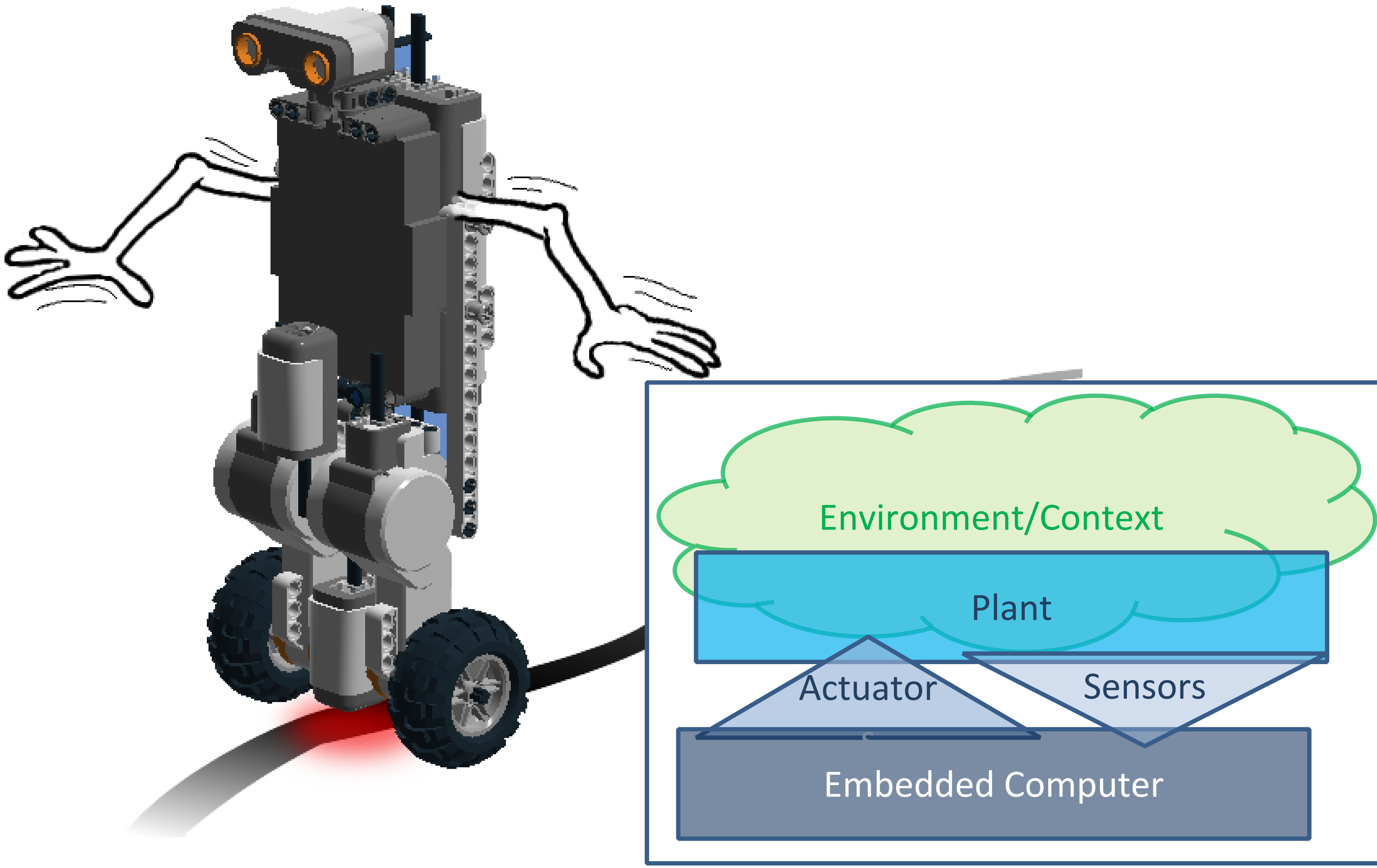
Analysis, Design and Implementation of NXT Embedded Software in Java - einer Leitfaden

Anders P. Ravn
Aalborg University

BEST Summer school
July 2013

Acknowledgement: Jan Sørensen , Nikolaj Rahbek, Arne Skou, Peter A. Nielsen, Stephan Korsholm, René R.Hansen, Hans Søndergaard, and Bent Thomsen.

Robot – CP-system



Object Oriented Analysis and Design

1. **Identfy the Problem Domain (Plant in Context)**
   - rich picture
   - system definition
   - plant model and identification
2. **Identify the Application Domain (Functionality/Control)**
   - functions (use cases)
   - temporal constraints
   - interfaces to the plant (actuators and sensors)
3. **Design**
   - architecture
     - detailed design of algorithms and data structures
4. **Implement and Test**
   - unit test
   - experiments

MDD, SysML, …

Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen and Jan Stage, *Object-oriented Analysis and Design*, MARKO Publishing, Aalborg 2000.

Problem Domain Analusis

1. Rich picture
2. System definition

   F *Functionality for end use*

   A *Application Domain of end use*

   C *Conditions for success*

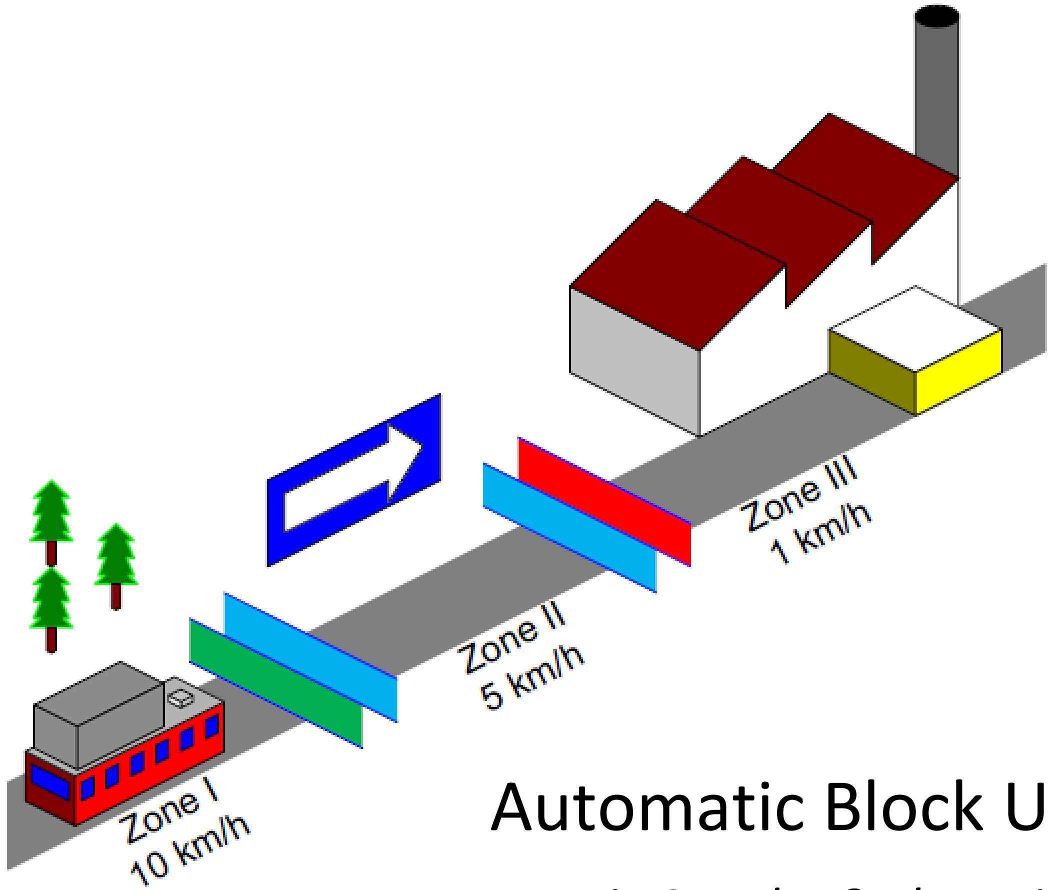
   T *Technology to be used*

   O *Objects in System*

   R *Realization conditions*

3. Plant model and identification
   – Class Diagrams (things and their relations)
   – Selected Behaviours (events and their sequence)

Rich Picture - example



Automatic Block Unloader

Martin Stender & Flemming Hald

FACTOR

Functionality for end use

- Starting and stopping the unloading process
- Detection of speed limit zones
- Detection of the end of the rails
- Controlling the main motor
- Unloading the wagon
- Returning the wagon

Application domain for end use This application is to be automatically operated.

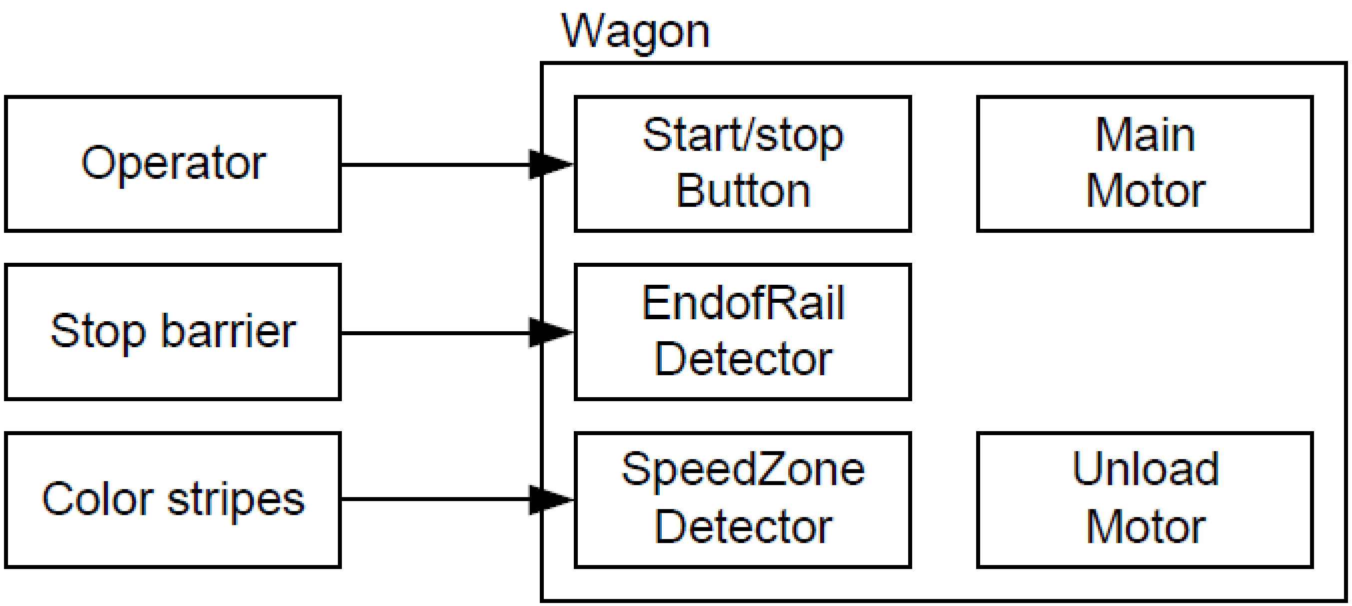
Conditions for success The solution shall ensure that a wagon can be fully automatically unloaded and returned to the starting position.

Technology to be used The prototype is realized using the hardware platform of Lego Mindstorms NXT. The Lejos operating system for Java is used to run the target software on the selected platform. The motors …

Object solution The objects of the solution are shown below

Realization conditions A full working prototype will be realized using the Lego Mindstorms NXT but scaled down to match the size of the prototype wagon compared to the real large size wagon. The solution will as a starting point expect to be in the speed zone with the maximum allowed speed.

Objects



Wagon
Operator
Start/stop Button
Main Motor
Stop barrier
EndofRail Detector
Color stripes
SpeedZone Detector
Unload Motor

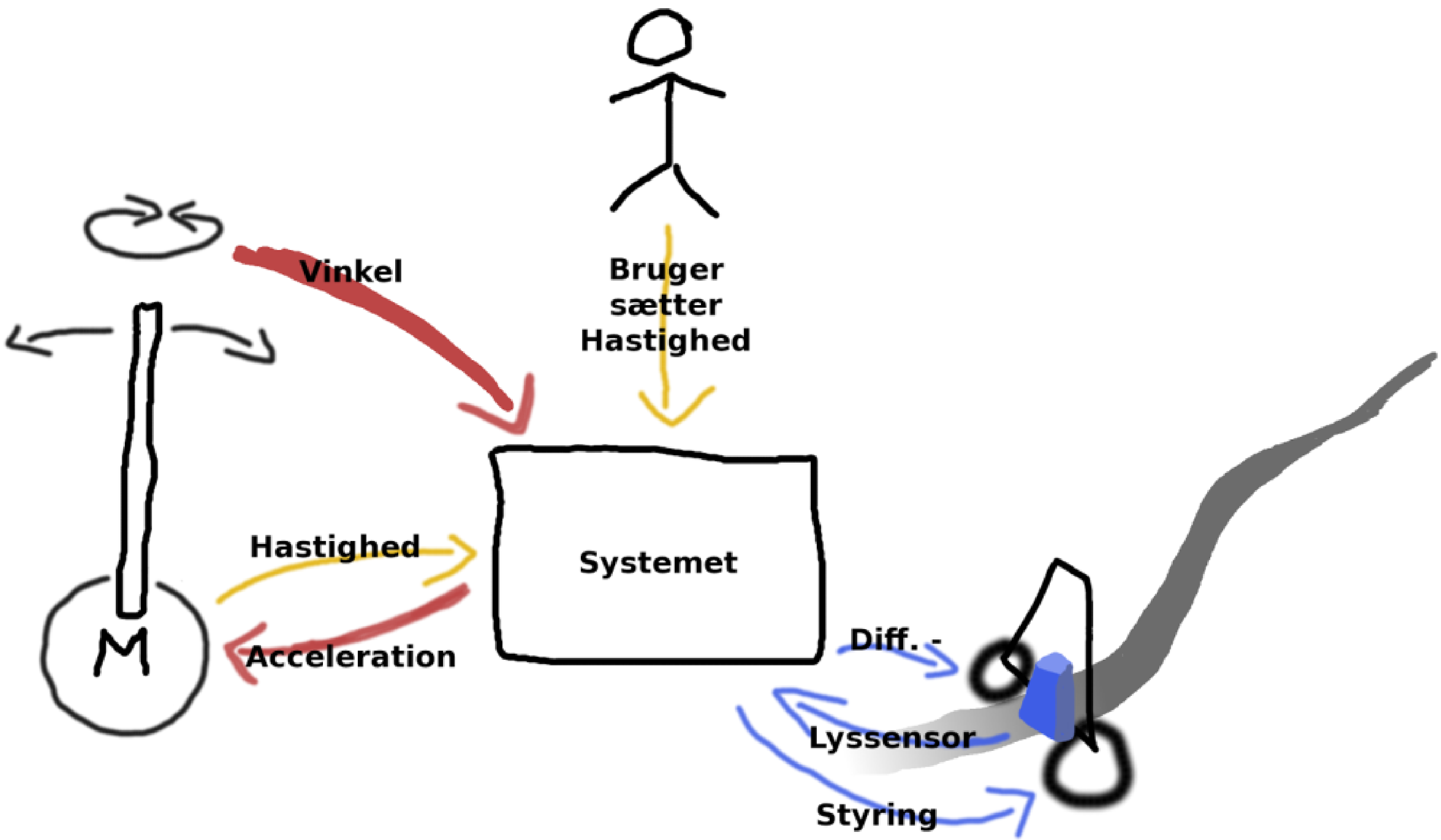
Behaviour

	Speed zone I	Speed zone II	Speed zone III
Maximum speed	25 cm/s	10 cm/s	5 cm/s
Color	Green	Blue	Red

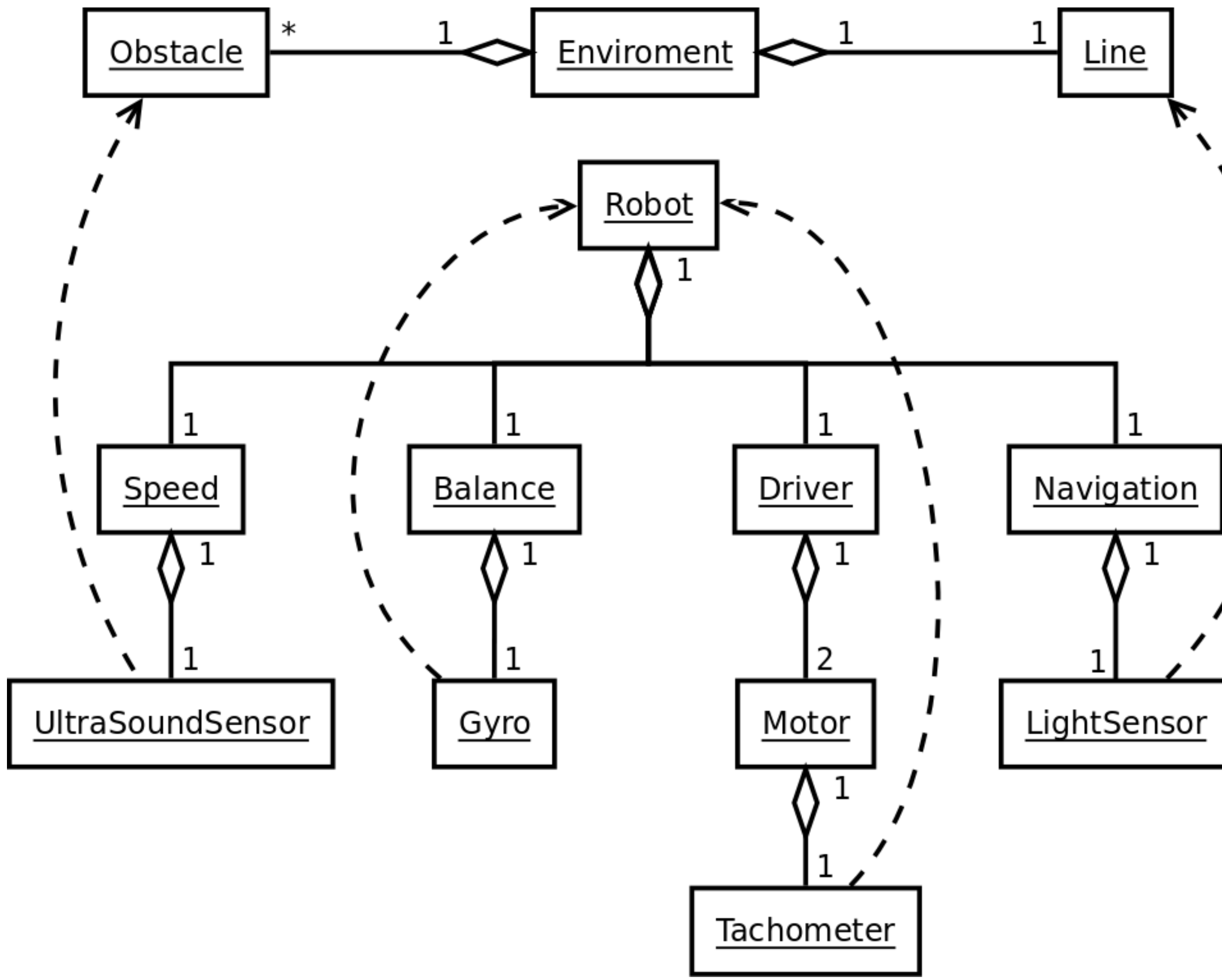
The colored stripes all have a width of 2 cm and the length of each speed zone is 50 cm.

In this downscaled version the wagon will have to fully stop within 5 to 10 cm from the end of the rails.

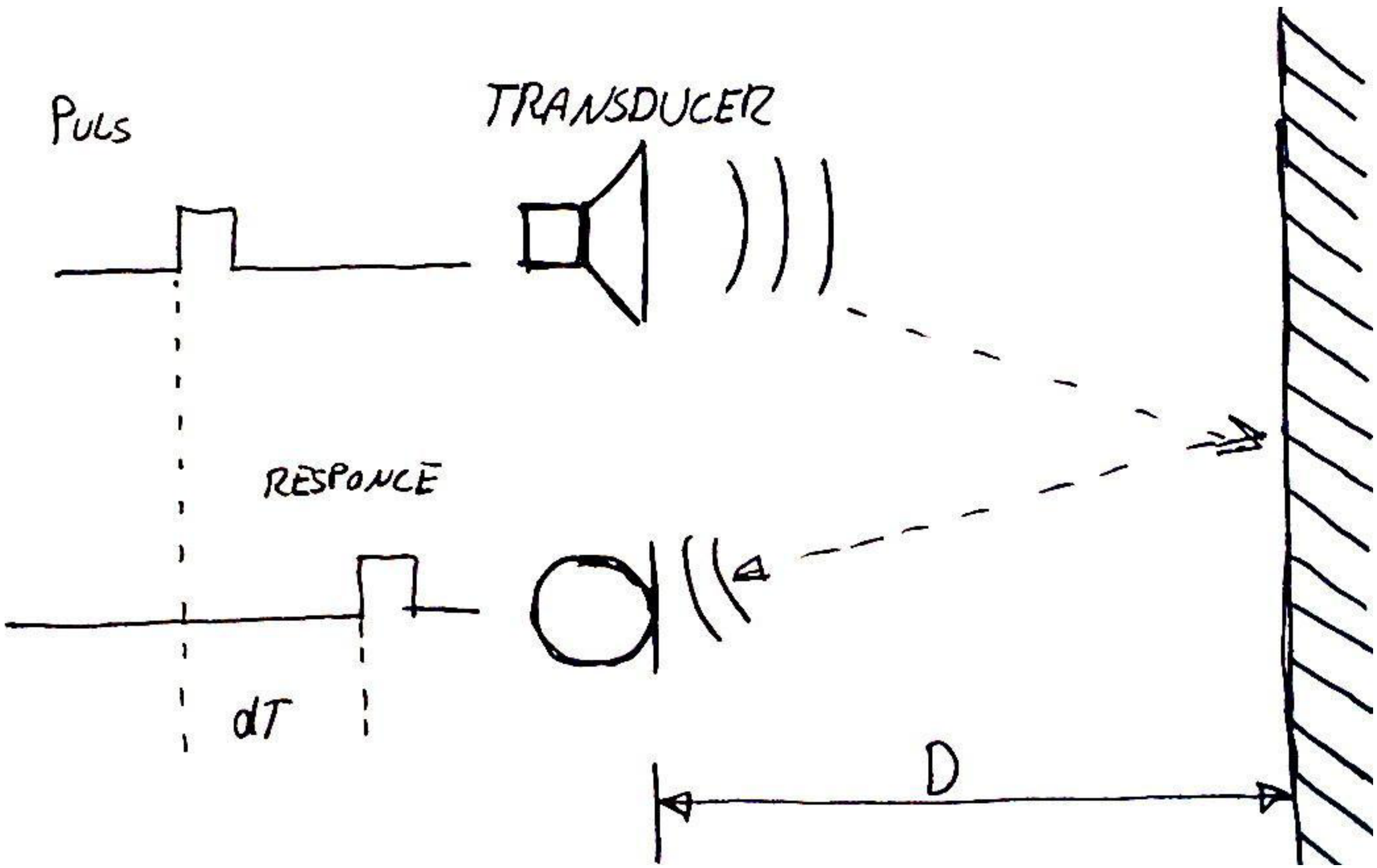
Another Project – Rich Picture



Objects


Obstacle
*
1
Enviroment
1
1
Line
Robot
1
1
Speed
1
Balance
1
Driver
1
Navigation
1
1
1
1
1
1
2
1
UltraSoundSensor
Gyro
Motor
LightSensor
1
1
Tachometer

Behaviours

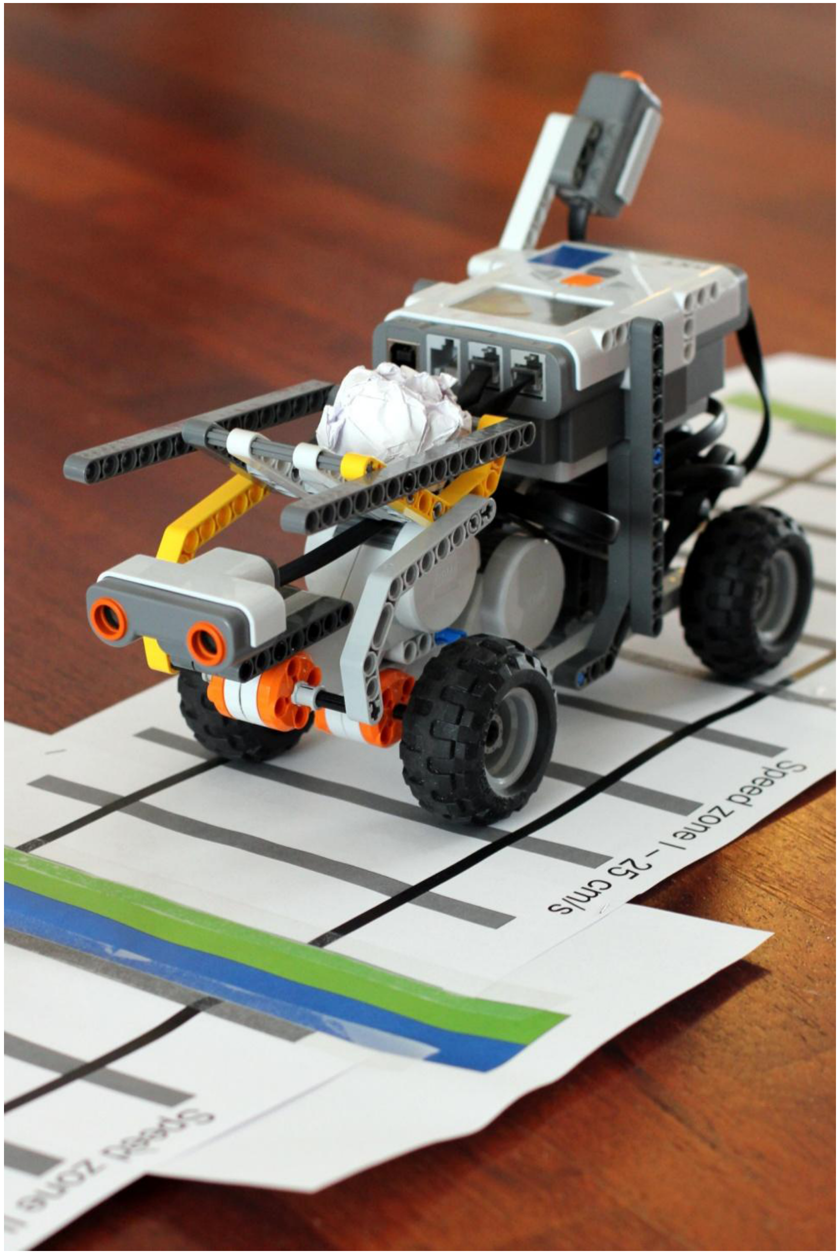


Application Domain Analysis **(Functionality/Control)**

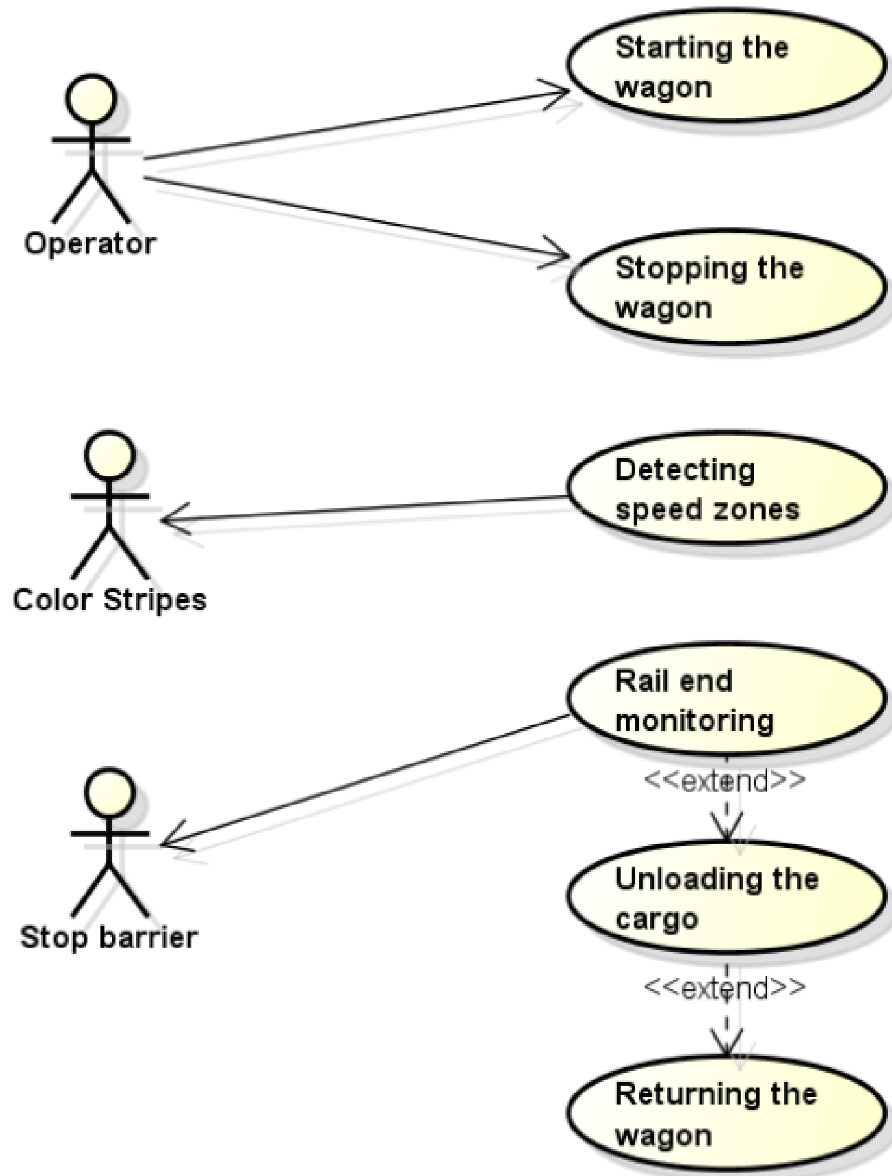
- functions (use cases)
- temporal constraints
- interfaces to the plant (actuators and sensors)

Mechanical Construction

This is not treated here!

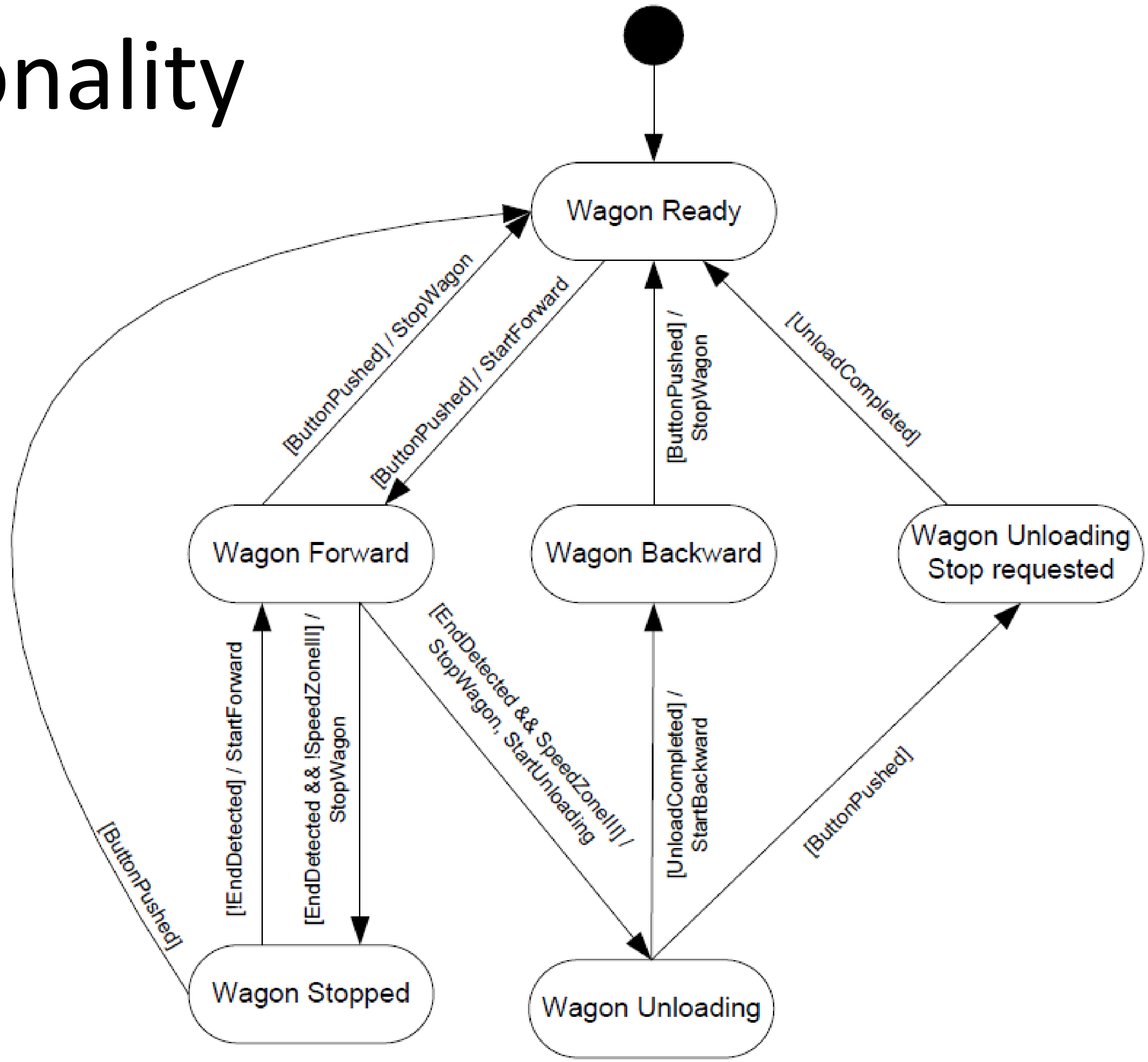


Use Cases



Operator
Starting the wagon
Stopping the wagon
Color Stripes
Detecting speed zones
Stop barrier
Rail end monitoring
<<extend>>
Unloading the cargo
<<extend>>
Returning the wagon

Functionality



Temporal Constraints

Process:	Kind	T [ms]	C [ms]	D [ms]
a) Detection of speed zone	Periodic	40	15	40
b) Detection of end of rails	Periodic	475	20	475
c) Setting the main motor speed	Sporadic	2000	5	50
d) Unloading the wagon	Sporadic	Inf.	5.000	10.000
e) Pushing the button	Periodic	100	2	100

Analysing R-T properties for FPP

B	Worst-case blocking time for the process
C	Worst-case computation time (WCET)
D	Deadline of the process
I	The interference time of the process
P	Priority assigned to the process
R	Worst-case response time of the process
T	Minimum time between releases(process period)

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Response Time Analysis

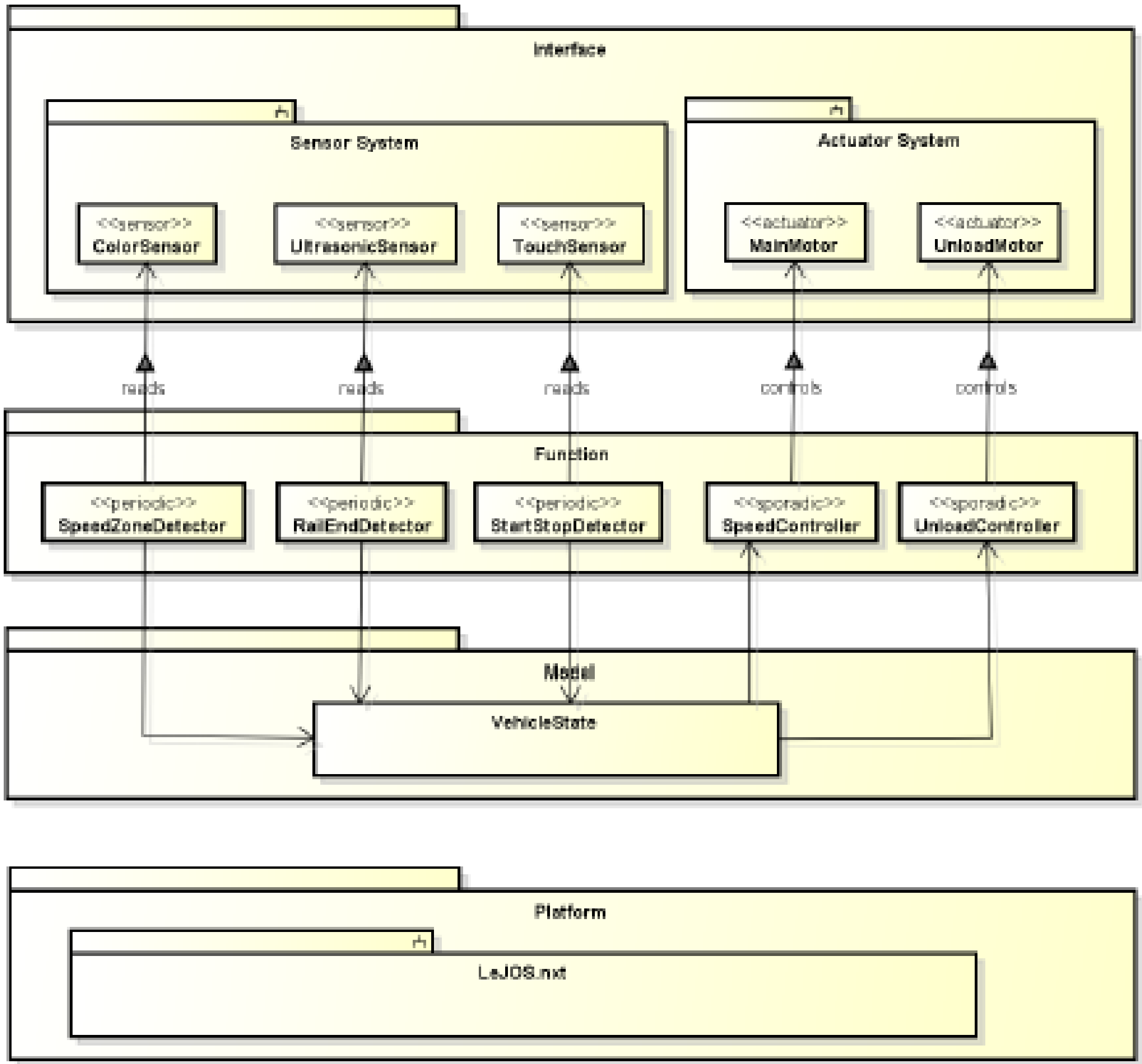
Process:	Kind	P	T [ms]	C [ms]	D [ms]	R [ms]
a) Detection of speed zone	Periodic	5	40	15	40	16
b) Detection of end of rails	Periodic	2	475	20	475	58
c) Setting the main motor speed	Sporadic	4	2000	5	50	21
d) Unloading the wagon	Sporadic	1	Inf.	5.000	10.000	8.945
e) Pushing the button	Periodic	3	100	2	100	23

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Design

- Architecture
- Detailed design of algorithms
  and data structures

Layered Architecture



Implementation and Test

- **SCJ classes**
- **Test**

  - unit test

  - WCET (C) evaluation

  - experiments

SCJ: The Application

```
public class Example implements Safelet {

  // here declaration of   application specific interface objects

  private MyMission mission = new MyMission();

  /* A  MisionSequencer that returns  the singleton <code>mission</code>.  */
  private class SimpleSequencer extends MissionSequencer {
    @Override
    protected Mission getNextMission() { return mission; }   }

  @Override
  public MissionSequencer getSequencer() {
    return new SimpleSequencer();
  }

 static void main(final String[] args) { Launcher.start(new Example());    }
}
```

SCJ: A Mission

```
public class MyMission extends Mission {
  @Override
   protected void initialize() {
   // here: declaration of shared objects  (model layer)
   //          declaration of Events
   AperiodicEvent a = new APeriodicEvent();

   new MyPeriodic(new PriorityParameters(1),
      new PeriodicParameters(null, new RelativeTime(200,0)),
          120,  a).register();
  new MyAperiodic(new PriorityParameters(5),
      new AperiodicParameters(new RelativeTime(20,0), null),
       a).register();
   }
}
```

SCJ: A Handler

```
public class MyPeriodic extends PeriodicEventHandler {

  private int counter;
  private final int MAX_COUNT;
  private final AperiodicEvent name;

  public MyPeriodic(PriorityParameters priority, PeriodicParameters release,
                    int MAX_COUNT, AperiodicEvent a) {
    super(priority, release, null);
    this.MAX_COUNT = MAX_COUNT;
    name = a;
  }
  @Override
  public handleAsyncEvent() {
   if (counter == MAX_COUNT) counter++; else name.fire(); }
  @Override
  public void cleanup() {
```

WCET evaluation

Process:	Kind	D [ms]	C_{est} [ms]	C_{real} [ms]	R_{est} [ms]	R_{real} [ms]
a) Detection of speed zone	Periodic	40	15	9	16	10
b) Detection of end of rails	Periodic	475	20	16	58	29
c) Setting the main motor speed	Sporadic	50	5	2	21	12
d) Unloading the wagon	Sporadic	10.000	5.000	4.722	8.945	6.865
e) Pushing the button	Periodic	100	2	1	23	13

$$R_i = C_i + B_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Summary of Topics

- Problem Domain Analysis
- Application Domain Analysis
- Design
- Implementation
- Test