Mobile Robot Control 2020: Tutorial Lecture #1

APRIL 29TH 2020

Jordy Senden, Hao Liang Chen, Wouter Kuijpers

Department of Mechanical Engineering, Control Systems Technology, Robotics

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Tutorial Lecture #1

Jordy Senden

1. Problem Statement
2. Systems Design

Hao Liang Chen

3. Information Architecture

Wouter Kuijpers

4. Implementation

Problem Statement

Hospital Challenge

- Doctors have limited time to have human-to-human contact

Hospital Challenge

- Doctors have limited time to have human-to-human contact
- Many patients

TU/e

Hospital Challenge

- Doctors have limited time to have human-to-human contact
- Many patients
- Simple tasks → automation

- Getting medicine
- Moving blood bags or syringe
- Getting food/drinks

URGENCY!

COVID-19 shows importance and shortcomings of healthcare systems

- Nurses and doctors are:
  - Scarce
  - Overworked
  - At risk of contracting virus
  - A risk of spreading virus
- Work is:
  - High load
  - Traumatizing

https://www.thelocal.it/20200406/italian-hospitals-turn-to-robots-to-help-monitor-coronavirus-patients

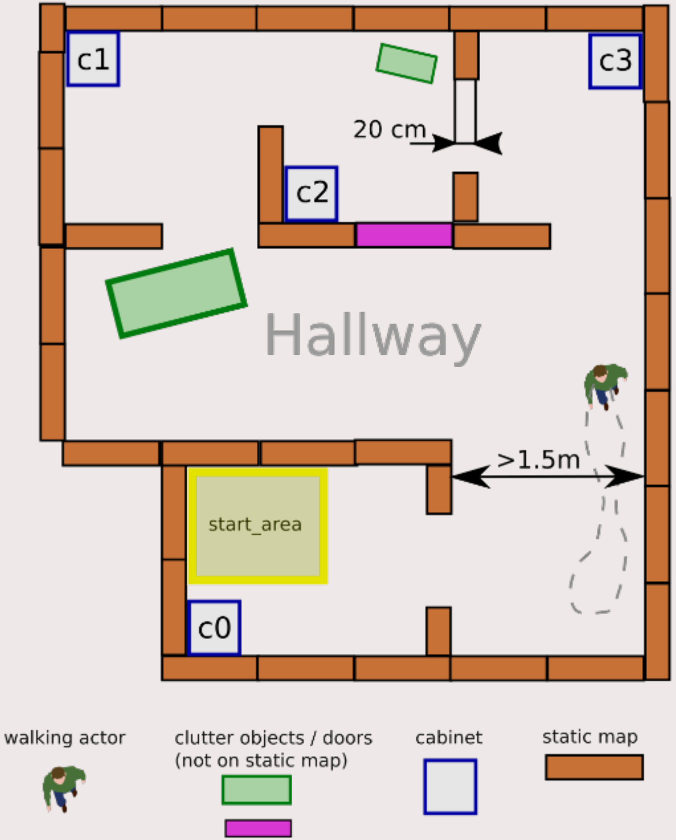
Hospital Challenge

Navigate the hospital environment, while avoiding collisions with walls and static- and dynamic objects.

Final test:
Consecutively visit a given set of cabinets in the hospital.

Example map

Objective: visit cabinets in order c0 -> c1 -> c3



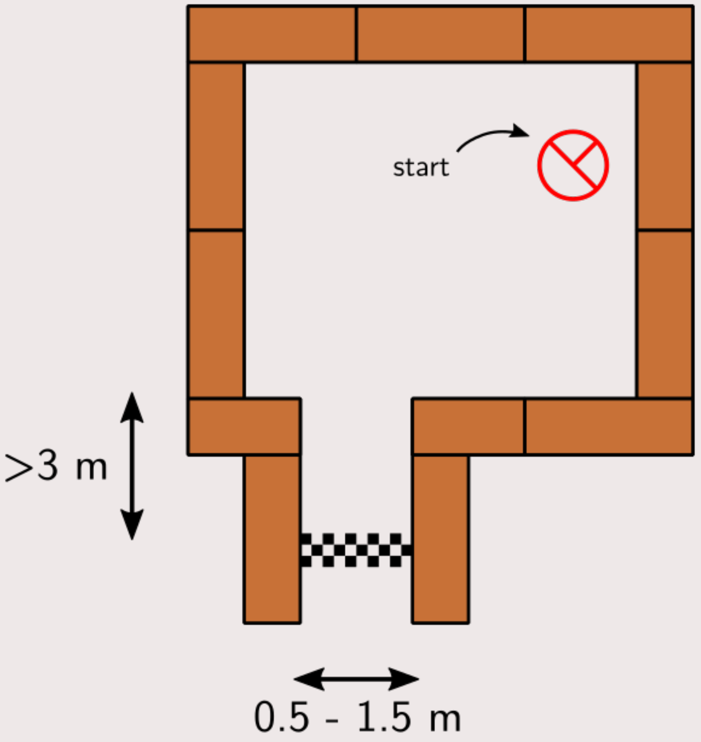
TU/e

Escape Room Challenge

Navigate the robot out of the room.

- Avoid collision
- Exact room layout not specified
- Stop after passing finish

- Check WIKI for more challenge constraints!



TU/e

Disclaimer

This course is about solving a real problem.

This problem does not have a wrong or right solution.

Robotic systems are complex! Creating an overview of the problem and getting a grip on it will be a challenge.

We will give you handles and share ideas, we do not claim that you SHOULD do it like that.

Be creative, be critical, question everything and have fun!

Systems Design

Design of High-Tech Systems

Complex and nontrivial: more art than exact science.

What is the *problem*?
Who is a *stakeholder* for the system?
In what *environment* will the system operate?
What *tasks* should the system be doing?
What are possible *concepts*?

What concept has best prospect at success?
How well does my system work?

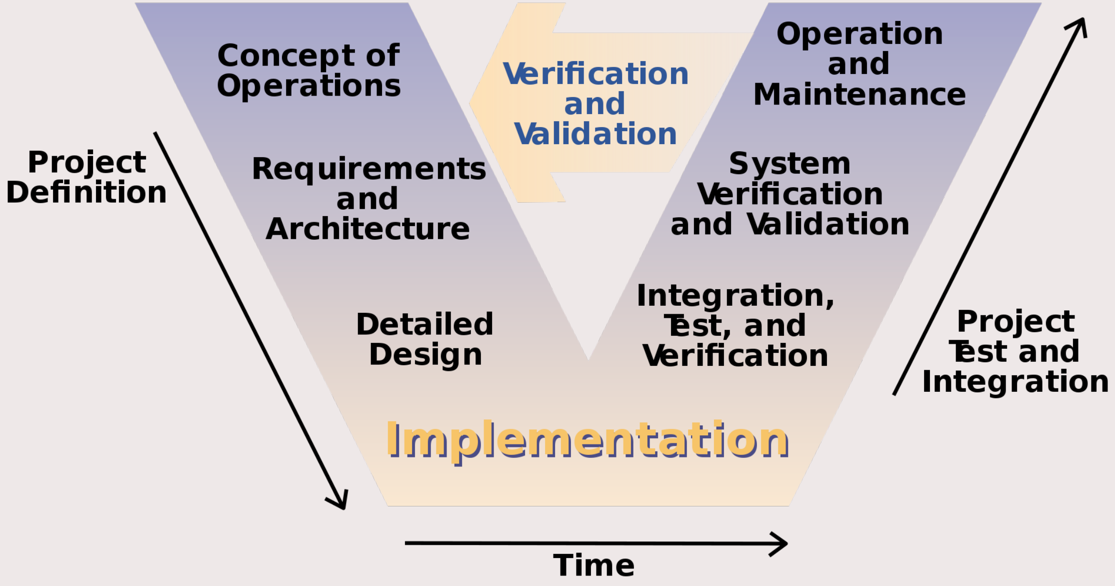
→ Requirements

How can we divide workload and cooperate?

→ Process

V-model

A model for software- and system design process



Requirements

What **should** the system do?

- Speed limits
- Wall clearance
- Driving lanes
- Driving heading
- …

- What if people are in the way?
- What if a door is blocked?
- How long to ‘idle’?
- …

Specifications

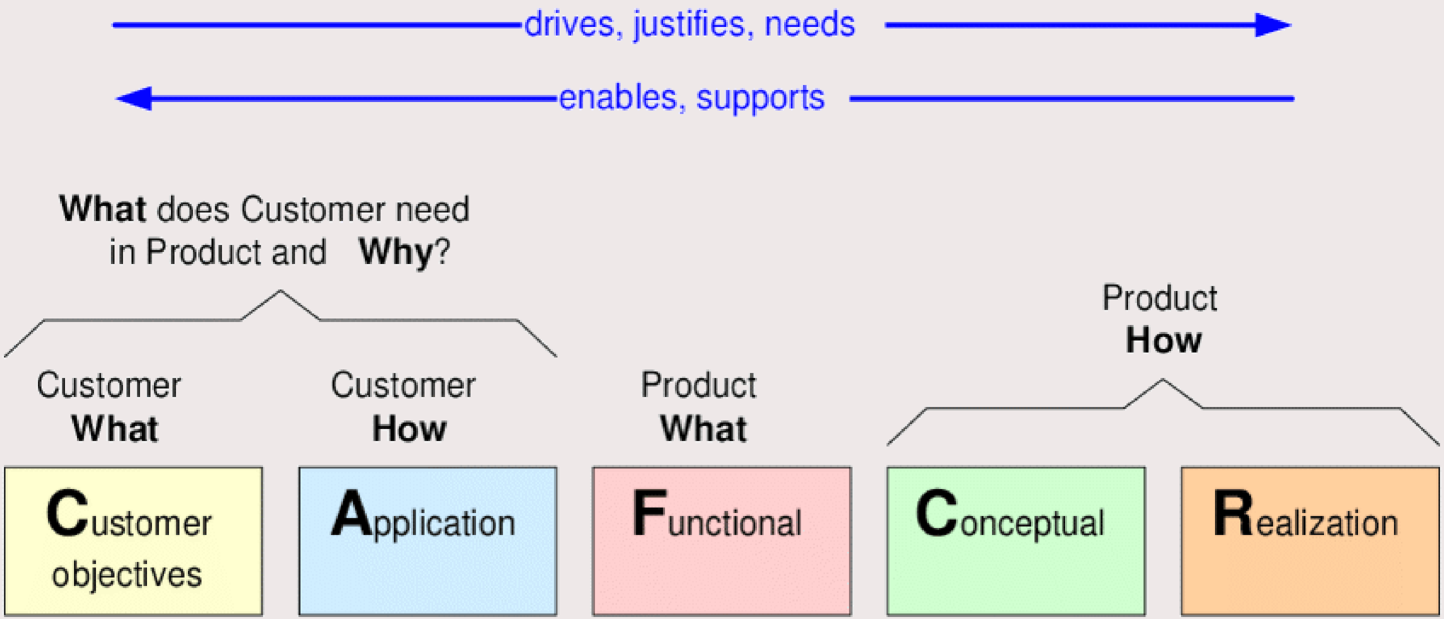
What **can** the system do?

System to use is PICO:

- Omni-wheels: Holonomic base
- Laser Range Finder (LRF)
- Sonar
- Camera
- Screen
- Audio-output

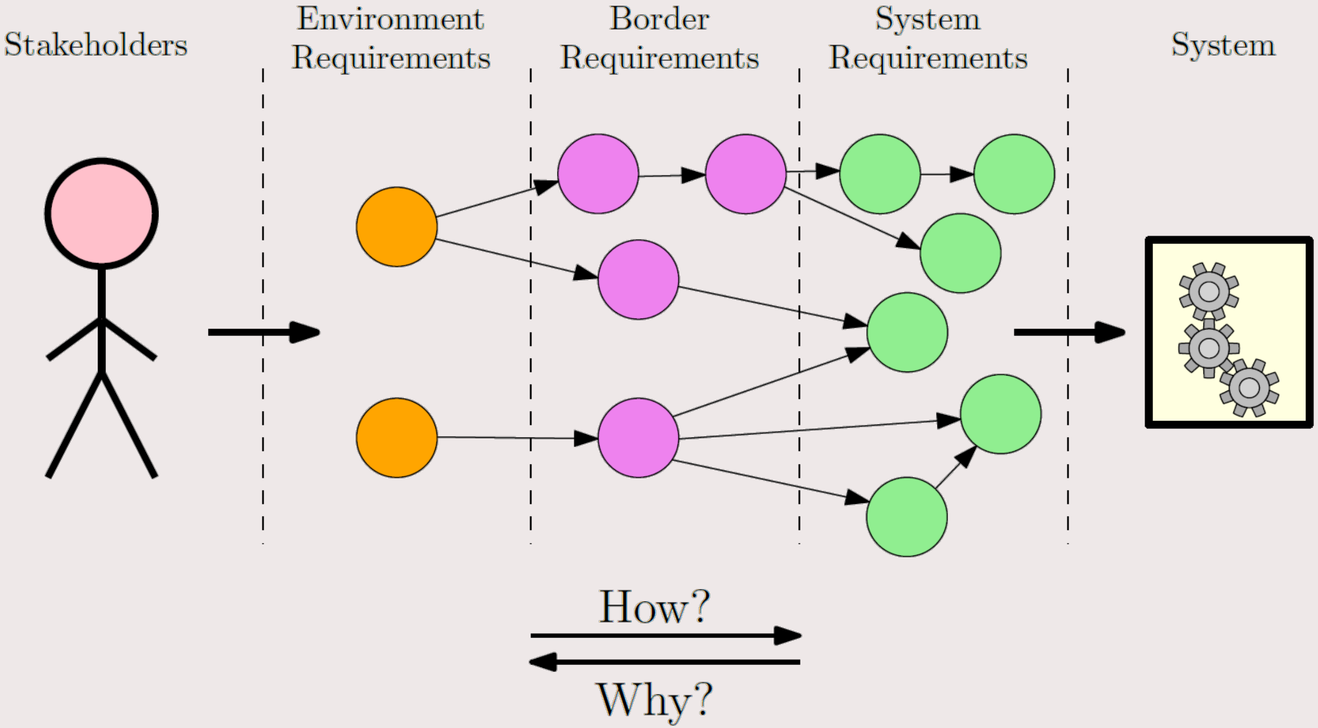
CAFCR

Model to organize different views on the system



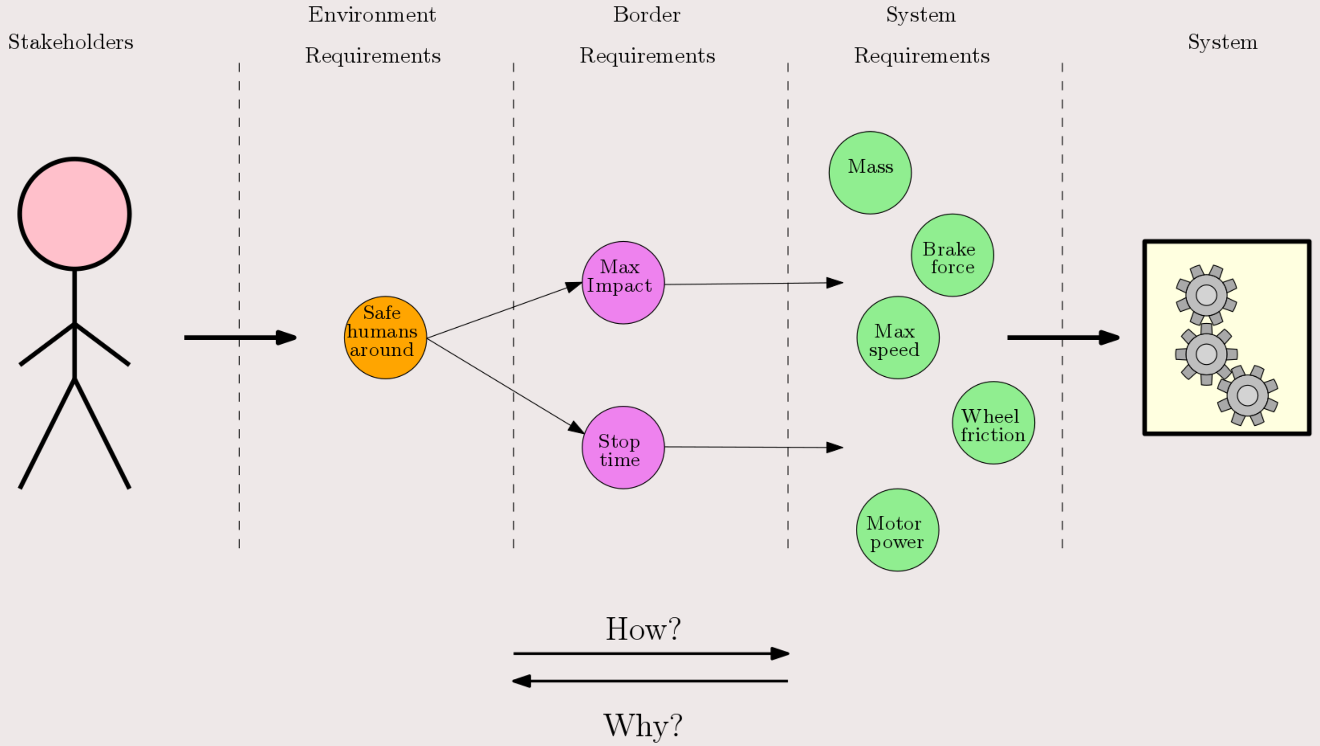
https://www.gaudisite.nl/ThesisBook.pdf

From Desires to Specs



TU/e

From Desires to Specs

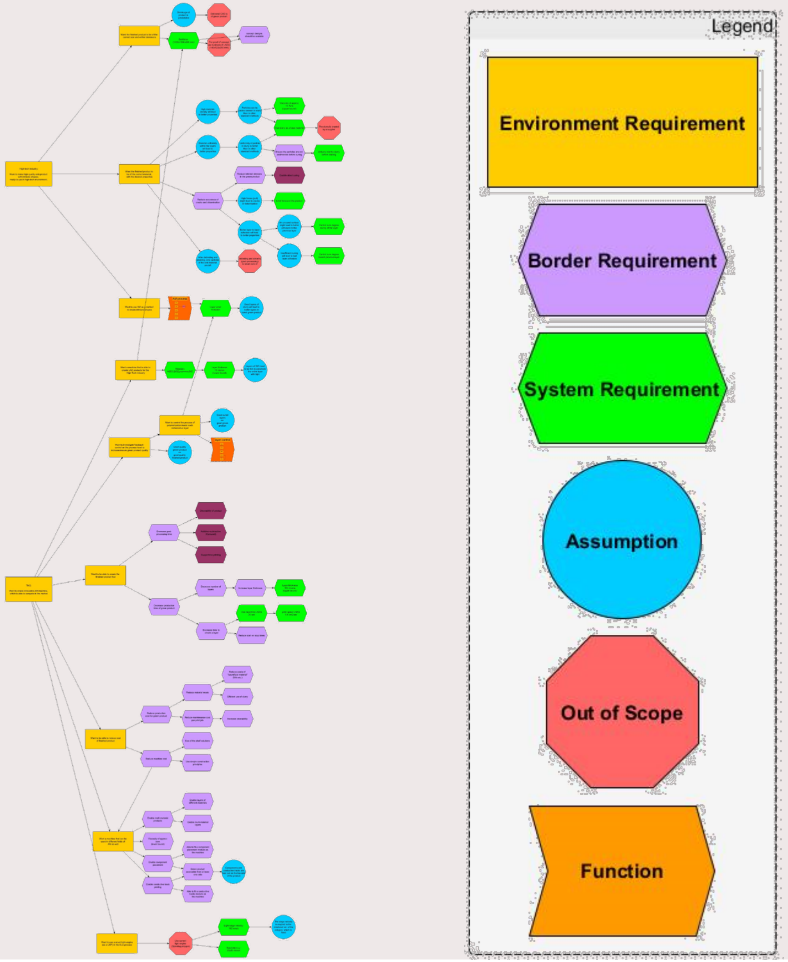


Why is this important?

- Bookkeeping
- Back-traceability
- Insight in conflicts
- Ordering of importance

- Coherence in group
- Discussion points
- Comparing system designs

- NO MORE MAGIC NUMBERS!



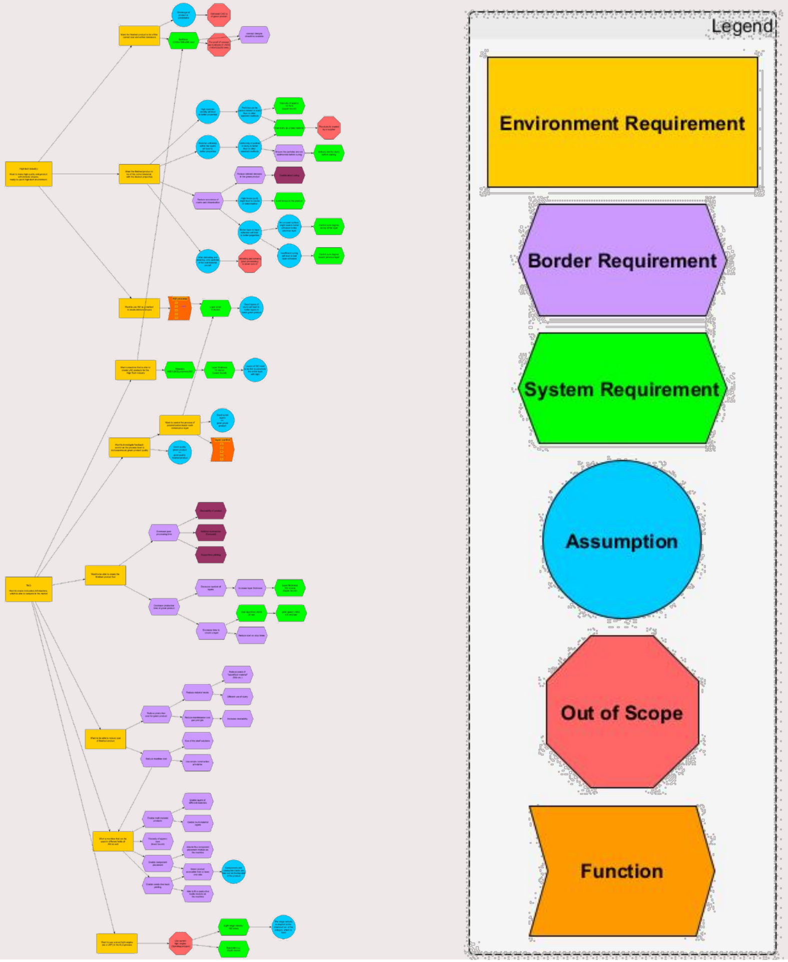
TU/e

Why is this hard?

- Many stakeholders
- Conflicting desires
- Design without system in mind
- No tangible output (at first)

- Boring (no it is not!)

- Not set in stone, fluid
- Not an exact science!
- Use it as a tool

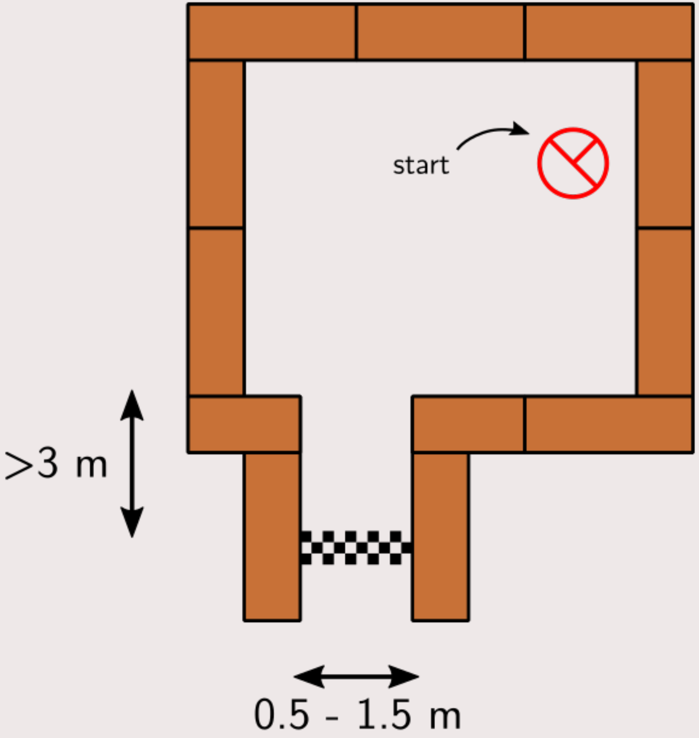


TU/e

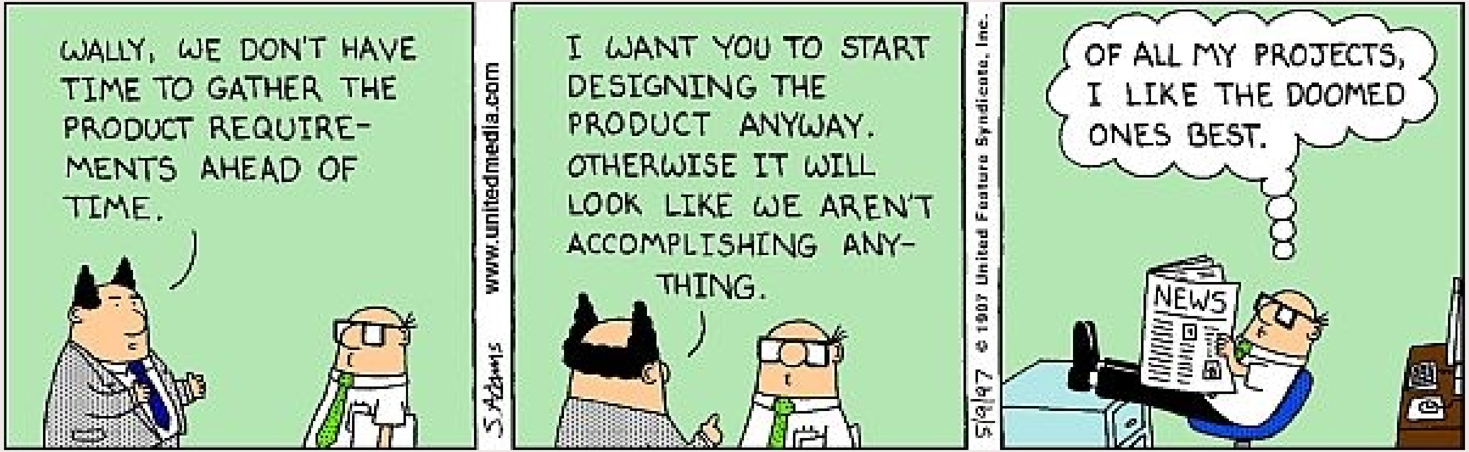
Escape Room Challenge

- Some requirements already on the WIKI

- Think about scalability towards the hospital challenge, this might save time in the future!



Take time to think



Challenge Specification

User Story

- Doctor addresses a patients needs

User Story

- Doctor addresses a patients needs
- Many patients
- Simple tasks → automation

User Story

- Doctor addresses a patients needs
- Many patients
- Simple tasks → automation

- Getting medicine
- Moving blood bags or syringe
- Getting food/drinks

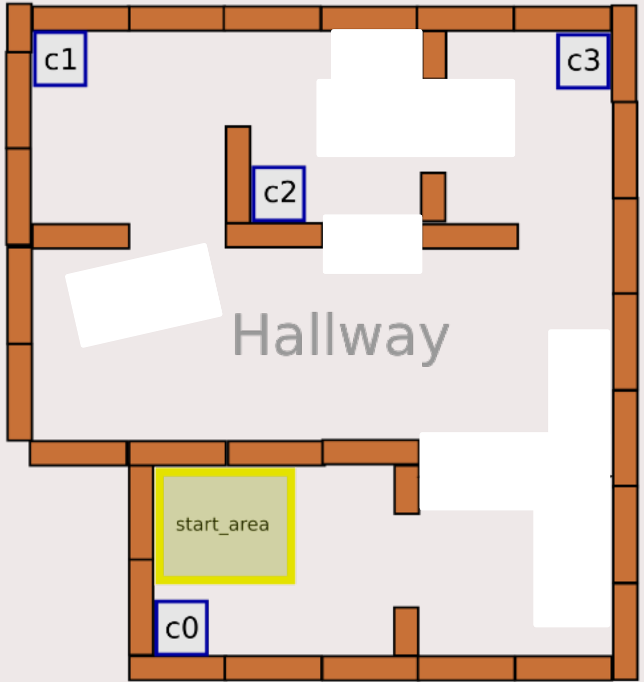
Challenges

- Localizing in the environment
- Know where to find cabinets
- Know order of visitation

- Fit through hall- or doorways?
- All doors open? Reroute
- Unforeseen objects?
- Moving objects?
  - Humans
  - Robots

Example map

Objective: visit cabinets in order c0 -> c1 -> c3



Requirements & Specifications

Requirements

What **should** the system do?

- Speed limits
- Wall clearance
- Driving lanes
- Driving heading
- …

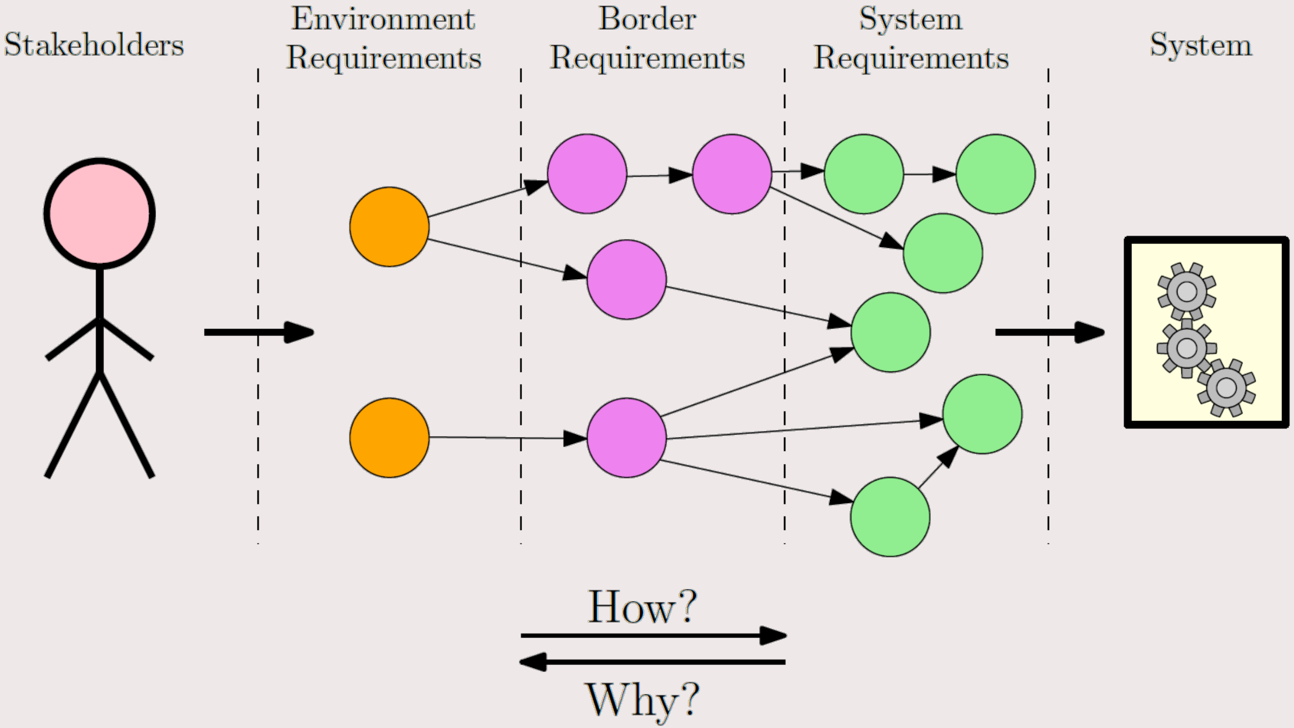
- What if people are in the way?
- What if a door is blocked?
- How long to ‘idle’?
- …

Specifications

What **can** the system do?

System to use is PICO:
- Omni-wheels: Holonomic base
- Laser Range Finder (LRF)
- Sonar
- Camera
- Screen
- Audio-output

From Desires to Specs

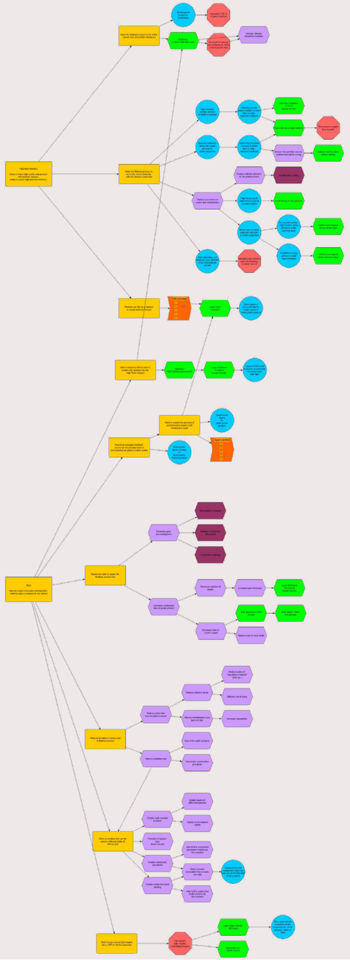


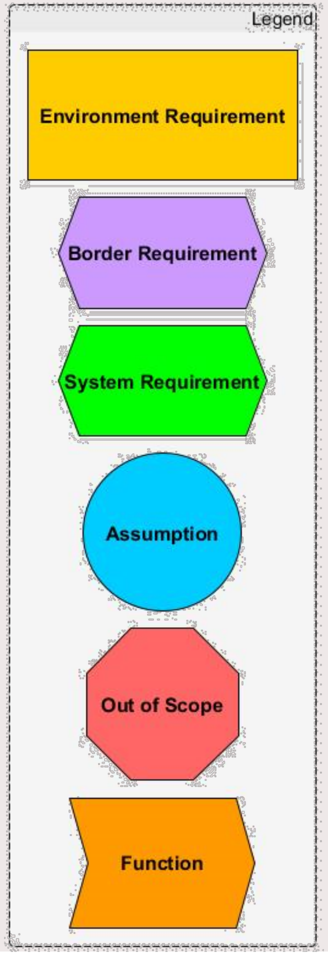
Why is this important?

- Bookkeeping
- Back-traceability
- Insight in conflicts
- Ordering of importance

- Coherence in group
- Discussion points
- Comparing system designs

- NO MORE MAGIC NUMBERS!





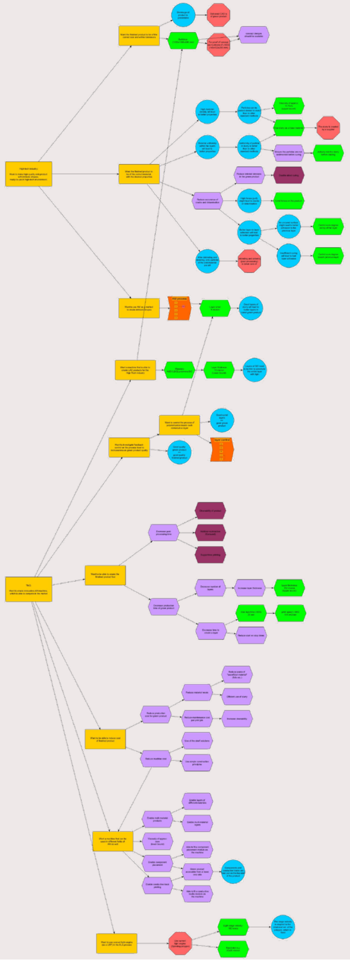
TU/e

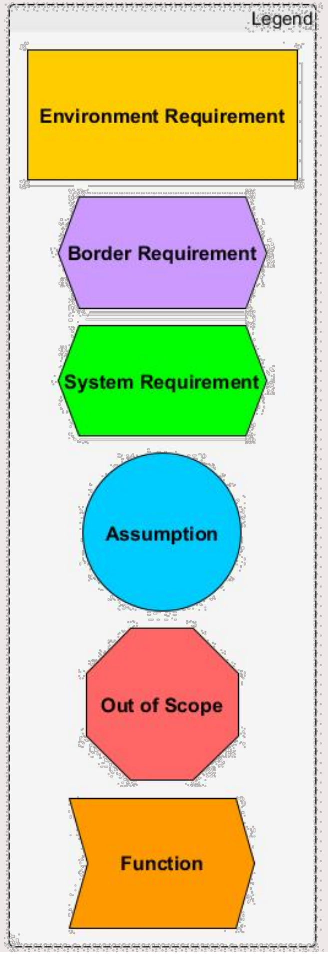
Why is this hard?

- Many stakeholders
- Conflicting desires
- Design without system in mind
- No tangible output (at first)

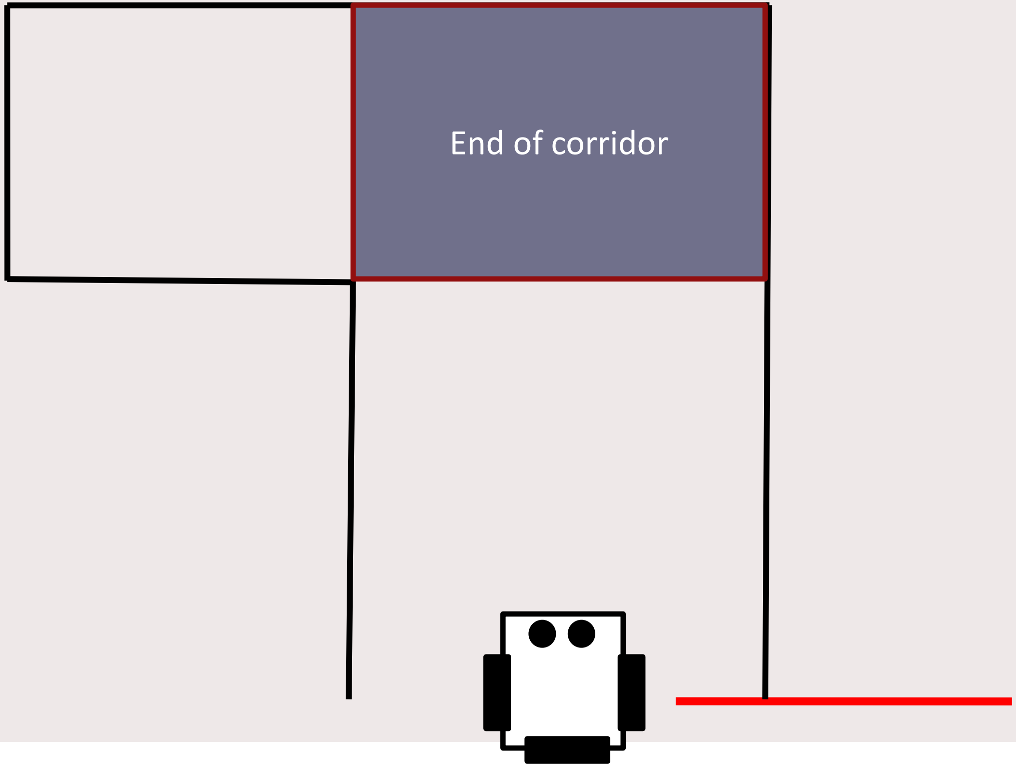
- Boring (no it is not!)

- Not set in stone, fluid
- Not an exact science!
- Use it as a tool





Start ‘simple’- Corridor Challenge



Information Architecture

Use-case dependent

No all-purpose architecture

- ➢Every use case is different
- ➢Flow of information is also different
- ➢Not **one** solution

Differing (perceived) context

- ➢Task, e.g., ‘navigate’ or ‘manipulate’
- ➢Stakeholders derived constraints, e.g., ‘regard humans as obstacles’ or ‘never block humans’
- ➢Environment features, e.g., ‘dynamic’ or ‘static’
- ➢Robot, e.g., ‘drone’ or ‘wheeled robot’

Context determines flow of information and

TU/e

Flow of information (1)

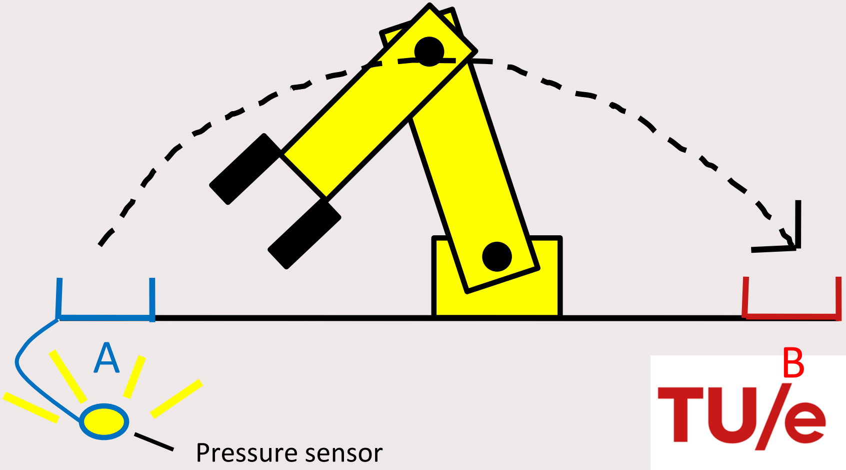
Context:

- ➢Task: move object A → B
- ➢Stakeholders derived constraints: time
- ➢Environment: no disturbances + sensors beneath box A
- ➢Robot: manipulator robot

Information architecture sketch:

1. **Plan time optimal path A → B**
2. **Sense activation pressure sensor A**
3. **Act on designed plan**
4. **Iterate steps 2-3**

Architecture can become 'simple' when considering confined environments



TU/e

Flow of information (2)

Context:

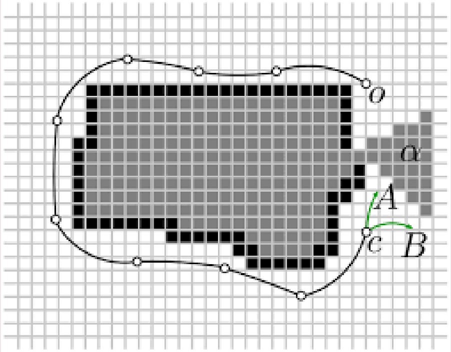
- Task: move to location X to grab something
- Stakeholders derived constraints: speed limits, clearance, etc.
- Environment: dynamic (humans, trolleys, etc.)
- Robot: wheeled robot

Information architecture sketch:

1. **Plan time optimal path to location X**
2. **Sense command to move**
3. **Act on designed plan**
4. **Iterate steps 2-3**

Problems:

1. No real-time perception for disturbances (humans)
2. Localization is typically not perfect, e.g., due to drift



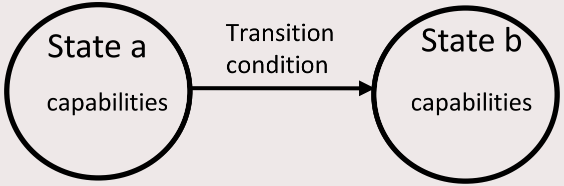
We can't re-use our previous architecture due to differing context

TU/e

How to design behavior?

1. Context.
   - ➢Task: when is it completed?
   - ➢Stakeholders derived constraints: what are the constraints that are imposed on the system?
   - ➢Robot: what are the hardware capabilities of the robot (and are they ‘perfect’)?
   - ➢Environment features: - how can the robot interact/perceive the environment?
     - what is the dynamics in the environment?

1. Draft a rough robot behavioral scheme with the ‘context’ in mind
   - ➢Design a Finite State Machine (FSM)
   - ➢No formal guideline of what should be in a ‘state’
   - ➢Every state has a certain ‘sketch’ of the environment
   - ➢Hypothesize about possible failures of behavior and iterate

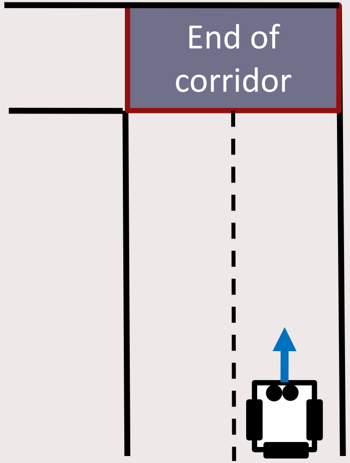


1. Design software that accomplishes the behavior
   - ➢ Code (or think about) the exact information exchange and algorithms
   - ➢ Review the information exchange and possibly iterate

1. Review the process for (hidden) assumptions and iterate process

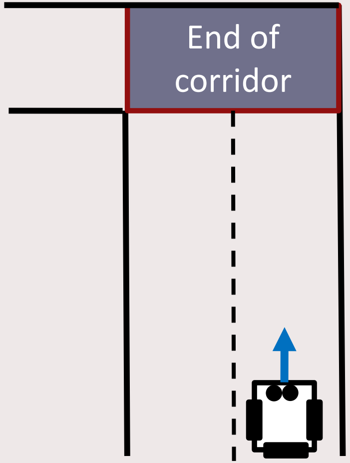
Context deduction

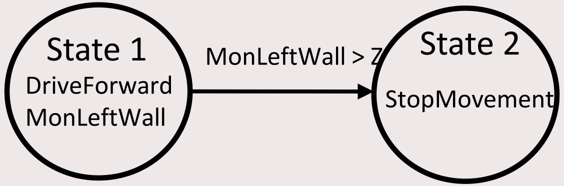
- **task:**
  1. Reach end of corridor
- **Stakeholders derived requirements**
  1. Stay on right side of the lane
- **Robot:**
  1. Omniwheels → motion
  2. 2D Laser Range Finder (LRF) → sense light-reflecting objects
  3. Odometry → sense wheel rotations
- **environment:**
  1. No dynamic objects in the corridor
  2. Corridor is split into two equal width virtual lanes
  3. Straight stone walls on both sides



Behavior draft (1)

- **Hypothesis:**
  - Situation is exactly as depicted on picture
- **Task completion (reach corridor)**
  - Sensing capabilities: 2D LRF reading, odometry
  - completion condition: left wall distance suddenly becomes bigger
- **Stakeholders derived requirements (stay in right lane)**
  - Sensing capabilities: 2D LRF reading, odometry
  - Condition completion: already fulfilled
- states
  1. Drive forward and monitor left wall distance
  - ❑ Environment sketch: static environment + parallel walls
  2. Stop movement
  - ❑ Environment sketch: static environment + no parallel walls



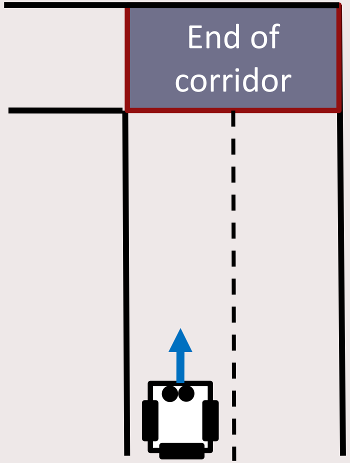


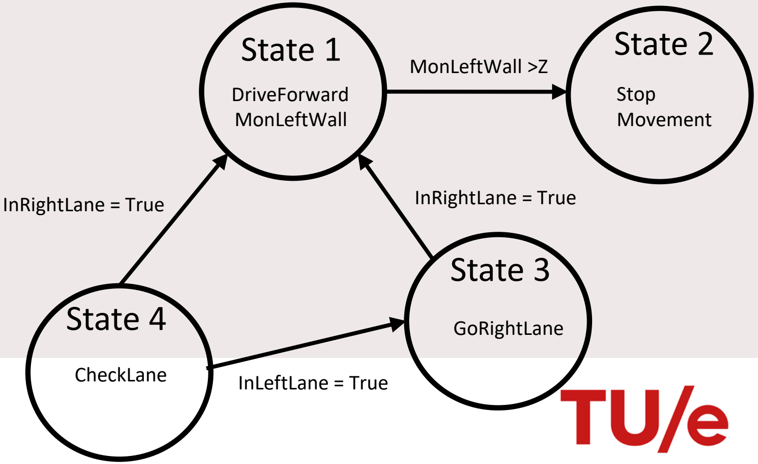
Review hypothesis and make new draft!

TU/e

Behavior draft (2)

- **Hypothesis:**
  - ➢ The robot can start in either lanes with heading parallel to wall
- **Task completion (reach corridor)**
  - ➢ Sensing capabilities: 2D LRF reading, odometry
  - ➢ completion condition: left wall distance suddenly becomes bigger
- **Stakeholders derived requirement (stay in right lane)**
  - ➢ Sensing capabilities: 2D LRF reading, odometry
  - ➢ Condition completion: robot checks its position w.r.t. the wall (and does a recovery action)
- **States**
  1. Drive forward and monitor left wall distance
  - ❑ Environment sketch: static environment + parallel walls + right side of lane
  2. Stop movement
  - ❑ Environment sketch: static environment + no parallel walls + right side of lane
  3. Move to right lane
  - ❑ Environment sketch: static environment + parallel walls + left side of lane
  4. Check lane position
  - ❑ Environment sketch: static environment + parallel walls + unknown lane





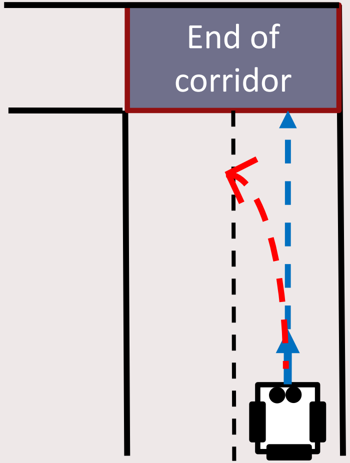
But what about drift?

TU/e

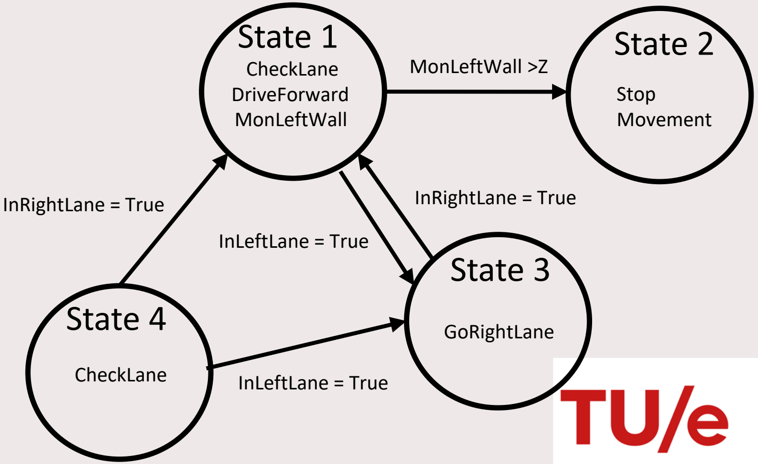
Behavior draft (3)

- **Hypothesis:**
  - The robot can start in either lanes with heading parallel to wall
  - The robot experiences drift for long distances

- **Possible solution:**
  - Checklane() functionality also present in state '1'
  - Subsequently we don't need state 4 anymore



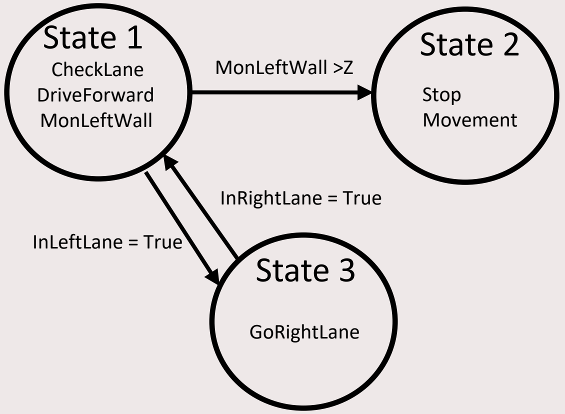
There is no <u>one</u> design of FSM as it depends On everything the designer thinks is relevant



TU/e

Capability design

Behavior scheme



influences →

Capabilities

- DriveForward()
- MonLeftWall()
- CheckLane()

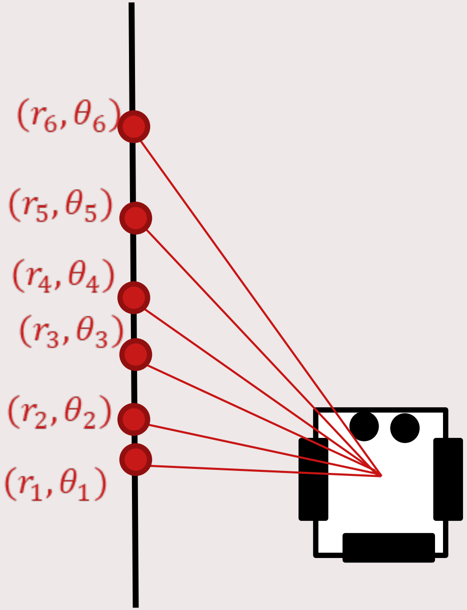
← influences

Robot hardware

- movement
- Laser Range Finder
- Odometry

Example capability MonLeftWall()

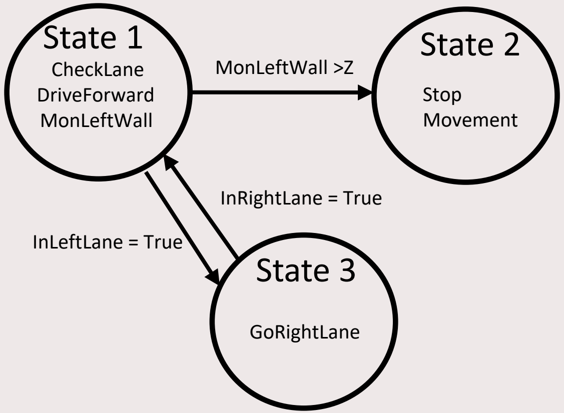
- Wall can be perceived laser range finder
- In the software, we will get the points of reflection back in polar coordinates
- Options 'MonLeftWall()'
  1. Left wall distance = r_1
     - hypothesis: robot heading direction is always parallel to wall
  2. Try to fit a line through the datapoints
     - hypothesis: the robot's heading is not always parallel to wall



- Function layout:
  1. Retrieve (the r_1 component of) LRF data
  2. Do processing on that data
  3. Give desired output back

Capability design

Behavior scheme



influences →

Capabilities

- DriveForward()
- MonLeftWall()
- CheckLane()

- World model

← influences

Robot hardware

- movement
- Laser Range Finder
- Odometry

Capabilities are not necessarily algorithms, it is everything that the robot has as a resource, e.g., a storage of knowledge.

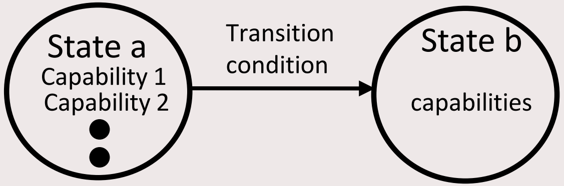
TU/e

Information exchange

- Information exchange also stored in Finite State Machine
  - State mentions sequencing of capabilities
  - State mentions frequency of capabilities

- Information transport
  - Store into separate containers
    - The coding may become quite complex when lots of 'dependencies' are present

  - World model (struct)
    - General place where all 'world' knowledge is stored → multiple algorithms may use same piece of information
    - Certain capabilities/algorithms are only allowed to 'update' certain properties
    - Something needs to coordinate this; the FSM itself or properties in the world model struct
    - Capabilities can do their own checking with all knowledge present in the 'world'. It is thus not sequencing dependent, it can check on its own whether it is allowed to do something.




```
A = capability1()
B = capability2(A)
C = capability3(B)
Etc.
```

```
Struct world_model {
    a = …
    b = …
    c = … } world;

World.a = capability1()
World.b = capability2(world)
World.c = capability3(world)
```

World model map information

- An important part of the world model is 'environment' knowledge
  - ➢ How do abstractize the world enough for our software?
- Resolution of map depends on the designer:
  - ➢ **The (hidden) assumptions behind designed FSM (and capabilities):**
    1. Corridor-like environment → parallel walls
    2. 2D LRF and movement → walls simplify to 'lines'
    3. Robot always start with heading parallel to walls

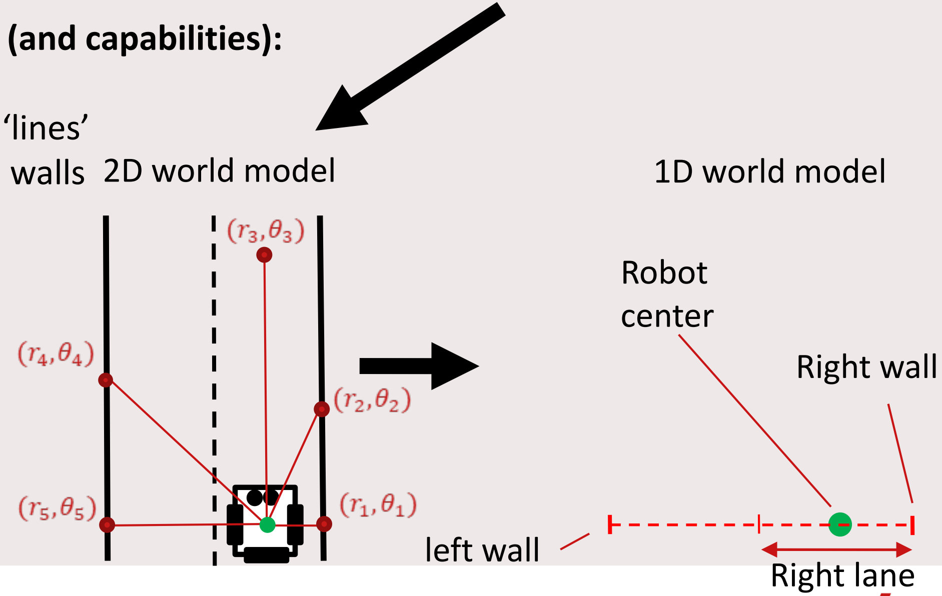
Real world

```
Struct 2d_world_model {
    vertical_line_1 = {(x1,y1),(x2,y2)}
    vertical_line_2= {(x3,y3), (x4,y4)}
} 2d_world;

Struct 1d_world_model {
wall_distance = Z
Lane_division_point = Z/2
} 1d_world;
```

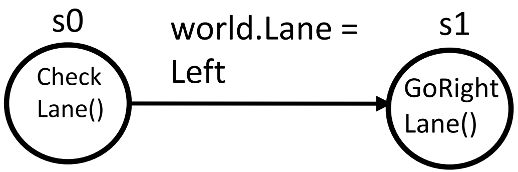
2D world model

1D world model



Interaction example

State machine

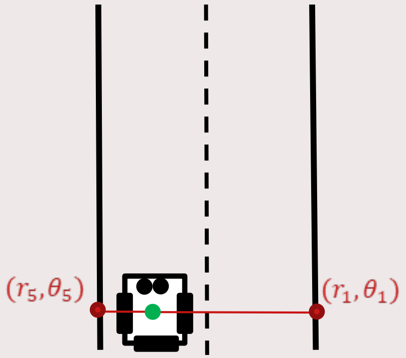


Function: CheckLane(world)

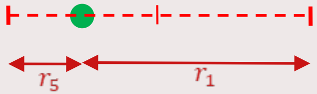
```
If world.initialization == false:
    break

X = world.wall_distance
If r_5 > X/2 AND r_1 < X/2:
    world.Lane = Right
Else:
    world.Lane = Left
return
```

2D world model



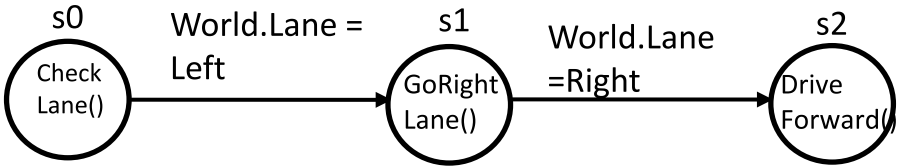
1D world model



TU/e

Interaction example

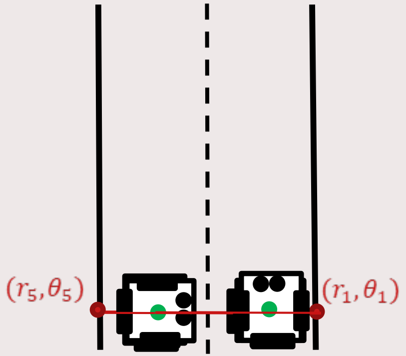
State machine



Function: GoRightLane(world)

```
    If world.Lane == Right:
        break
    RotateRight(90)
    Y = world.WallClearance
    Displacement = r_1 – Y
    MoveForward(Displacement)
    RotateLeft(90)
    World.Lane = Right
```

2D world model



1D world model