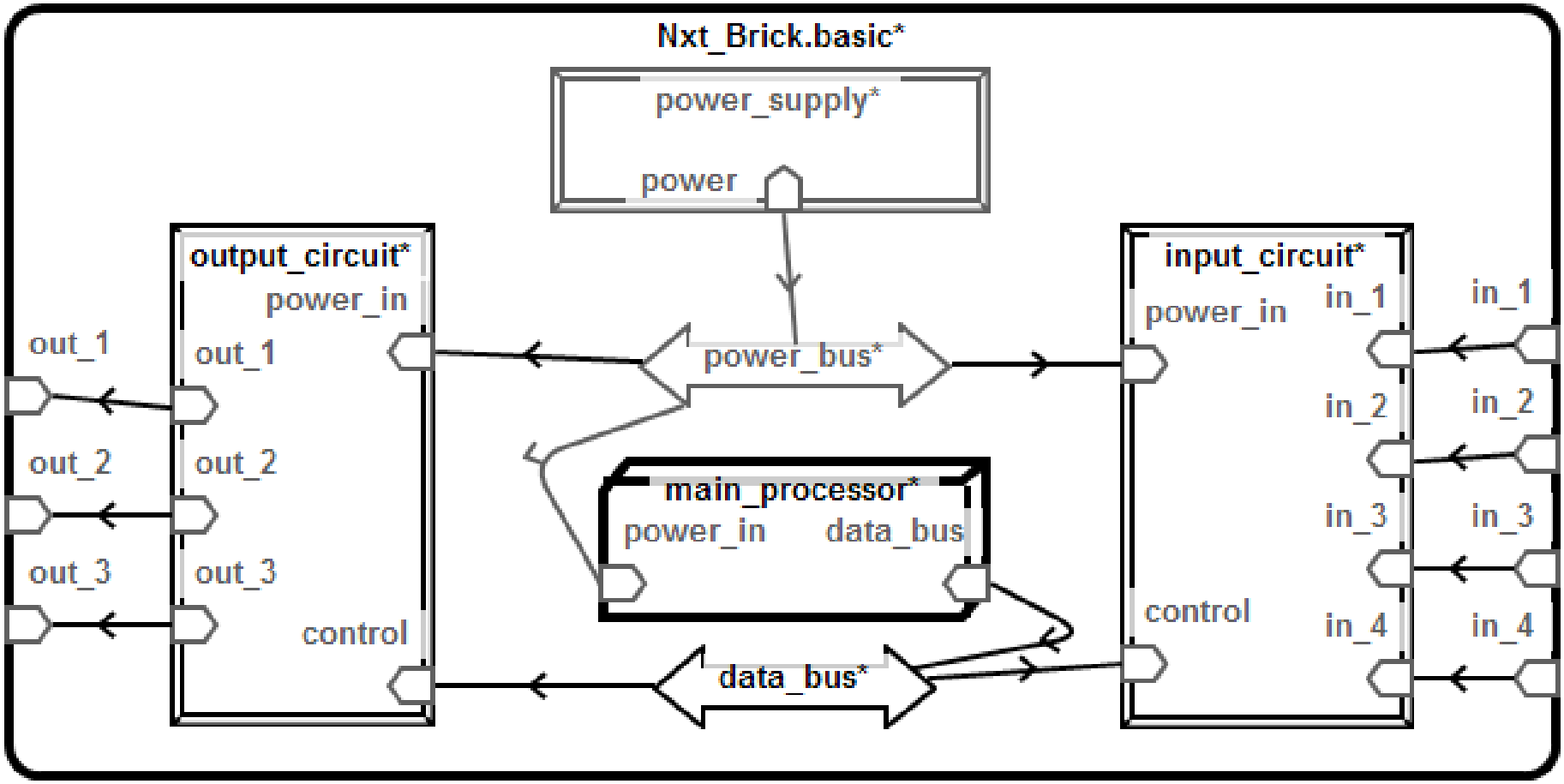


NXT Brick Hardware Kit Specification

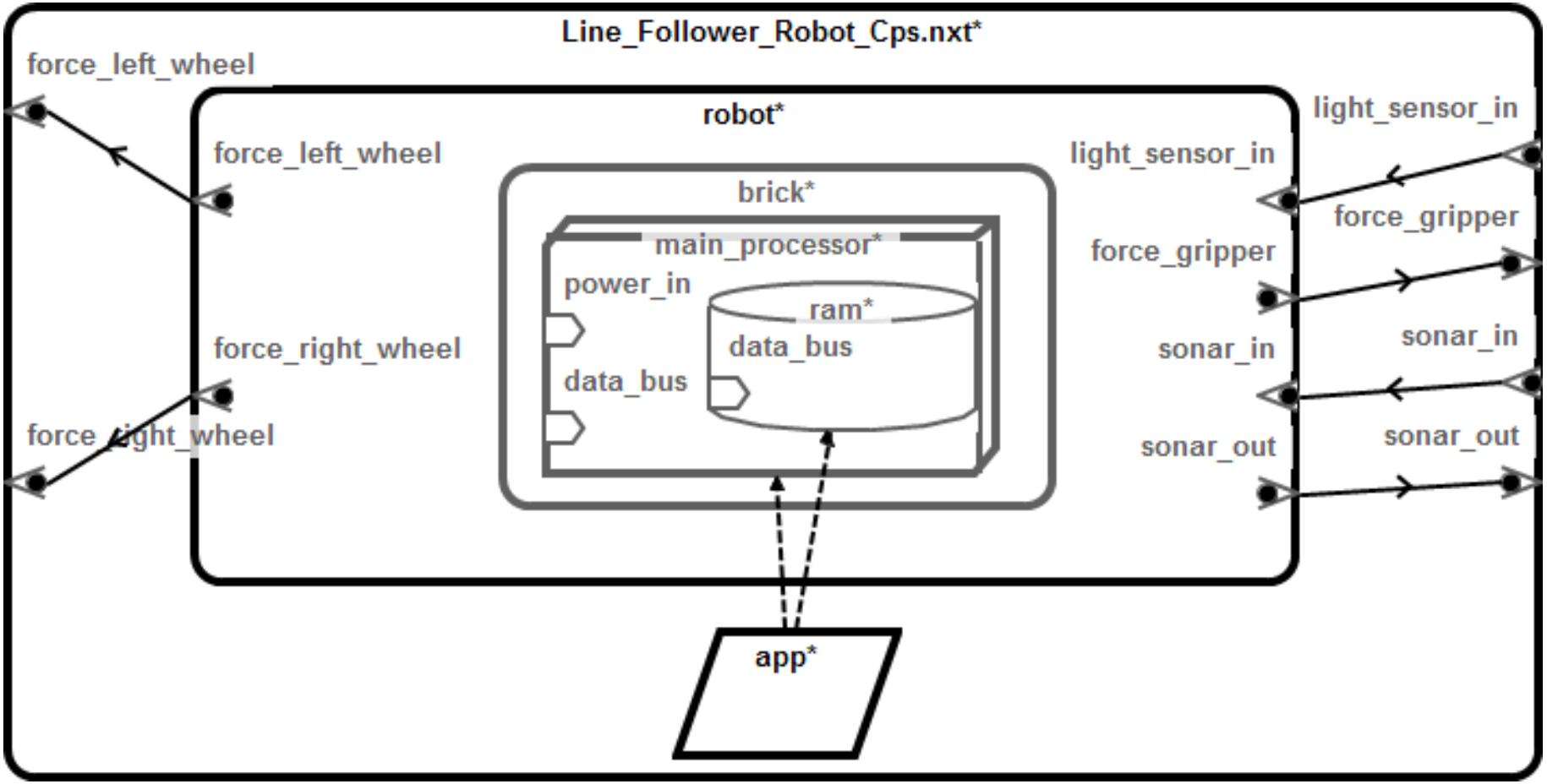


Execution Platform: NXT Brick

Architecture

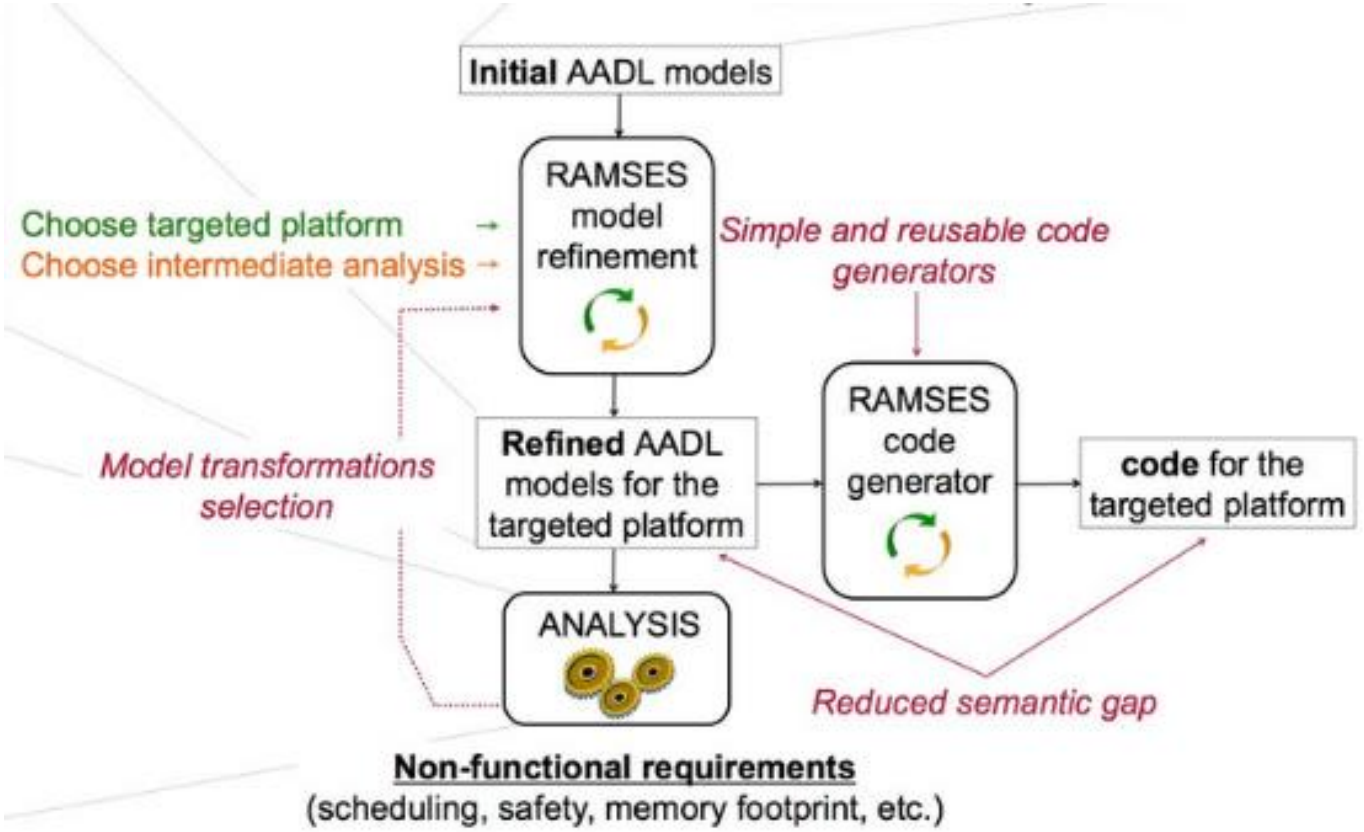


Deployment: Line Follower NXT Robot CPS

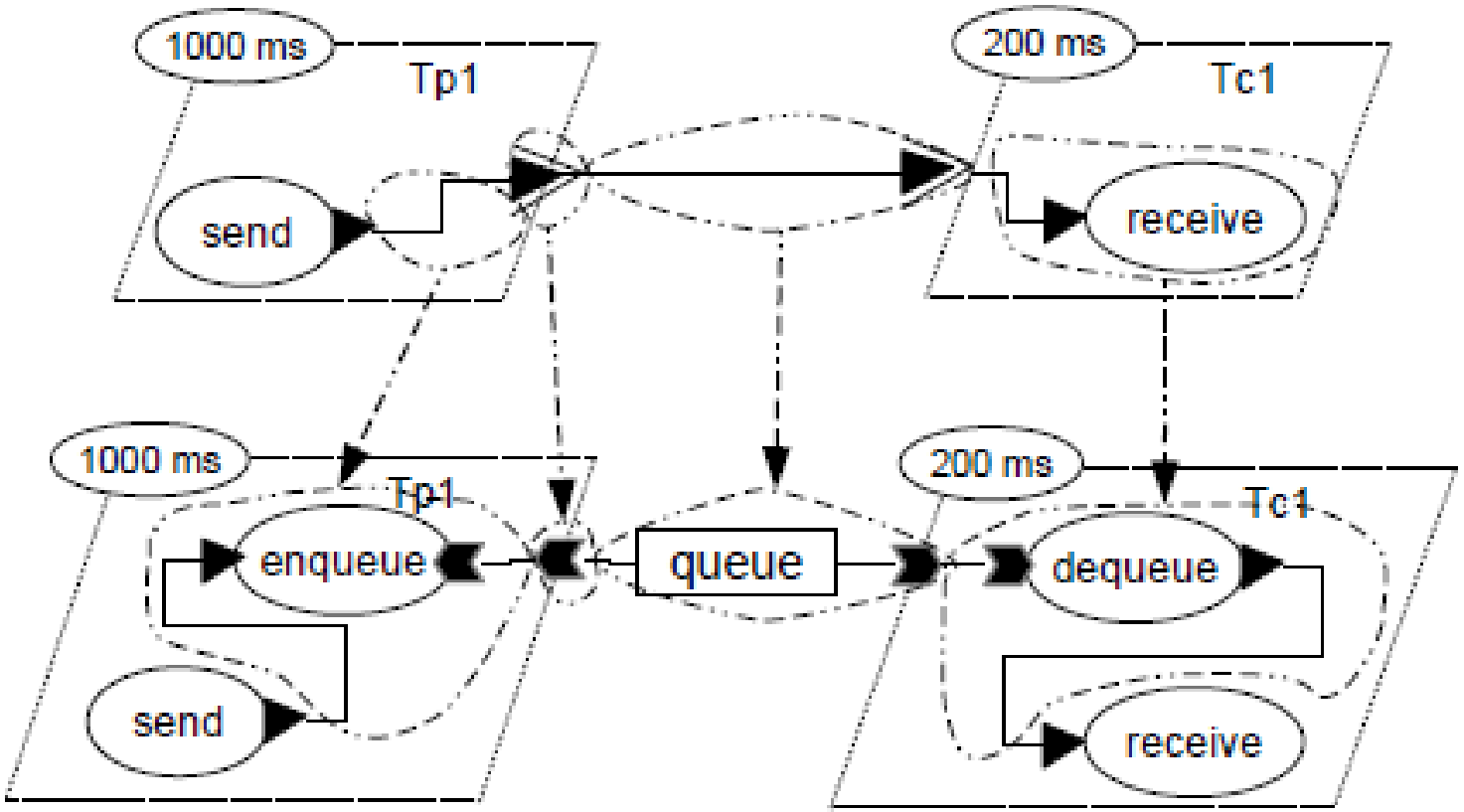


RAMSES Code Generation

- **Refinement of AADL Models for the Synthesis of Embedded Systems**
  - https://mem4csd.telecom-paristech.fr/blog/index.php/ramses/



RAMSES Refinement Rules Example: Local Communications



Supported Operating System Platforms

- **POSIX**
  - Linux
- **ARINC653:**
  - POK: https://pok-kernel.github.io/
  - VxWorks: https://www.windriver.com/products/vxworks/
  - Standard
- **OSEK**
  - nxtOSEK: http://lejos-osek.sourceforge.net/

Outline

- Introduction
- Model-Based Engineering with AADL
- Overview of Line Follower Robot Example
- Modeling the Line Follower Robot Example
- Conclusion and Perspectives

Conclusion and Perspectives

- **ALISA / AADL well suited for CPS**

- **Requirements:**
  - Stakeholder goals and system requirements.
  - **V**erification methods, activities and verification plans and claims
  - Assurance plans
  - Assurance case results

- **CPS:**
  - Cyber
  - Physical
  - CPS (deployment)

- **Development Process**
  - Requirements and architecture twin peaks iterative and incremental model
  - Architecture-centric virtual integration