AADL Standards Meeting
RAMSES Tutorial

Dominique Blouin and Etienne Borde
Telecom ParisTech
dominique.blouin@telecom-paristech.fr
etienne.borde@telecom-paristech.fr

Outline

- Context
- AADL for CPSs: Lego Robot Case Study
- RAMSES and Code Generation for Case Study
- Mixed Criticality Scheduling

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”**

Cyber-Physical Systems

- **Multi-physics systems (mechanical, electrical, hydraulic, biochemical, ...)**
- **Computational (control systems, signal processing, logical inferencing, planning, ...)**
- **Uncertain environments, with human actors, in a socio-economic context**

Multi-Paradigm Modeling

- **Prof Hans Vangueluwe**
  - Co-founder of MODELICA
  - Chair of MPM4CPS
- **Model every part and aspect of a system explicitly**
- **At the most appropriate level(s) of abstraction**
- **Using the most appropriate modeling formalism(s)**
- **Answer to the challenges of designing CPSs**

Objectives of MPM4CPS

- Enhance trans-disciplinary area of CPSs by unification through Multi-Paradigm Modeling

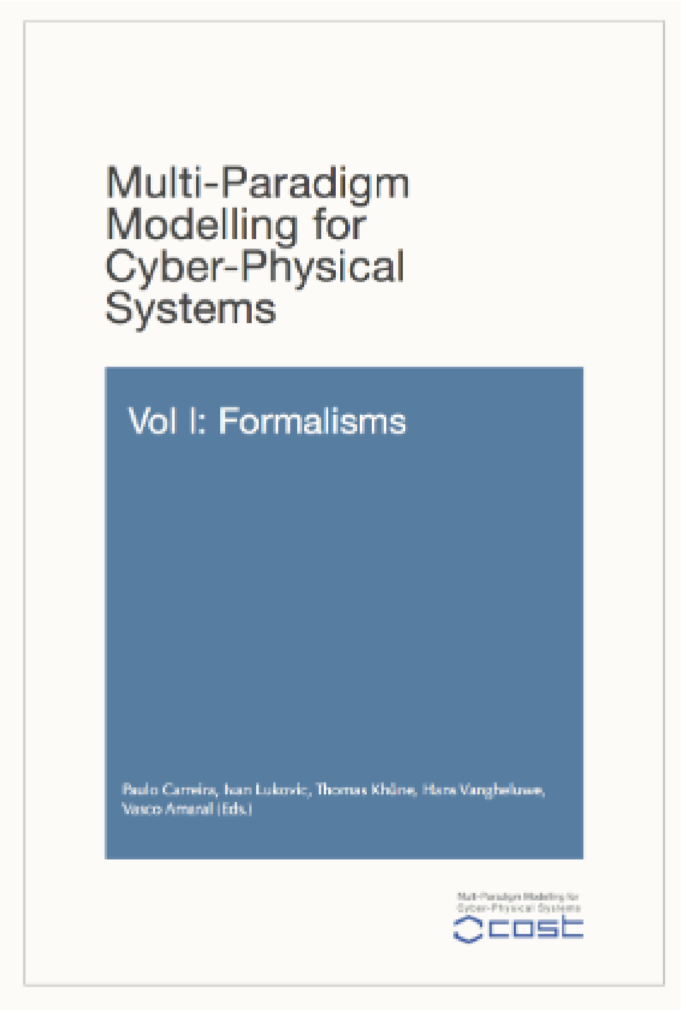
- Working groups:
  - WG1: Foundations
  - ~~WG2: Techniques~~
  - ~~WG3: Application domains~~
  - WG4: Education and dissemination

Outcomes

- **Book on formalisms for CPS**
  - Chapter on AADL for architecture

- **Training school November 18-21**
  - http://mpm4cps.eu/WGs/WG1/foundations/ningSchools/pisa2018
  - 3 hours on architecture with AADL
    - Analysis with OSATE and AADL Inspector
    - Code generation with RAMSES
  - **Lego Mindstorm line follower robot as case study for the whole training schoo**

- **Ontology of MPM4CPS (OWL)**



Multi-Paradigm Modelling for Cyber-Physical Systems

Vol I: Formalisms

Paulo Carreira, Ivan Lukovic, Thomas Kühne, Hans Vangheluwe, Vasco Amaral (Eds.)

Modeling the Architecture of Cyber-Physical Systems with AADL

- **Model both the cyber (software) and physical parts (hardware) and deployment**
- **Precise and rigorous semantics**
- **Allows several levels of abstractions down to deployment**
- **Components families**

Outline

- Context
- AADL for CPSs: Lego Robot Case Study
- RAMSES and Code Generation for Case Study
- Mixed Criticality Scheduling

Using of AADL for CPS Architecture Modeling

- **Is AADL appropriate for modeling CPS architectures?**

- **Originally targeting safety-critical real-time embedded systems**

Line Follower Robot Case Study: Functional Overview

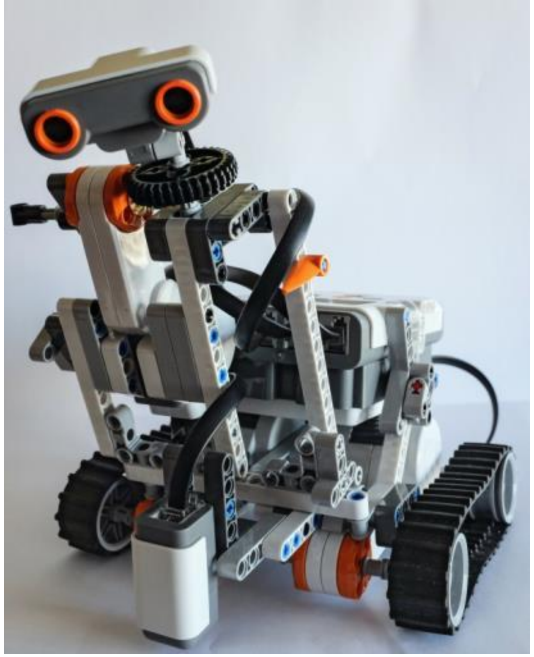
- **Line follower robot to carry objects in a warehouse**
  - Pick-up object
  - Follow a line on the floor
  - Drop-off object
- **Perform obstacles detection (e.g. other robots on crossing lines)**
- **Log events periodically**

Robot Overview

- **NXT Mindstorm Lego Robot**

- **Automatic C code generation from AADL models with RAMSES**
  - NXT OSEK middleware

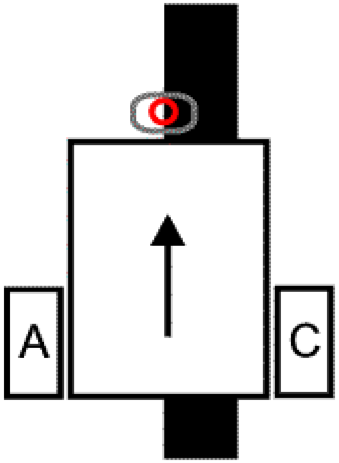
- **Well documented**
  - Hardware developer kit
  - NXT OSEK (http://lejos-osek.sourceforge.net/)
  - Example line follower application on the web

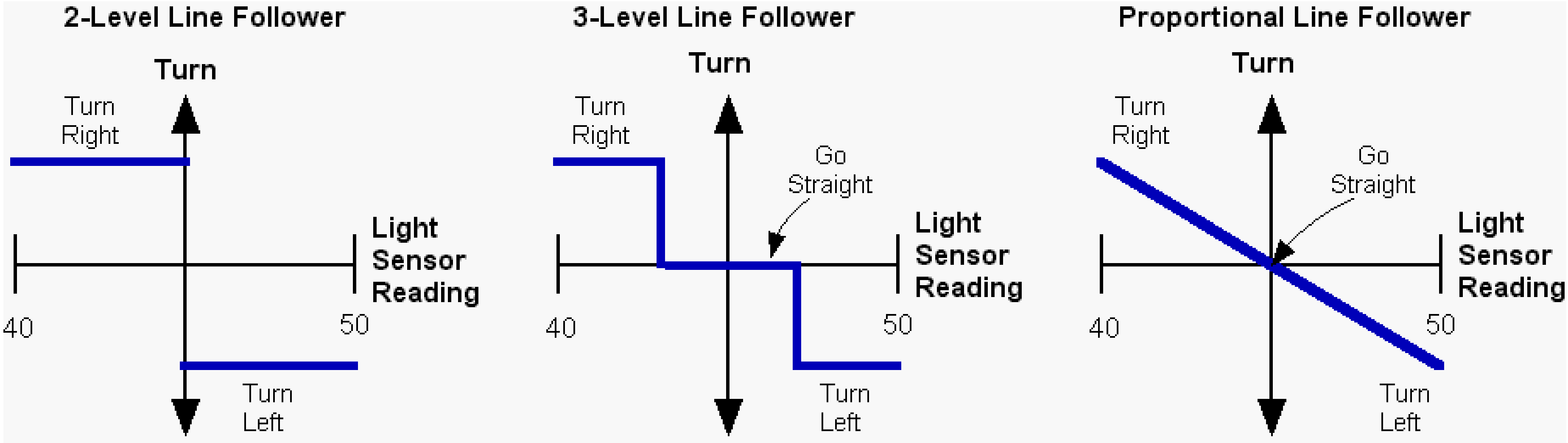


RAMSES

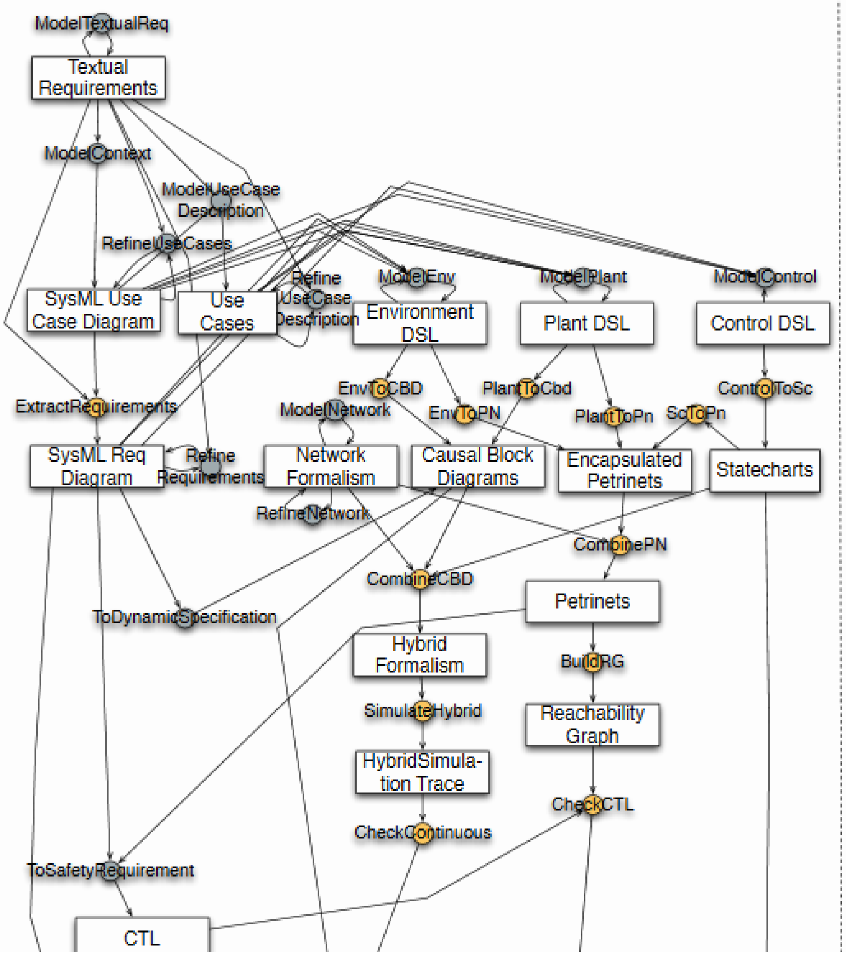
Robot Overview

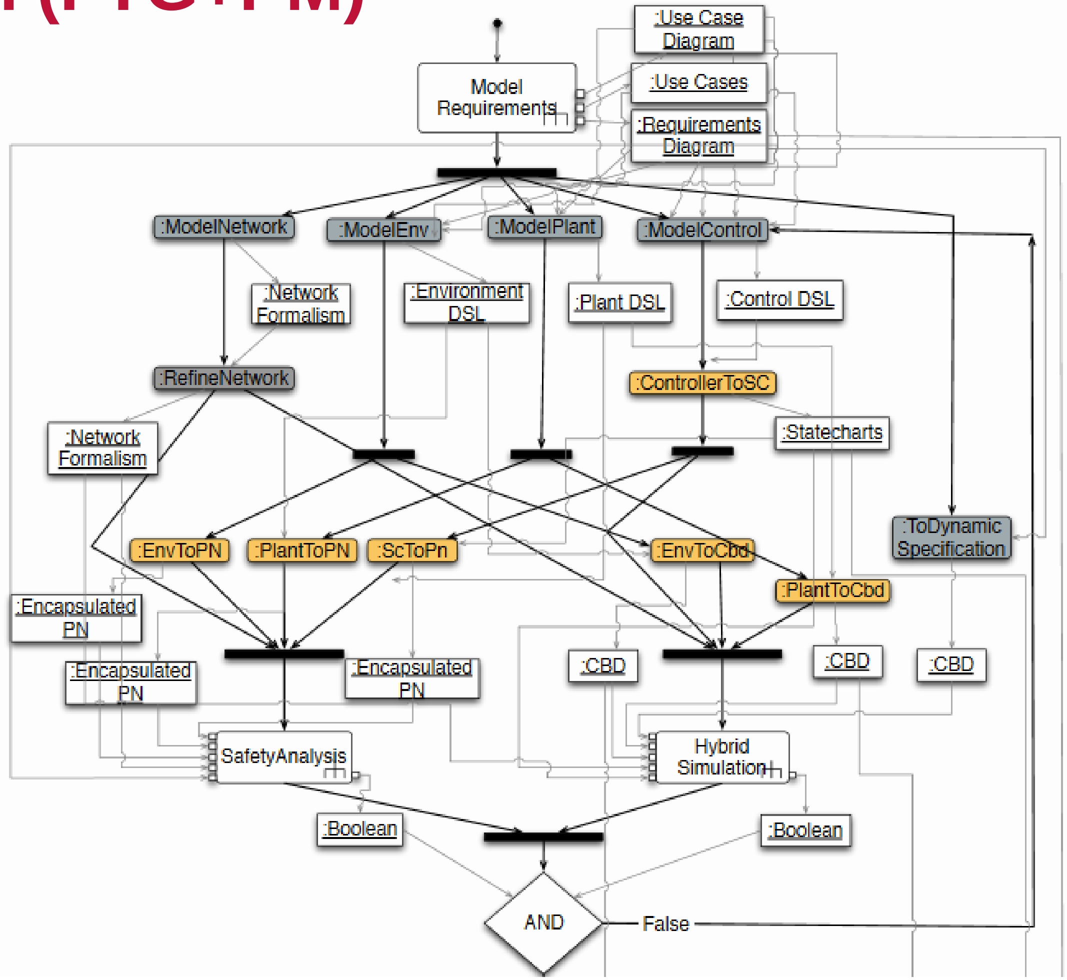
- **Line following using a PID controller**
  - http://www.nxtprograms.com/line_follower/steps.html
  - http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html





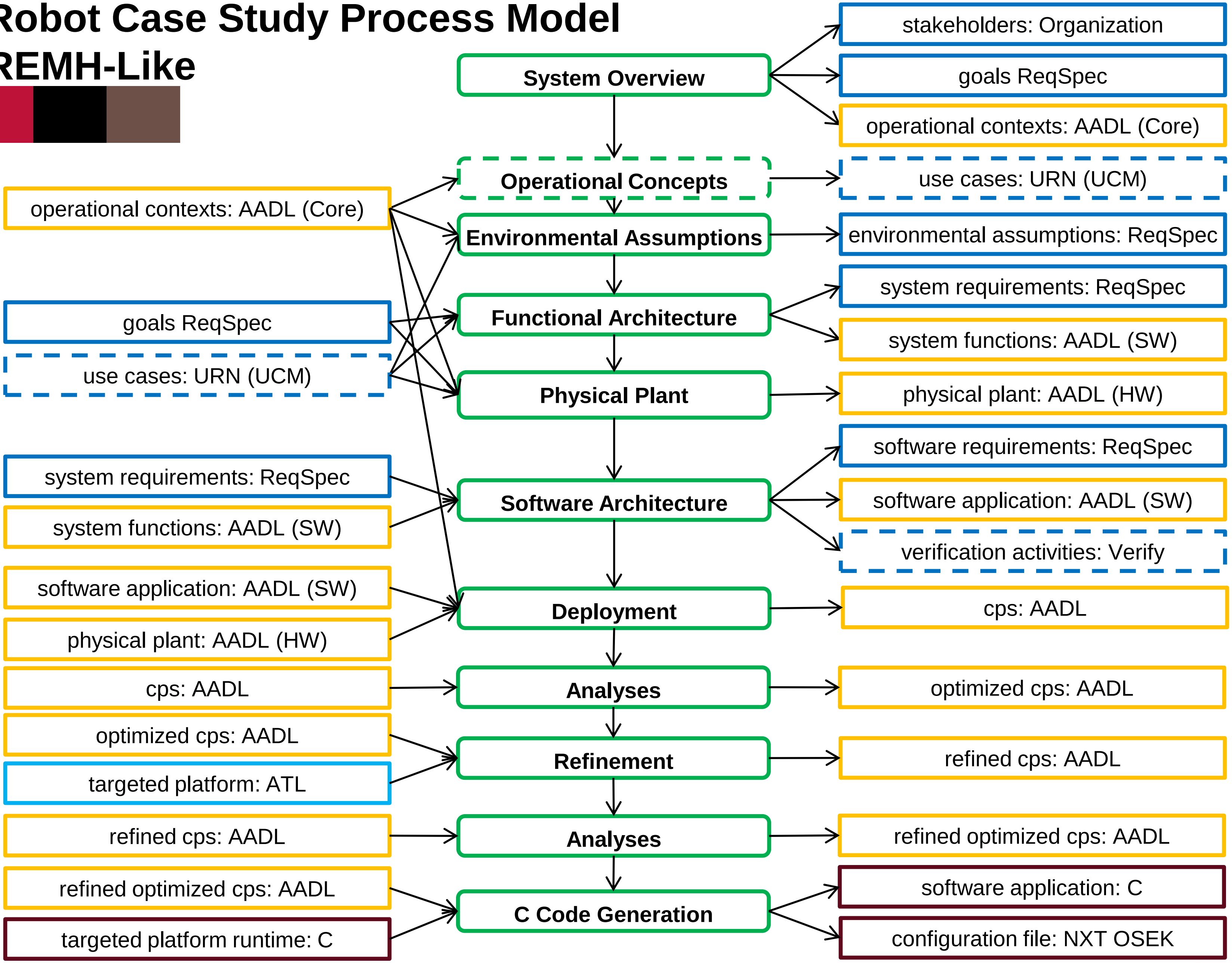
Formalisms Transformation Graph and Process Modem (FTG+PM)



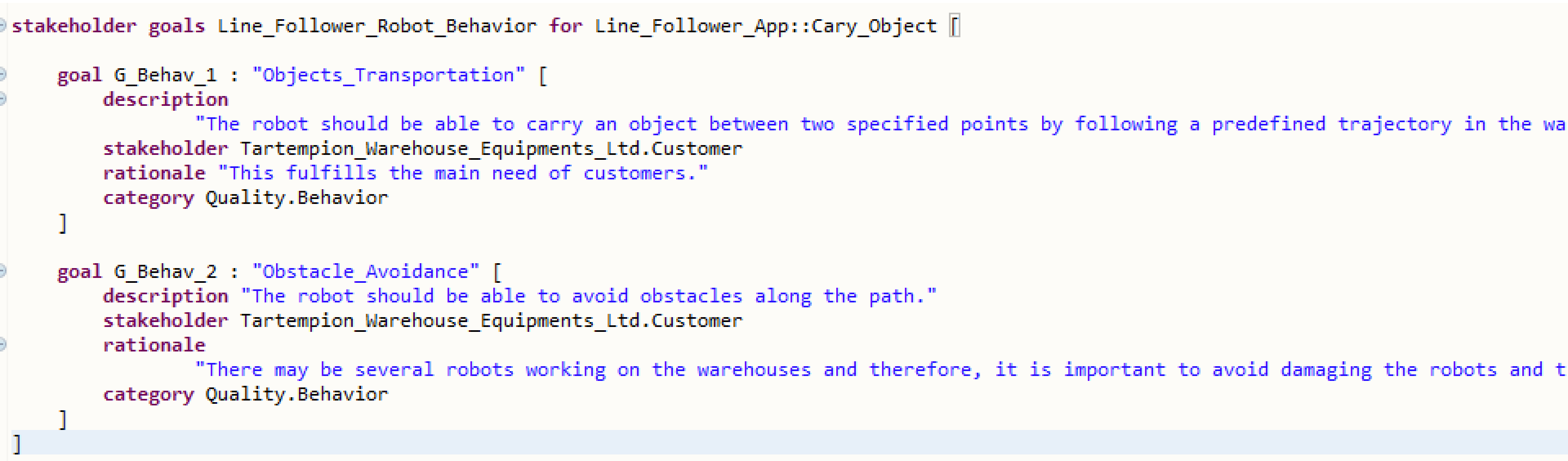


- ***The FTG+PM Framework for Multi-Paradigm Modelling: An Automotive Case Study***, **Mustafiz, Denil, Lucio and Vangheluwe, 2017**

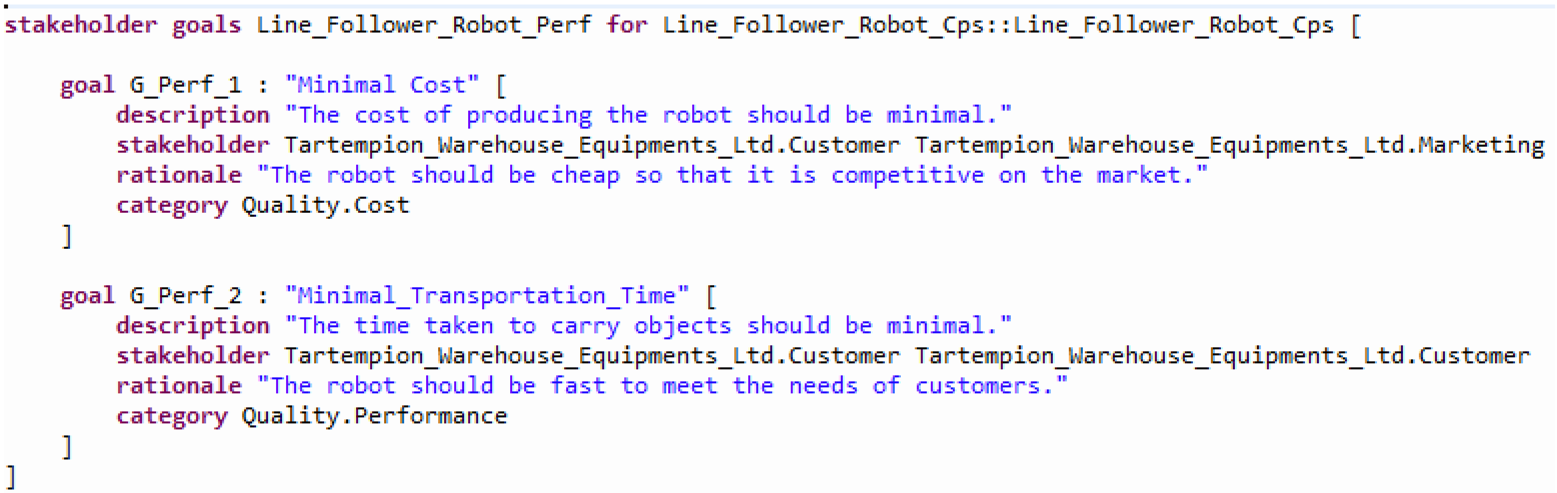
Robot Case Study Process Model REMH-Like


System Overview
stakeholders: Organization
goals ReqSpec
operational contexts: AADL (Core)
Operational Concepts
use cases: URN (UCM)
operational contexts: AADL (Core)
Environmental Assumptions
environmental assumptions: ReqSpec
Functional Architecture
system requirements: ReqSpec
system functions: AADL (SW)
goals ReqSpec
use cases: URN (UCM)
Physical Plant
physical plant: AADL (HW)
Software Architecture
software requirements: ReqSpec
software application: AADL (SW)
verification activities: Verify
system requirements: ReqSpec
system functions: AADL (SW)
Deployment
cps: AADL
software application: AADL (SW)
physical plant: AADL (HW)
Analyses
cps: AADL
optimized cps: AADL
Refinement
optimized cps: AADL
targeted platform: ATL
refined cps: AADL
Analyses
refined cps: AADL
refined optimized cps: AADL
C Code Generation
refined optimized cps: AADL
targeted platform runtime: C
software application: C
configuration file: NXT OSEK

System Overview: Stakeholder Goals

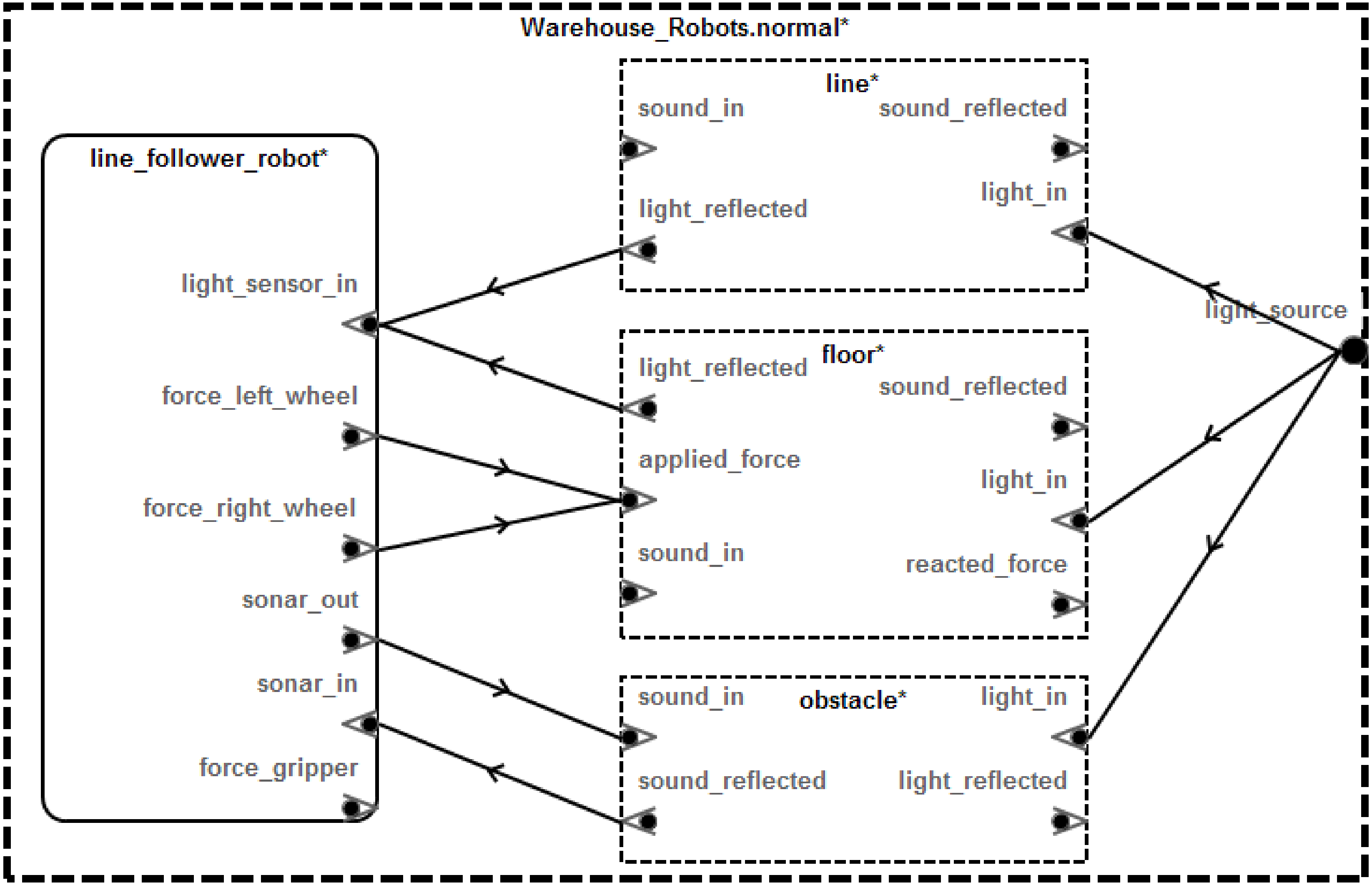


```
stakeholder goals Line_Follower_Robot_Behavior for Line_Follower_App::Cary_Object [

    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
    ]
]
```

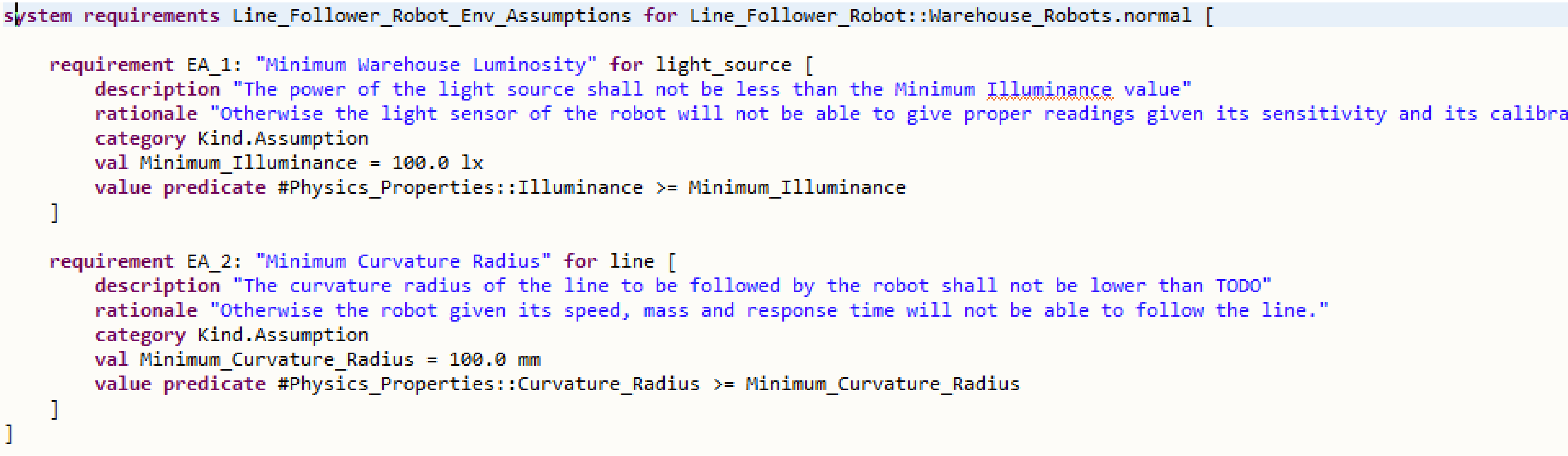


```
stakeholder goals Line_Follower_Robot_Perf for Line_Follower_Robot_Cps::Line_Follower_Robot_Cps [

    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 Overview: Normal Operational Context

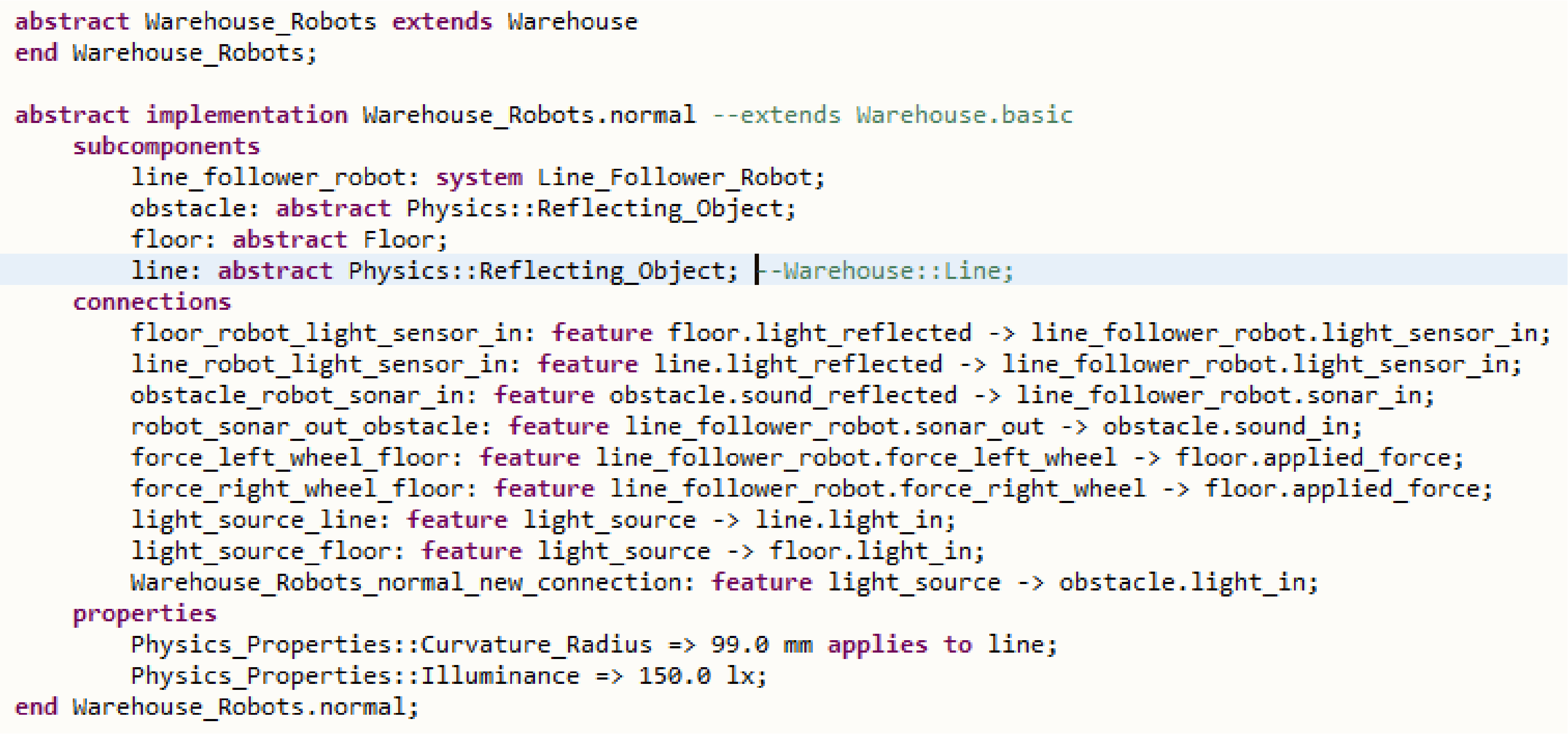


Environmental Assumptions



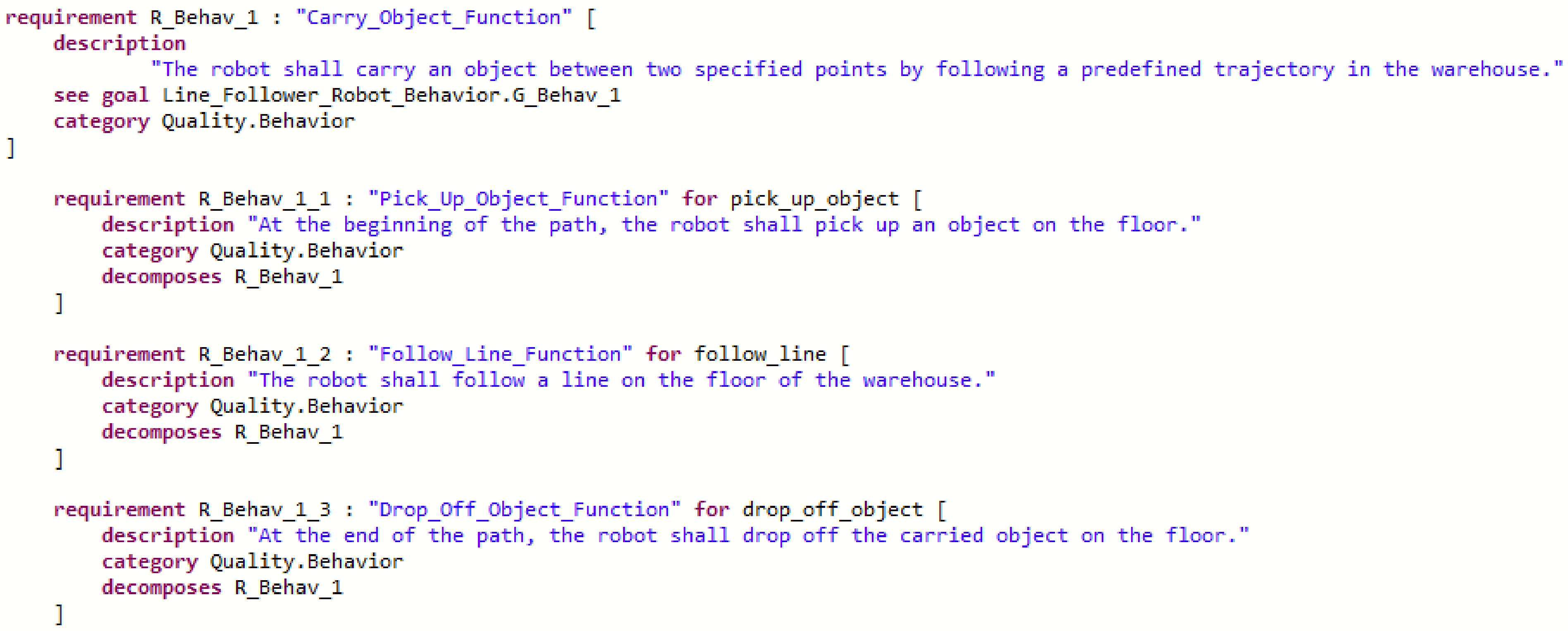
```
system requirements Line_Follower_Robot_Env_Assumptions for Line_Follower_Robot::Warehouse_Robots.normal [

    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
    ]
]
```

Properties on Normal Context



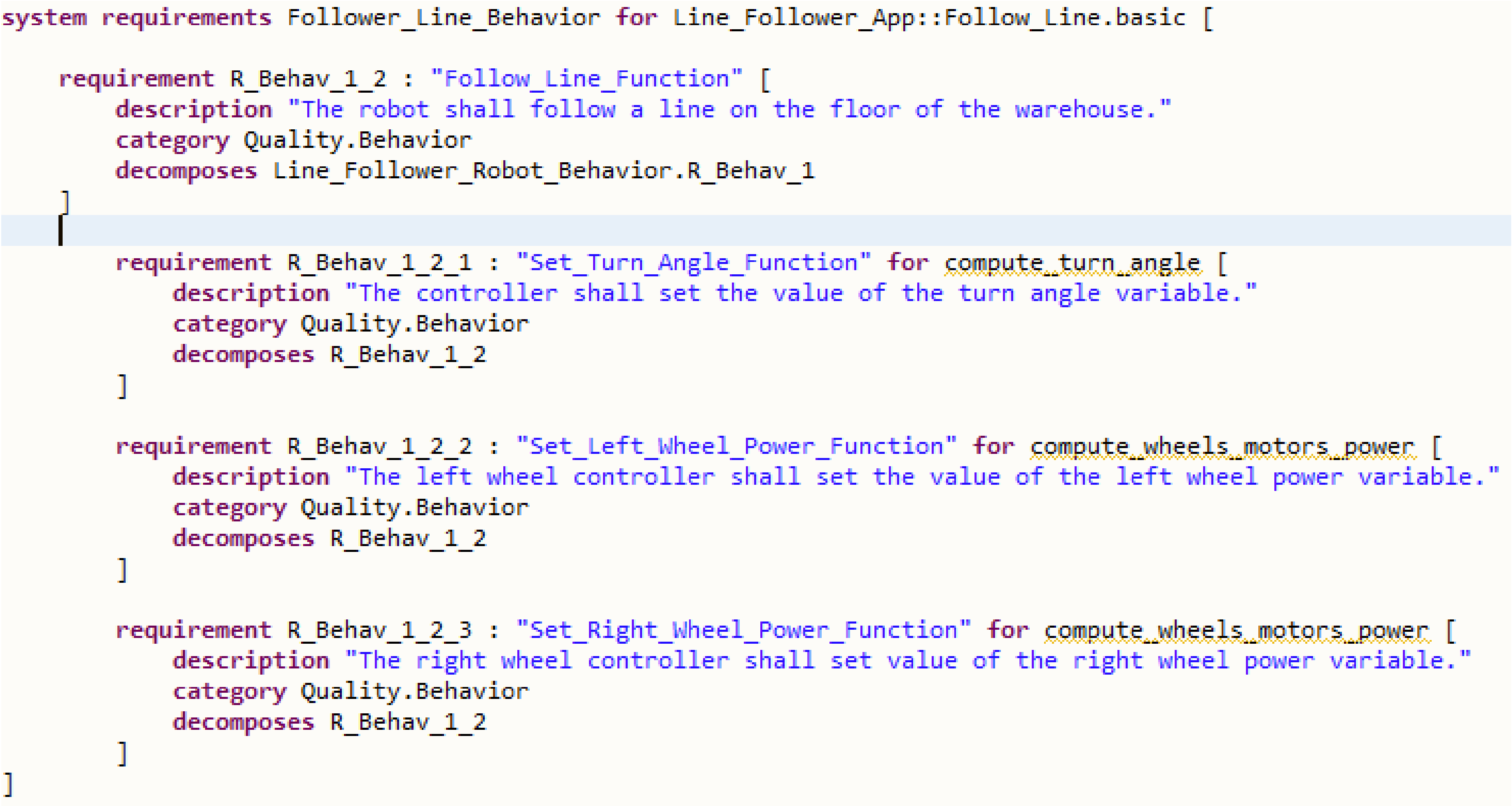
```
abstract Warehouse_Robots extends Warehouse
end Warehouse_Robots;

abstract implementation Warehouse_Robots.normal --extends Warehouse.basic
    subcomponents
        line_follower_robot: system Line_Follower_Robot;
        obstacle: abstract Physics::Reflecting_Object;
        floor: abstract Floor;
        line: abstract Physics::Reflecting_Object; --Warehouse::Line;
    connections
        floor_robot_light_sensor_in: feature floor.light_reflected -> line_follower_robot.light_sensor_in;
        line_robot_light_sensor_in: feature line.light_reflected -> line_follower_robot.light_sensor_in;
        obstacle_robot_sonar_in: feature obstacle.sound_reflected -> line_follower_robot.sonar_in;
        robot_sonar_out_obstacle: feature line_follower_robot.sonar_out -> obstacle.sound_in;
        force_left_wheel_floor: feature line_follower_robot.force_left_wheel -> floor.applied_force;
        force_right_wheel_floor: feature line_follower_robot.force_right_wheel -> floor.applied_force;
        light_source_line: feature light_source -> line.light_in;
        light_source_floor: feature light_source -> floor.light_in;
        Warehouse_Robots_normal_new_connection: feature light_source -> obstacle.light_in;
    properties
        Physics_Properties::Curvature_Radius => 99.0 mm applies to line;
        Physics_Properties::Illuminance => 150.0 lx;
end Warehouse_Robots.normal;
```

Functional Architecture:
High Level Requirements



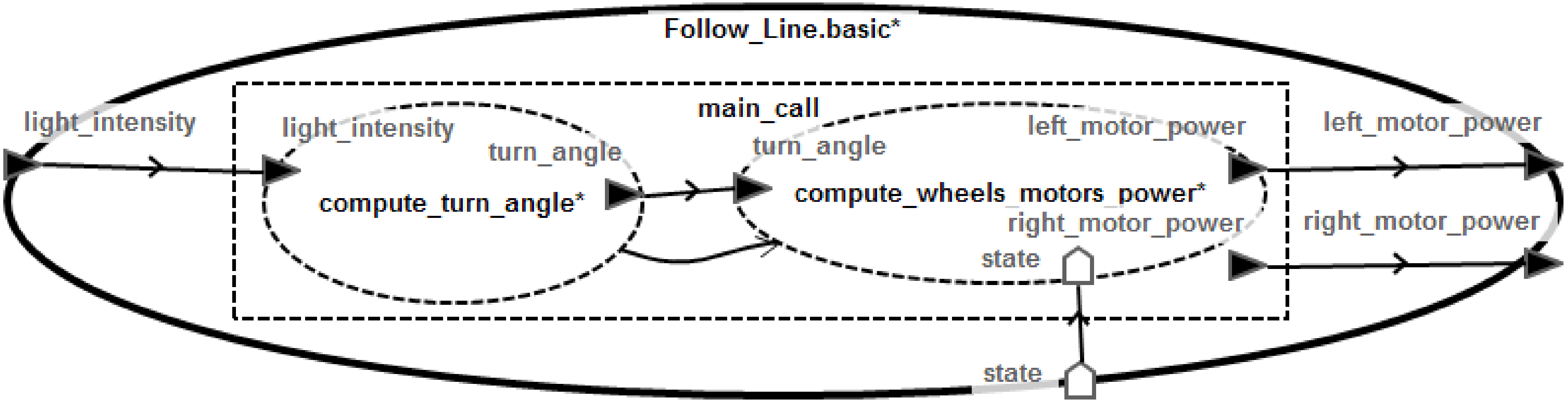
```
requirement R_Behav_1 : "Carry_Object_Function" [
    description
            "The robot shall carry an object between two specified points by following a predefined trajectory in the warehouse."
    see goal Line_Follower_Robot_Behavior.G_Behav_1
    category Quality.Behavior
]

    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

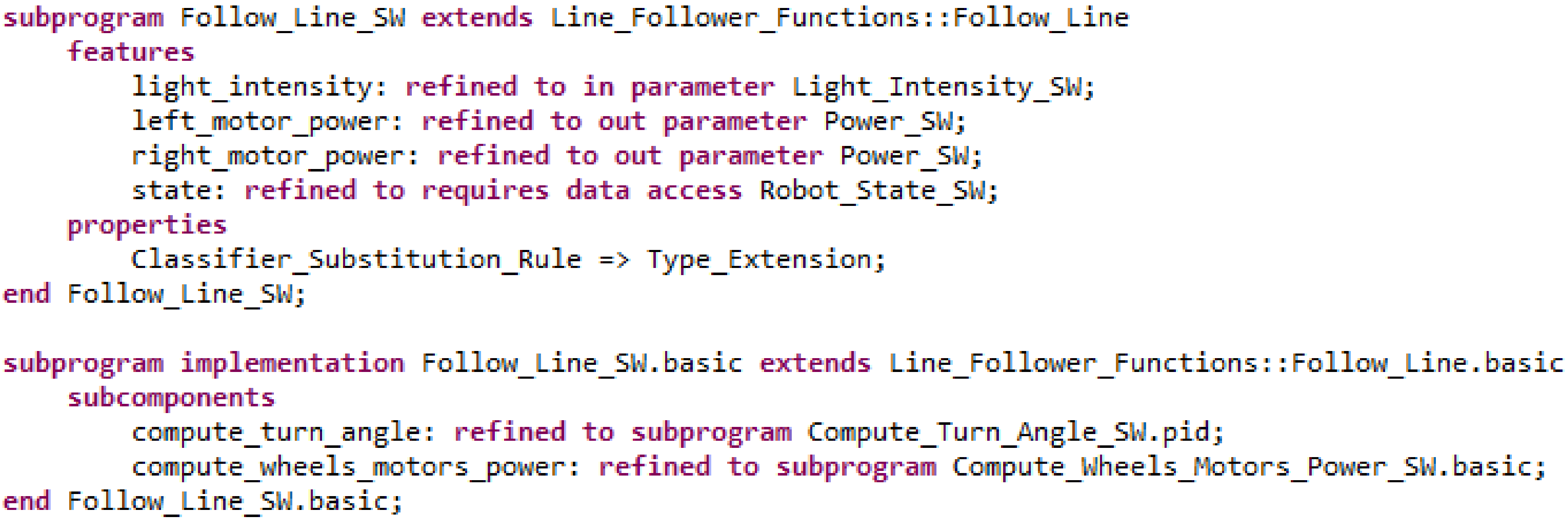


```
system requirements Follower_Line_Behavior for Line_Follower_App::Follow_Line.basic [

    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

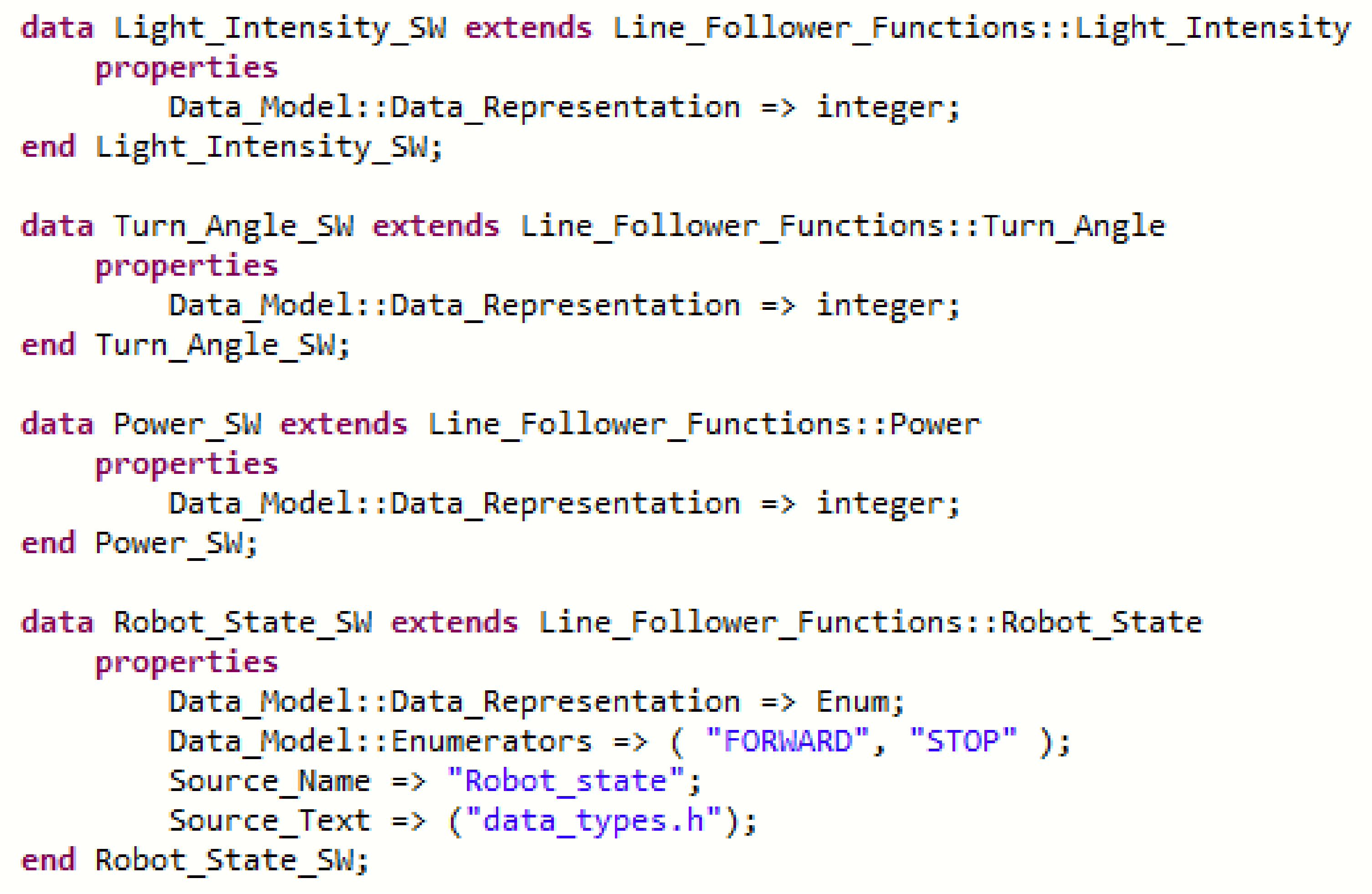


Software Architecture:
Follow Line Subprogram



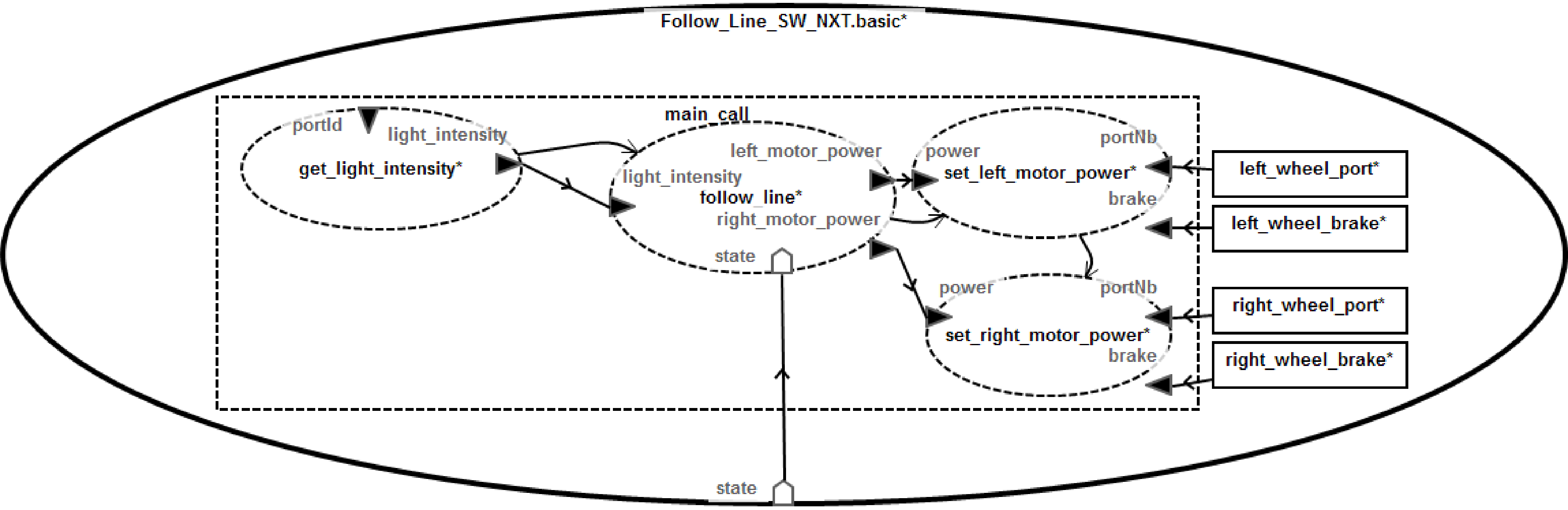
```
subprogram Follow_Line_SW extends Line_Follower_Functions::Follow_Line
    features
        light_intensity: refined to in parameter Light_Intensity_SW;
        left_motor_power: refined to out parameter Power_SW;
        right_motor_power: refined to out parameter Power_SW;
        state: refined 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

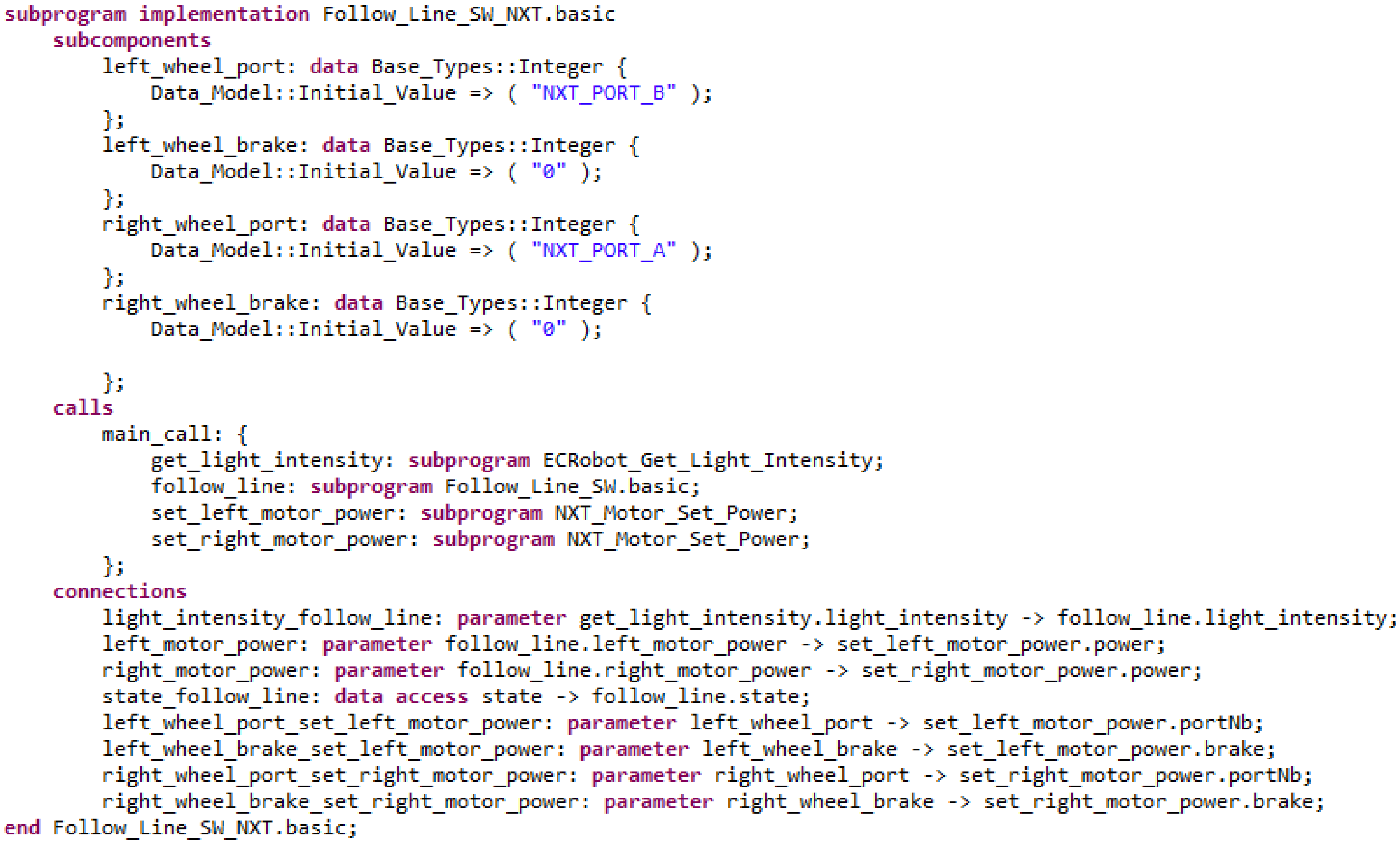


```
data Light_Intensity_SW extends Line_Follower_Functions::Light_Intensity
    properties
        Data_Model::Data_Representation => integer;
end Light_Intensity_SW;

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

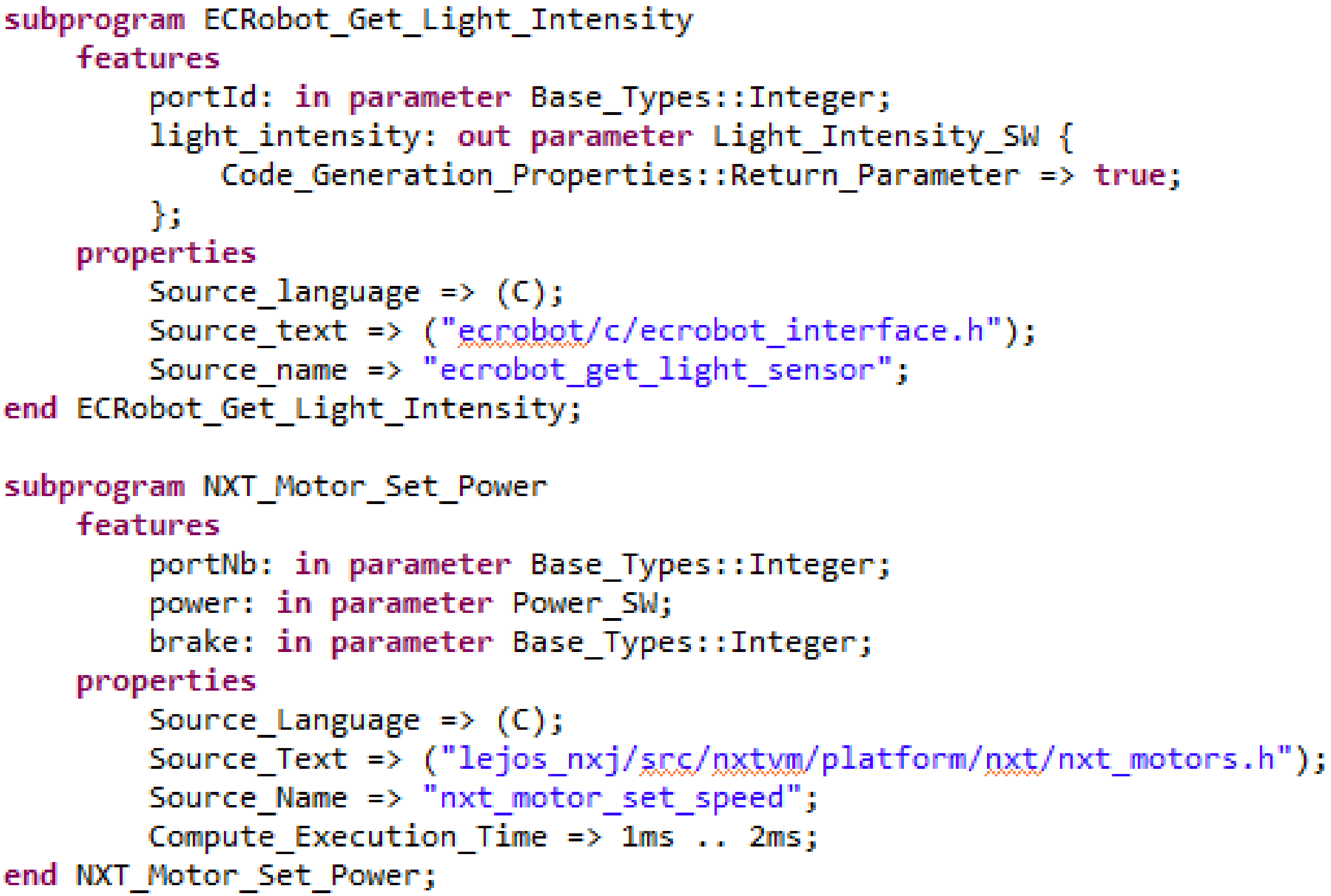


Software Architecture: NXT Follow Line Subprogram



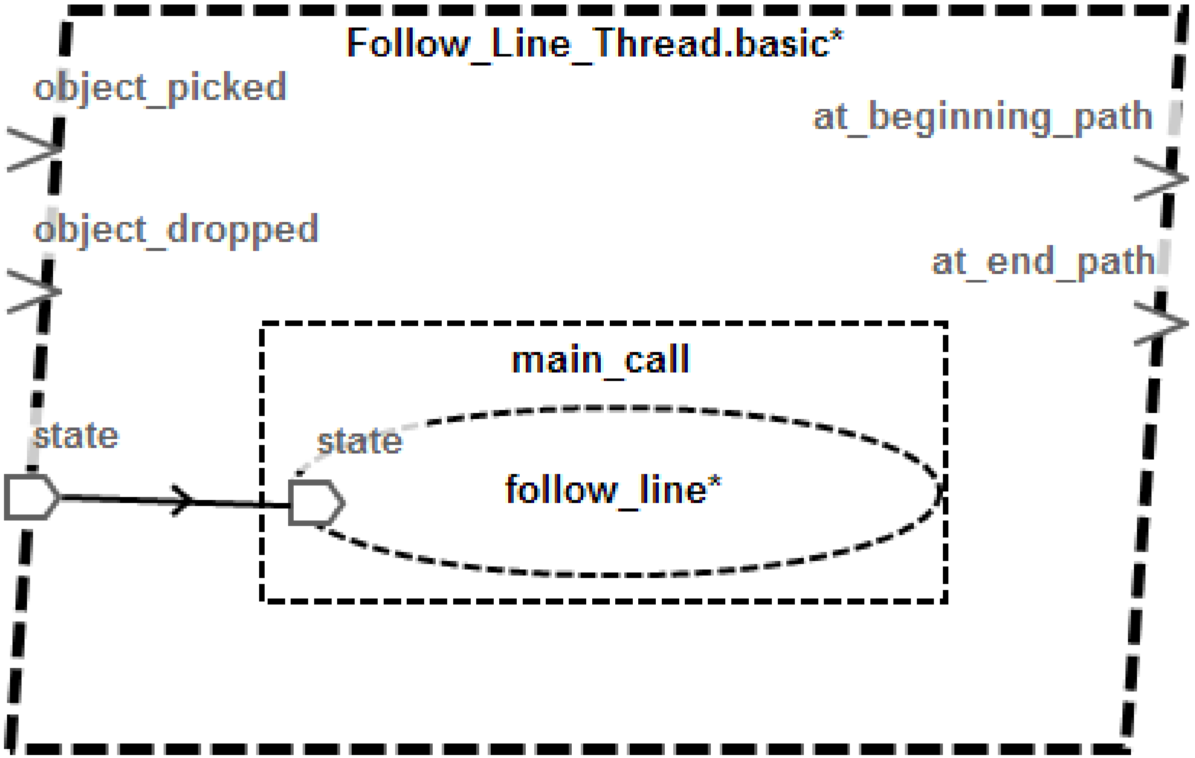
```
subprogram implementation Follow_Line_SW_NXT.basic
    subcomponents
        left_wheel_port: data Base_Types::Integer {
            Data_Model::Initial_Value => ( "NXT_PORT_B" );
        };
        left_wheel_brake: data Base_Types::Integer {
            Data_Model::Initial_Value => ( "0" );
        };
        right_wheel_port: data Base_Types::Integer {
            Data_Model::Initial_Value => ( "NXT_PORT_A" );
        };
        right_wheel_brake: data Base_Types::Integer {
            Data_Model::Initial_Value => ( "0" );

        };
    calls
        main_call: {
            get_light_intensity: subprogram ECRobot_Get_Light_Intensity;
            follow_line: subprogram Follow_Line_SW.basic;
            set_left_motor_power: subprogram NXT_Motor_Set_Power;
            set_right_motor_power: subprogram NXT_Motor_Set_Power;
        };
    connections
        light_intensity_follow_line: parameter get_light_intensity.light_intensity -> follow_line.light_intensity;
        left_motor_power: parameter follow_line.left_motor_power -> set_left_motor_power.power;
        right_motor_power: parameter follow_line.right_motor_power -> set_right_motor_power.power;
        state_follow_line: data access state -> follow_line.state;
        left_wheel_port_set_left_motor_power: parameter left_wheel_port -> set_left_motor_power.portNb;
        left_wheel_brake_set_left_motor_power: parameter left_wheel_brake -> set_left_motor_power.brake;
        right_wheel_port_set_right_motor_power: parameter right_wheel_port -> set_right_motor_power.portNb;
        right_wheel_brake_set_right_motor_power: parameter right_wheel_brake -> set_right_motor_power.brake;
end Follow_Line_SW_NXT.basic;
```

Software Architecture:
NXT ECRobot Library Subprograms

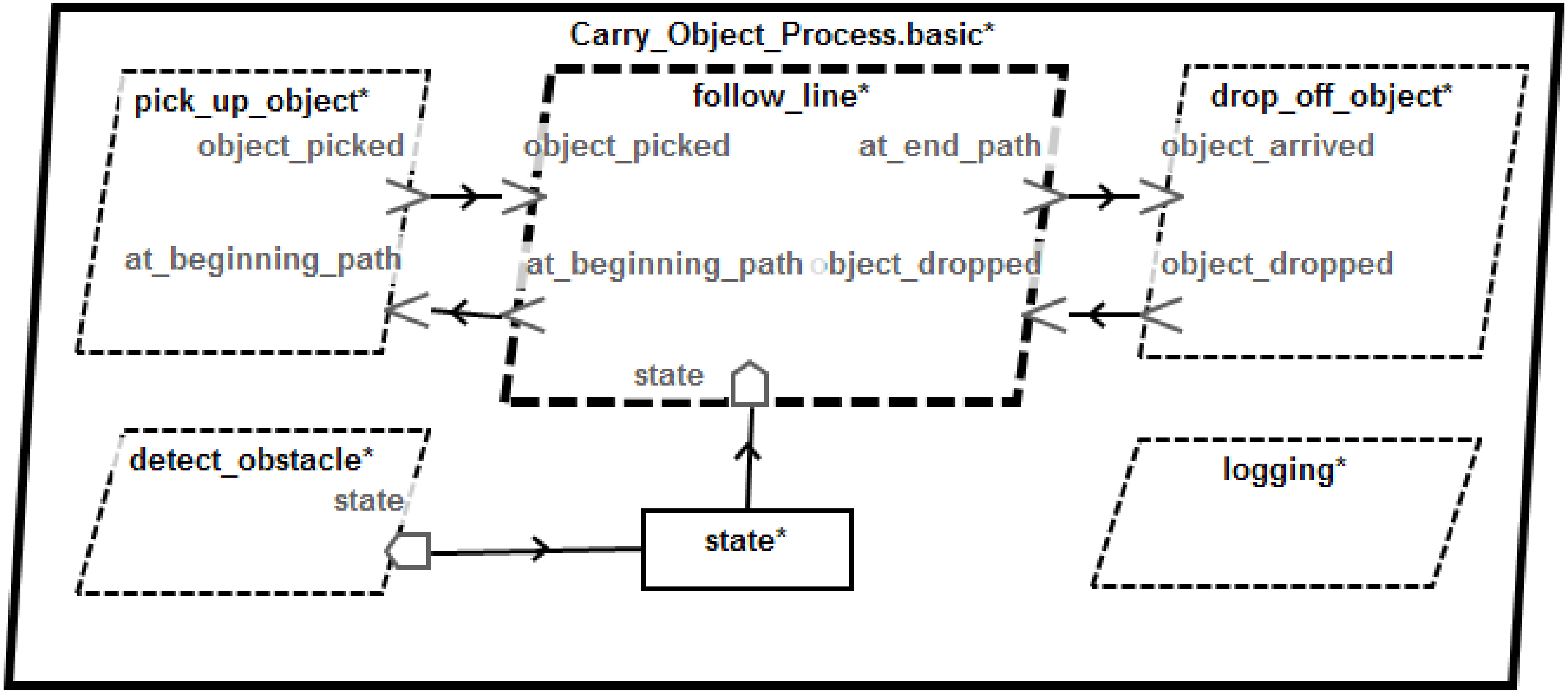


```
subprogram ECRobot_Get_Light_Intensity
    features
        portId: in parameter Base_Types::Integer;
        light_intensity: out parameter Light_Intensity_SW {
            Code_Generation_Properties::Return_Parameter => true;
        };
    properties
        Source_language => (C);
        Source_text => ("ecrobot/c/ecrobot_interface.h");
        Source_name => "ecrobot_get_light_sensor";
end ECRobot_Get_Light_Intensity;

subprogram NXT_Motor_Set_Power
    features
        portNb: in parameter Base_Types::Integer;
        power: in parameter Power_SW;
        brake: in parameter Base_Types::Integer;
    properties
        Source_Language => (C);
        Source_Text => ("lejos_nxj/src/nxtvm/platform/nxt/nxt_motors.h");
        Source_Name => "nxt_motor_set_speed";
        Compute_Execution_Time => 1ms .. 2ms;
end NXT_Motor_Set_Power;
```

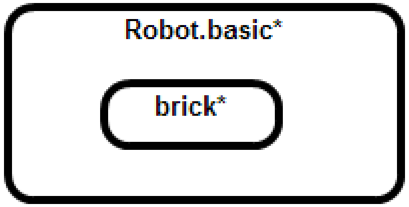
Software Architecture: Follow Line Thread

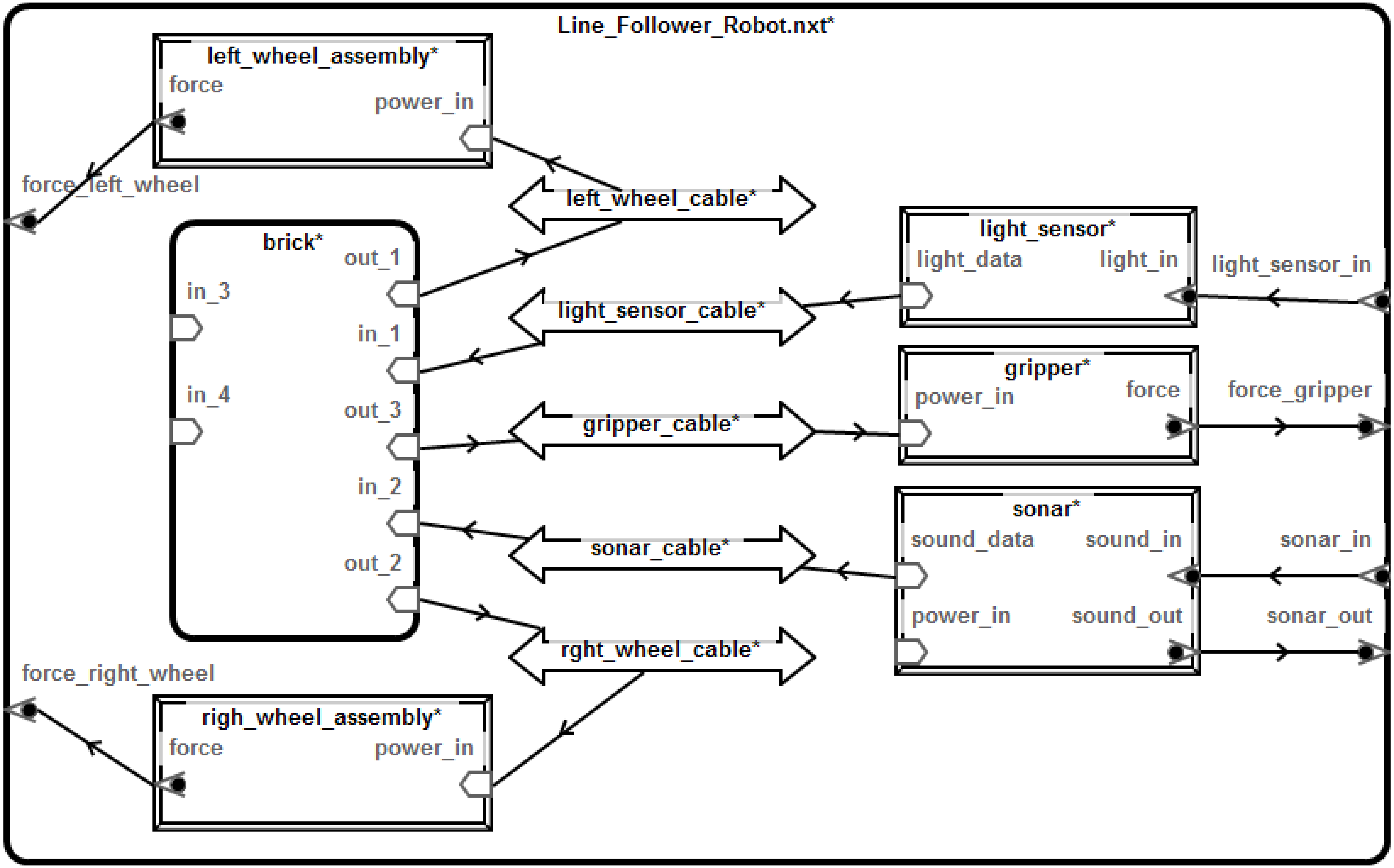


Software Architecture: Carry Objects Process

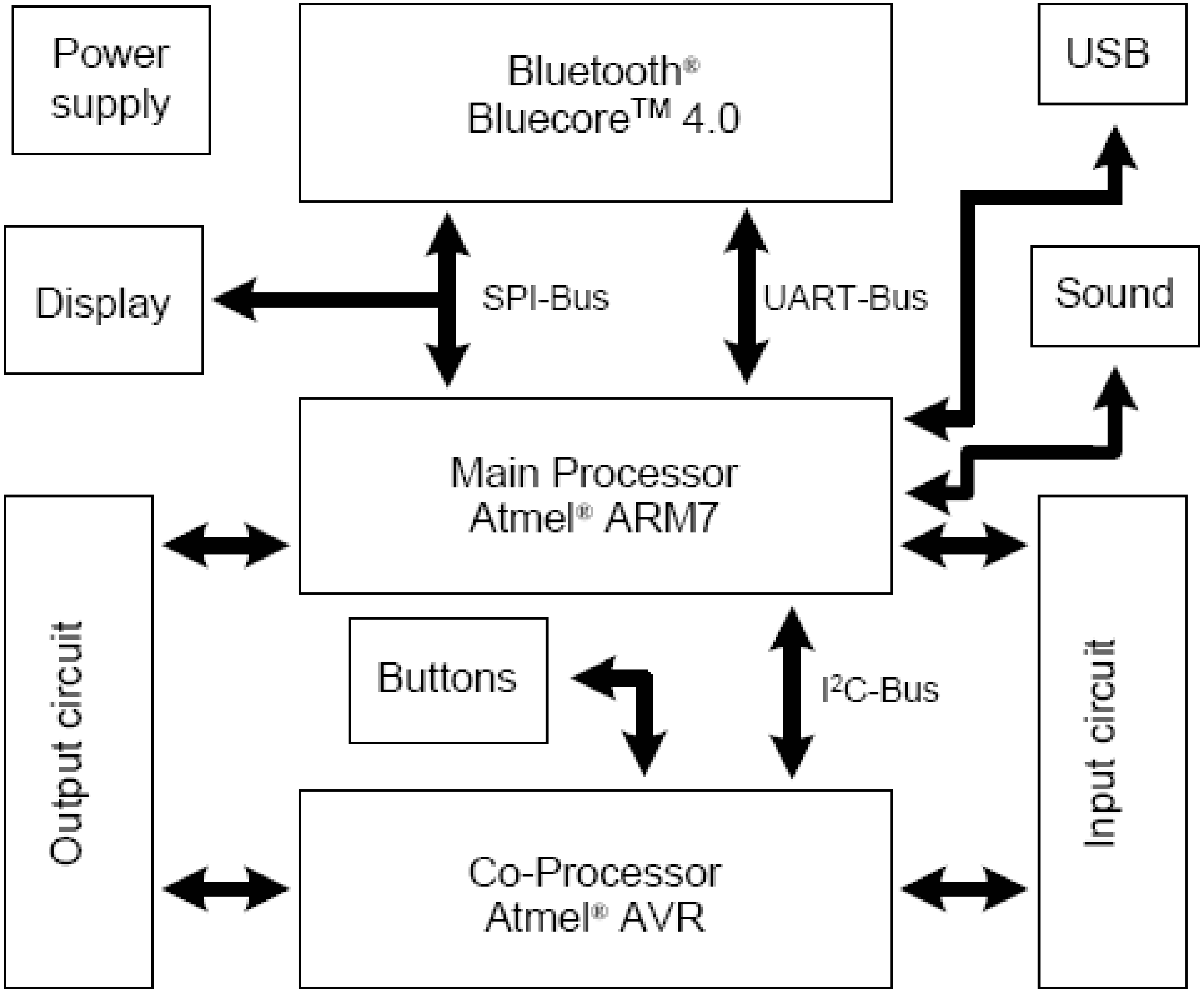


Physical Plant Model: Line Follower NXT Robot

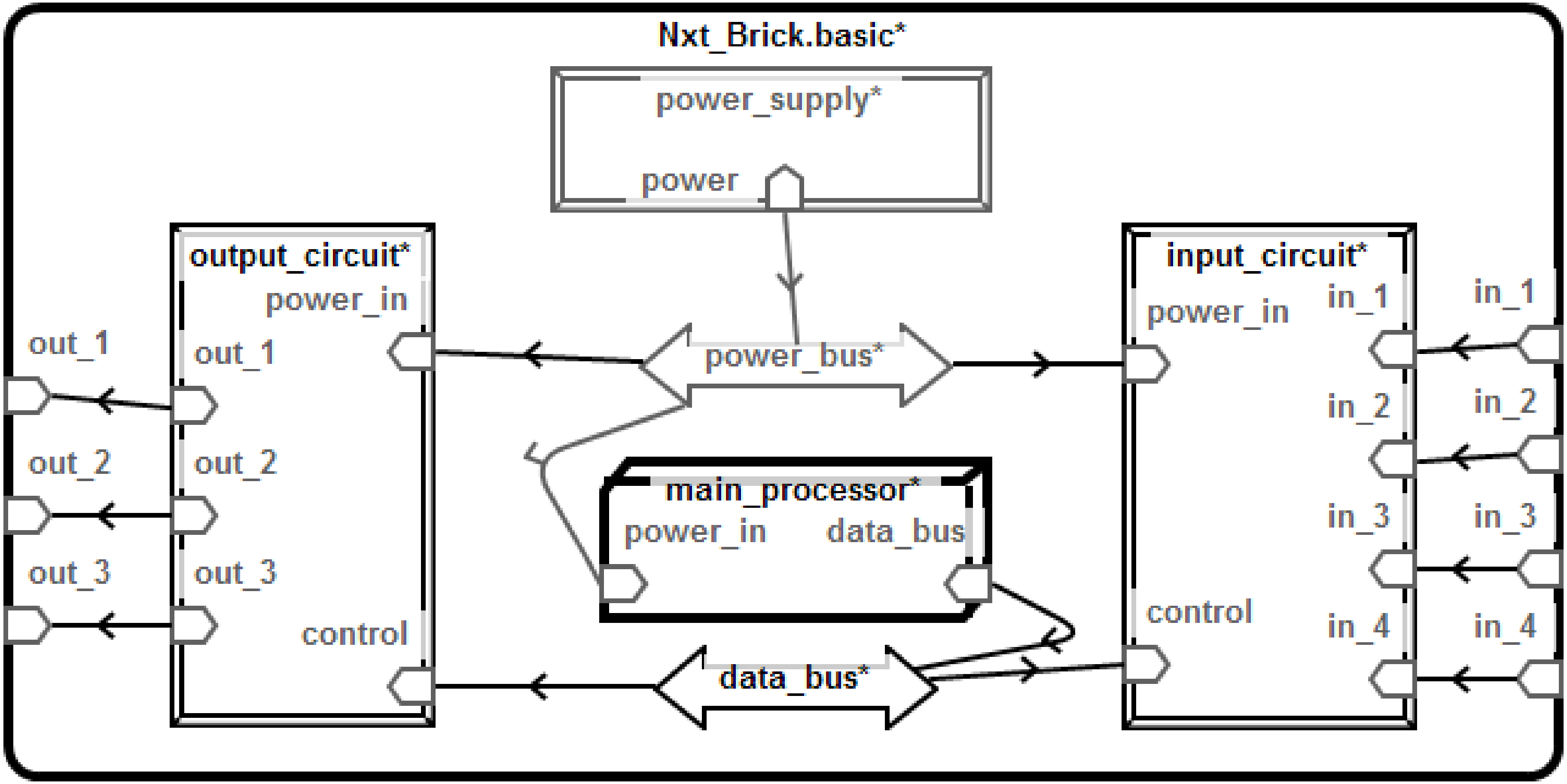




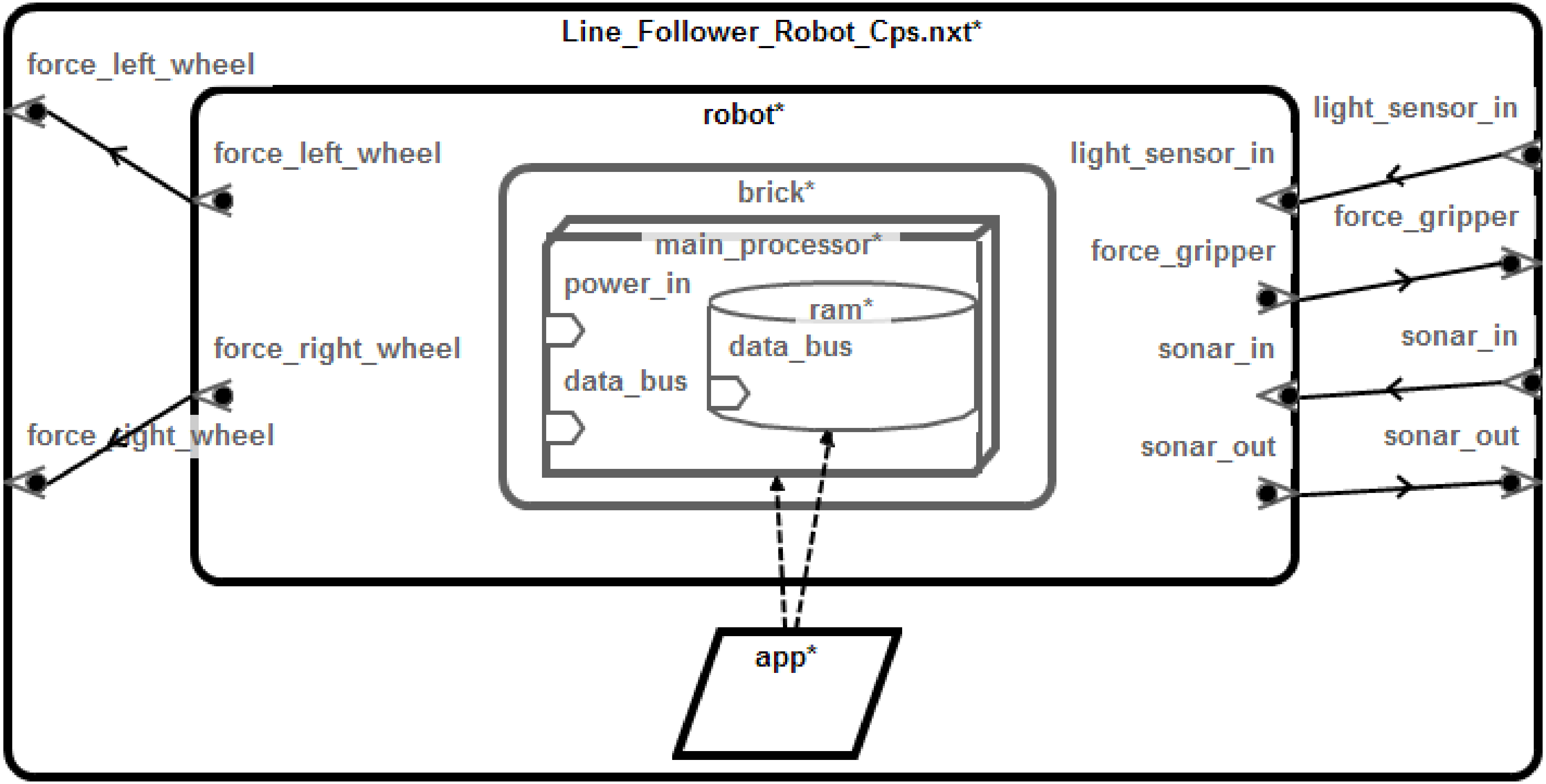
NXT Brick Hardware Kit Specification



Physical Plant Model: NXT Brick



Deployment:
Line Follower NXT Robot CPS



Analyses

- **Measure execution time of subprograms on brick**

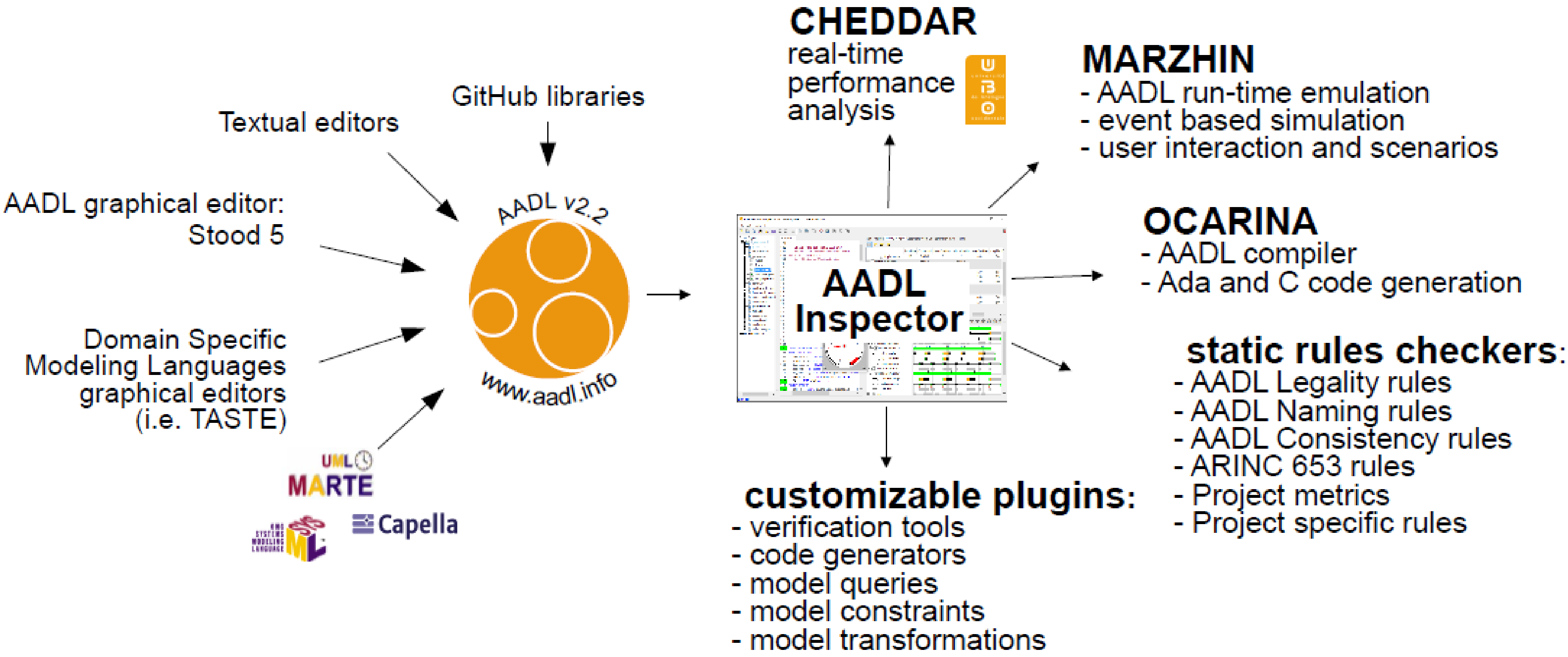
- **Latency in steering the robot**
  - Given the minimal curvature radius environmental assumption, mass of robot and carried object and latency, what is the maximum speed?

- **Scheduling of three threads:**
  - Object carrying
  - Obstacle detection
  - Logging

AADL Inspector

Ellidiss Technologies

- **Commercially developed by Ellidiss Technology**
  - Free license obtained for training school
  - https://www.ellidiss.com/products/aadl-inspector/



GitHub libraries

Textual editors

AADL graphical editor: Stood 5

Domain Specific Modeling Languages graphical editors (i.e. TASTE)

UML MARTE

SysML

Capella

AADL v2.2

www.aadl.info

AADL Inspector

CHEDDAR
real-time performance analysis

MARZHIN
- AADL run-time emulation
- event based simulation
- user interaction and scenarios

OCARINA
- AADL compiler
- Ada and C code generation

static rules checkers:
- AADL Legality rules
- AADL Naming rules
- AADL Consistency rules
- ARINC 653 rules
- Project metrics
- Project specific rules

customizable plugins:
- verification tools
- code generators
- model queries
- model constraints
- model transformations

Outline

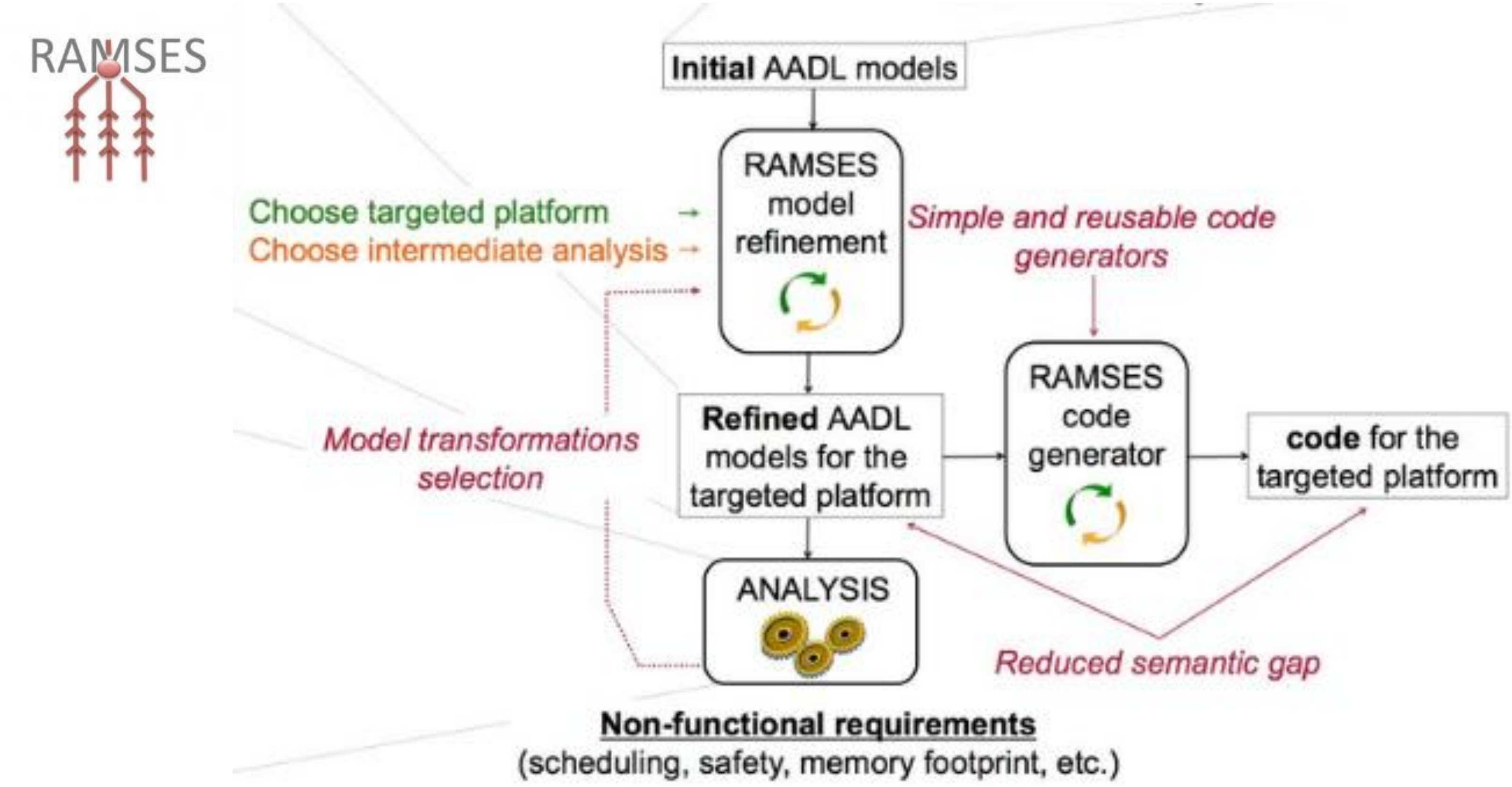
- Context
- AADL for CPSs: Lego Robot Case Study
- RAMSES and Code Generation for Case Study
- Mixed Criticality Scheduling

MEM4CSD

- **Model-based Engineering Methods for Complex Systems Design**
- **https://mem4csd.telecom-paristech.fr/**
- **Strategy: develop tools that can be used independently**
- **Website:**

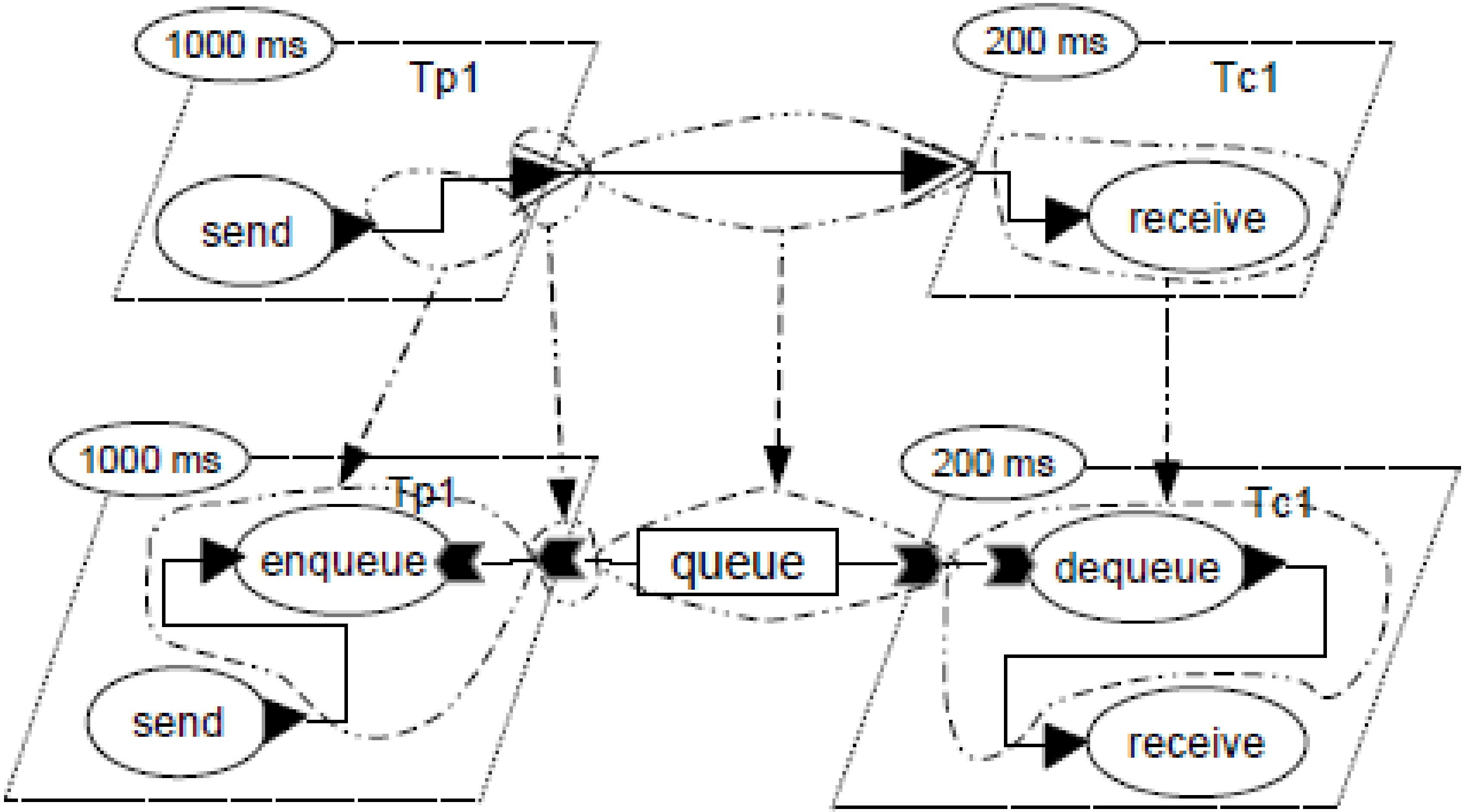
RAMSES

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

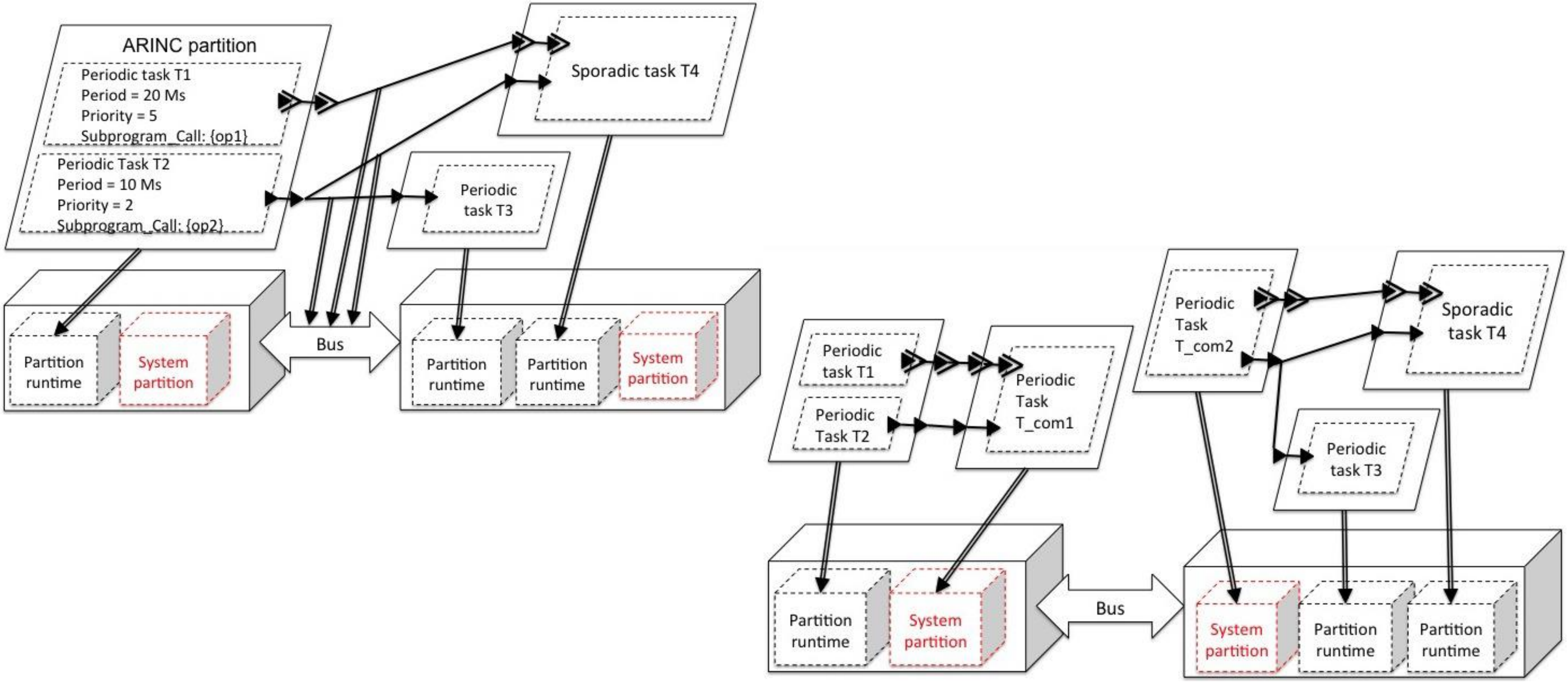


RAMSES
Initial AADL models
RAMSES model refinement
Choose targeted platform →
Choose intermediate analysis →
Simple and reusable code generators
Model transformations selection
Refined AADL models for the targeted platform
RAMSES code generator
code for the targeted platform
ANALYSIS
Reduced semantic gap
Non-functional requirements
(scheduling, safety, memory footprint, etc.)

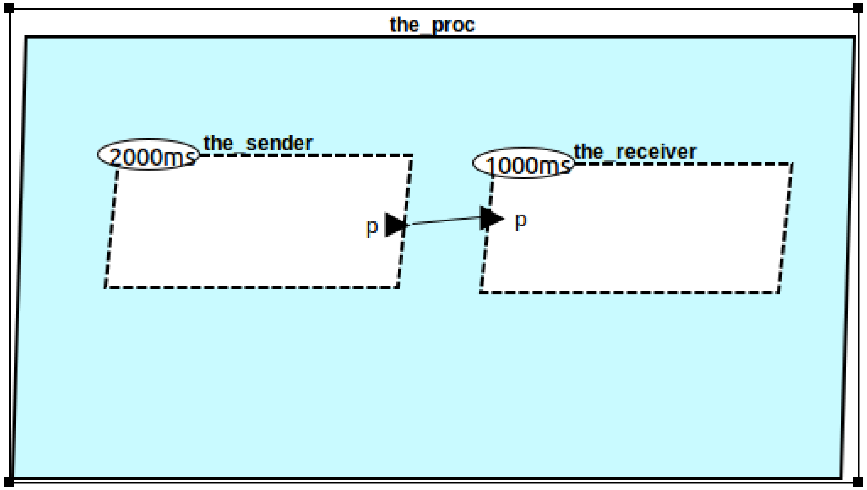
RAMSES Refinement Rules: Local Communications

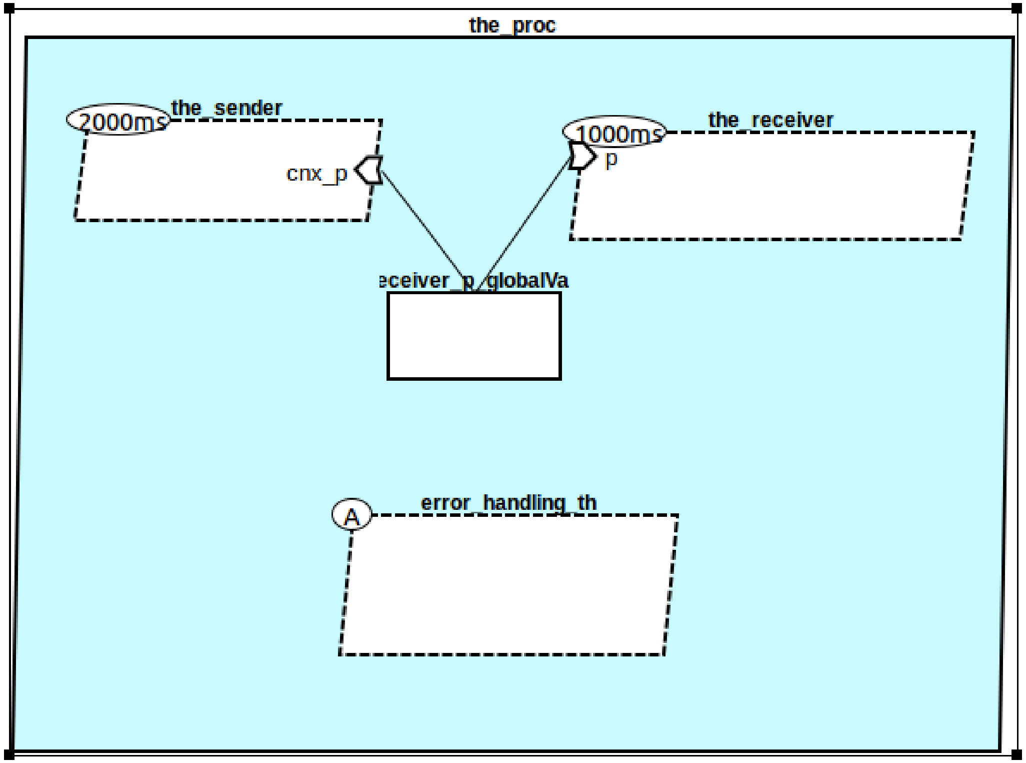


RAMSES Refinement Rules: Remote Communications



RAMSES Refinement Rules: Health Monitoring





Supported 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/

Demo: Generating Line Follower Code for the NXT Lego Mindstorm Robot

- **NXT OSEK: http://lejos-osek.sourceforge.net/**
  - C/C++ programming environment using GCC
  - C and C++ API for NXT Sensors, Motor and other devices
  - TOPPERS/ATK provided real-time multi tasking features proven in automotive industry
  - TOPPERS/JSP provided real-time multi tasking features complied with Japan original open RTOS specification µITRON 4.0
- **Overview of refined model**
- **Overview of generated code**
- **Deployment on the robot**
- **Running the robot**

RAMSES Implementation

- **Four steps process:**
- **Model validation:**
  - Model transformation AADL ➔ Validation report
  - Implemented in ATL
  - Resolute?
- **Refinement:**
  - Model transformation AADL ➔ AADL
  - Implemented in ATL
- **Code generation:**
  - AADL ➔ Code (C or ADA)
  - Implemented in Java
- **Code compilation**

Current Issues: Model Transformations

- **Low performances of ATL**

- **Low maintainability of ATL EMFTVM variant:**
  - Usability
  - Sometimes unpredictable behavior

Current Issues: Workflows

- **Current process of validation, refinement, code generation, compilation hard coded in the tool**
- **Several other workflows are required**
- **Customizable workflows required**
  - Too many clicks issue

Roadmap for RAMSES

- **RAMSES Runtime and AADL Runtime services**
  - See Etienne’s presentation today!

- **Integration of a workflow approach**
  - To be demonstrated with mixed-criticality scheduling

- **AADL←→ ROS (Robotics Operating System)**

Roadmap for RAMSES

- **Validation for refinement and code generation:**
  - Re-implement with Resolute?

- **Refinement: performance and expressivity issues**
  - Benchmark of main model transformation tools
    - QVT, TGG, Story diagrams, Viatra, Prolog, etc.
  - Study / develop incremental approaches (e.g. TGGs) for scalability
  - Study bi-directional approaches
  - Study model management approaches
  - Properties preservation of refinements (using graph transformations systems theory)

Outline

- Context
- AADL for CPSs: Lego Robot Case Study
- RAMSES and Code Generation for Case Study
- Mixed Criticality Scheduling

Mixed-Criticality Scheduling

- More and more functionalities share a common execution platform
- Due to safety requirements, only functionalities with the same level of criticality should share resources
- Leads to potential waste of computation power
- Mixed-Criticality model proposes an approach to execute high and low-criticality tasks in a single platform
- In nominal mode, tasks executed with an “optimistic” timing budget (e.g. a WCET)
- Upon a time failure event, switch to high criticality mode (only critical tasks are executed with extended timing budget)

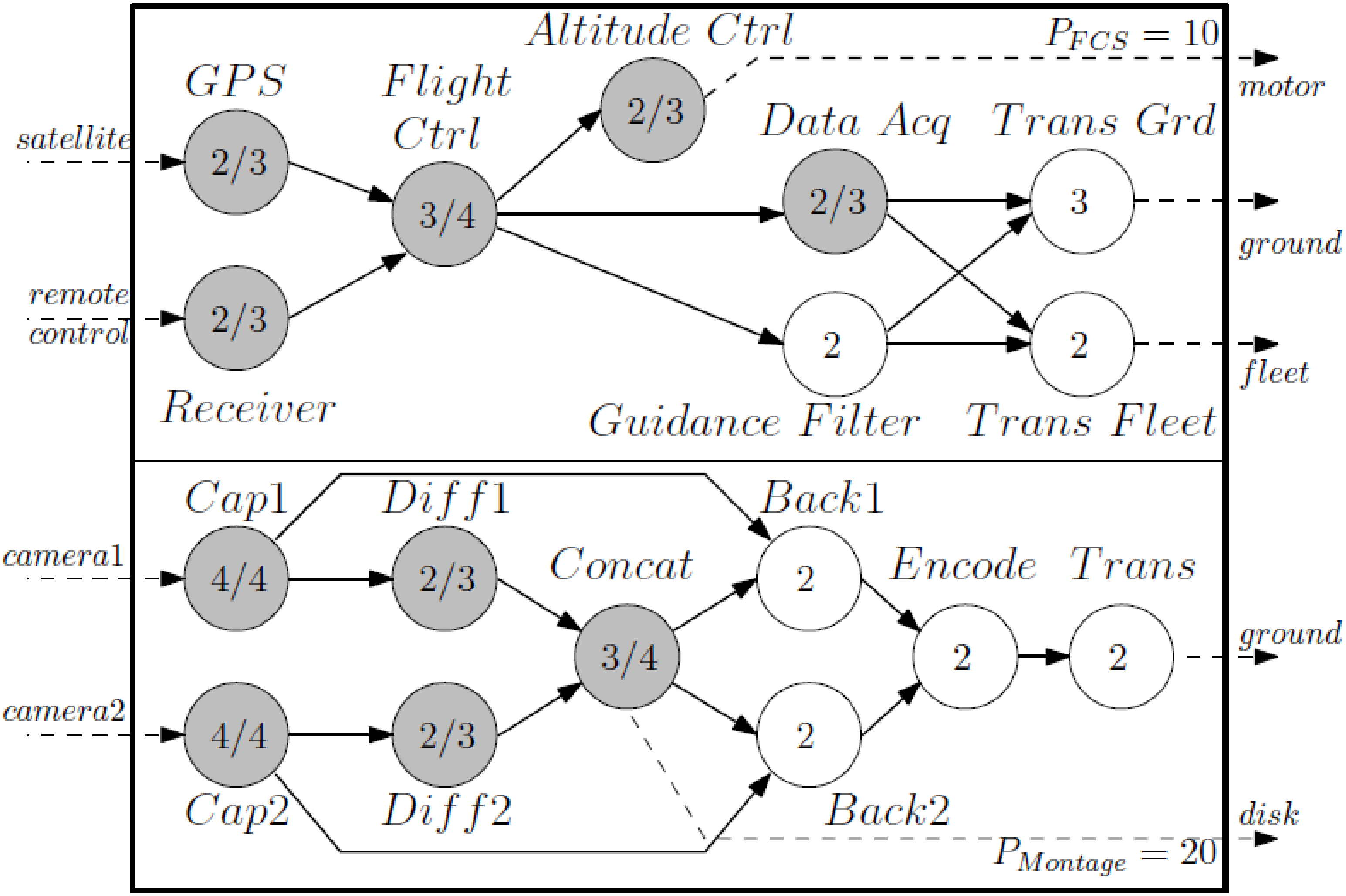
Mixed-Criticality Scheduling on Several Cores

- *Scheduling Multi-Periodic Mixed-Criticality DAGs on Multi-Core Architectures*, Medina et al., RTSS conference 2018.

- Developed the so-called MH-MCDAG heuristic.

- Principle: execute HI criticality tasks as late as possible, giving more flexibility for the execution of LO-criticality tasks

UAV Example


Altitude Ctrl
$P_{FCS} = 10$
GPS
Flight Ctrl
2/3
motor
Data Acq
Trans Grd
satellite
2/3
3/4
2/3
3
ground
remote control
2/3
2
2
fleet
Receiver
Guidance Filter
Trans Fleet
Cap1
Diff1
Back1
camera1
4/4
2/3
Concat
2
Encode
Trans
3/4
2
2
ground
camera2
4/4
2/3
2
Cap2
Diff2
Back2
disk
$P_{Montage} = 20$

MEM4CSD MC-DAG Toolset

- Developed a small DSL to represent MC DAGs
- Easy to use to develop schedulers
- Usable alone without an ADL
- Integrated with other AADL via model transformations
- Can also potentially be used with other Architecture Description Languages (AUTOSAR, AF3, MARTE, etc.)

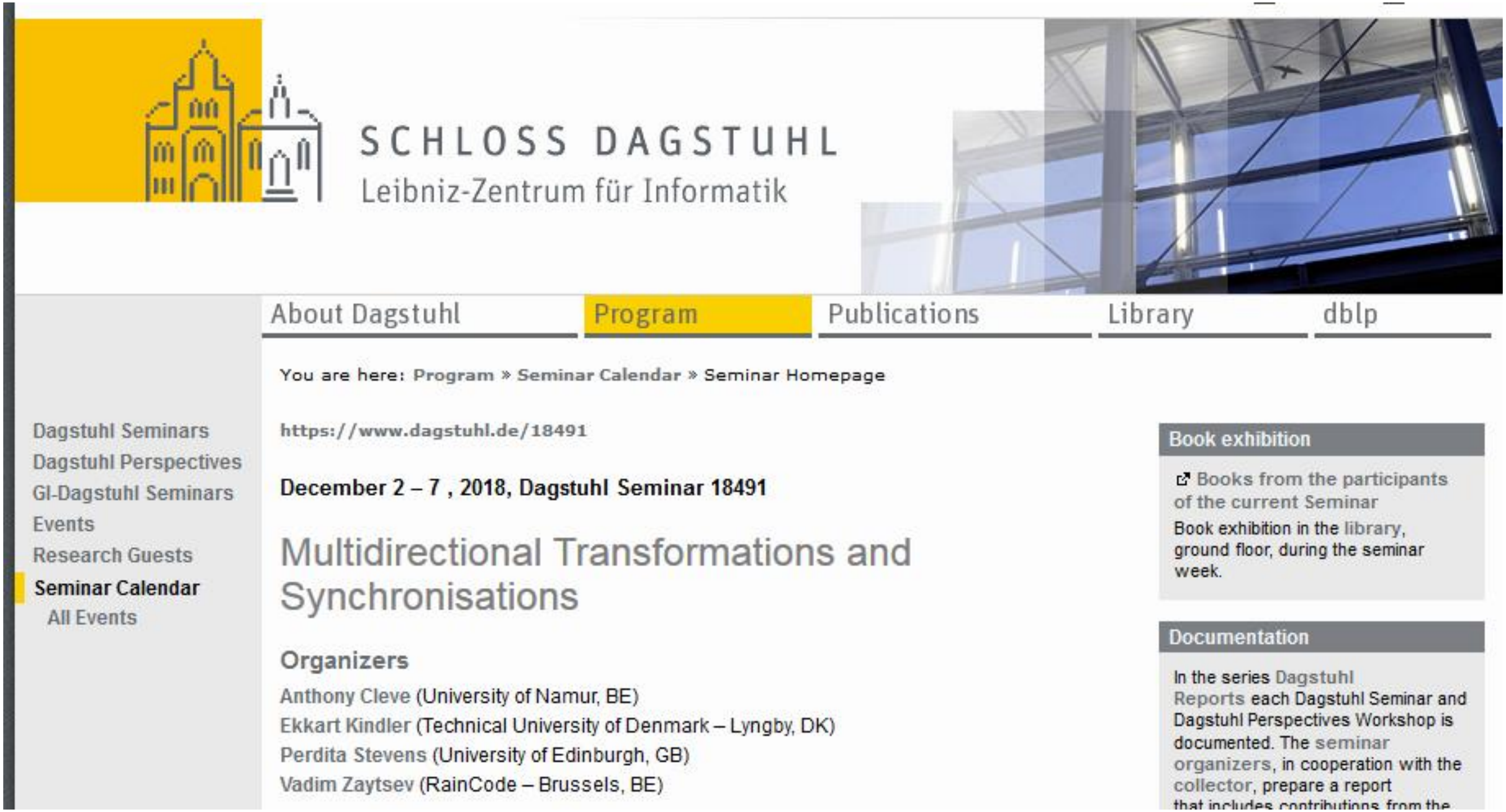
MC-DAG Demo (WIP)

- Workflow
- Model Synchronization
- Scheduling
- Static scheduling table properties
- Code generation

More on Model Synchronization

- **RAMSES and AADL are very good case studies!**
  - AADL: Rich and complex language
  - Need for incremental approaches
- **MoTE – TGG-based Model Transformation Engine**
- **Graph transformations as programming paradigm**
  - "Introduction to Graph-Oriented Programming“
  - Olivier Rey (CEO of GraphApps)
- **miGMM DFG funded project**
  - Modular Incremental Global Model Management
  - Megamodeling

Research Topic: Multidirectional Transformations and Synchronizations



You are here: Program » Seminar Calendar » Seminar Homepage

https://www.dagstuhl.de/18491

December 2 – 7 , 2018, Dagstuhl Seminar 18491

Multidirectional Transformations and Synchronisations

Organizers

Anthony Cleve (University of Namur, BE)
Ekkart Kindler (Technical University of Denmark – Lyngby, DK)
Perdita Stevens (University of Edinburgh, GB)
Vadim Zaytsev (RainCode – Brussels, BE)

Book exhibition

Books from the participants of the current Seminar

Book exhibition in the library, ground floor, during the seminar week.

Documentation

In the series Dagstuhl Reports each Dagstuhl Seminar and Dagstuhl Perspectives Workshop is documented. The seminar organizers, in cooperation with the collector, prepare a report that includes contributions from the