Final presentation EMC03

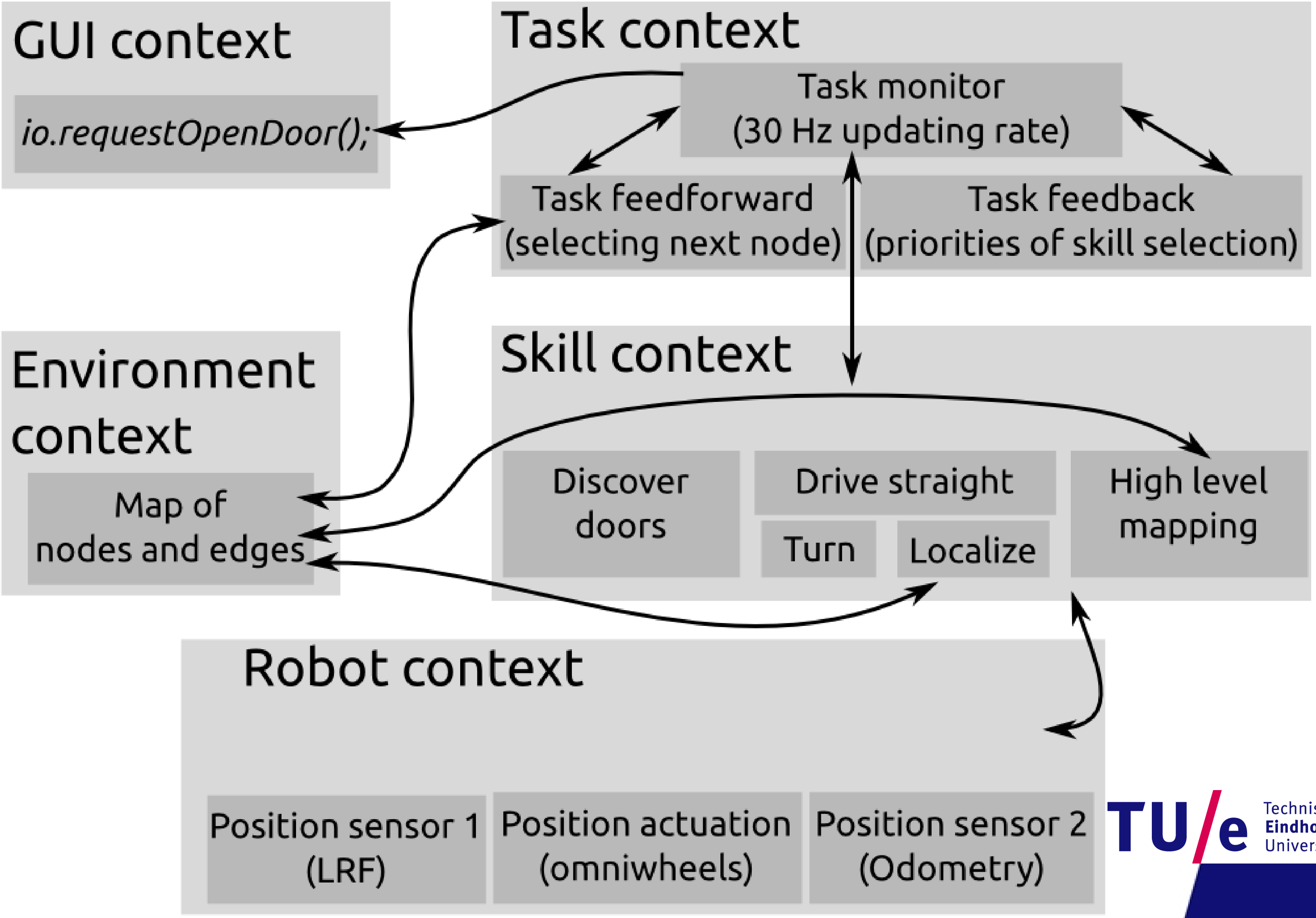
Max van Lith	0767328
Shengling Shi	0925030
Michel Lammers	0824359
Jasper Verhoeven	0780966
Ricardo Shousha	0772504
Sjors Kamps	0793422
Stephan van Nispen	0764290
Luuk Zwaans	0743596
Sander Hermanussen	0774293
Bart van Dongen	0777752

Contents

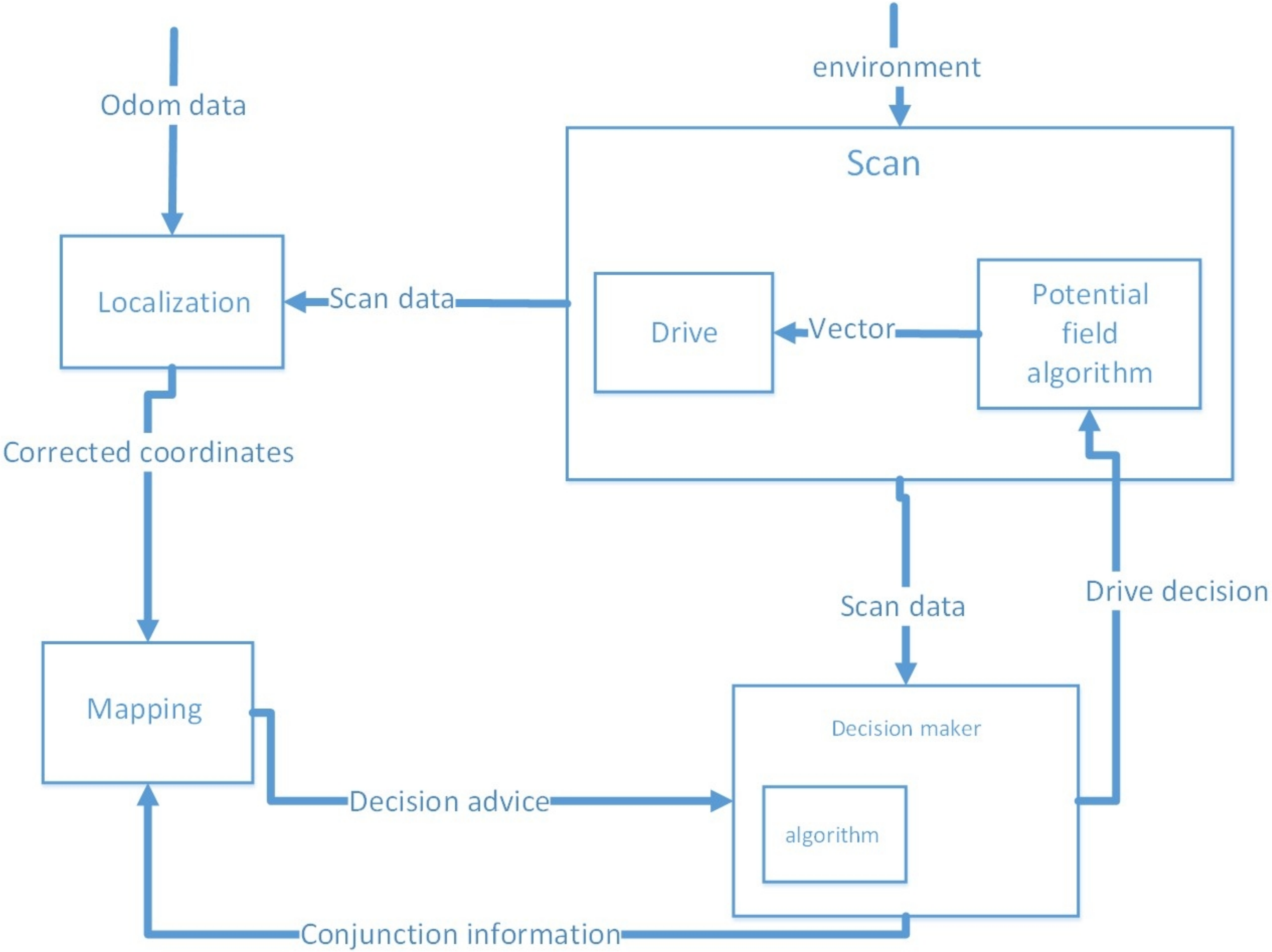
- Behavior Design
- Decision Maker Composition Pattern
- Localization and Scanning Composition Pattern
- Mapping and Solving Composition Pattern
- Software Summary
- Stuff that we expected from EMC
- Stuff that we've learned from EMC

Technische Universiteit
Eindhoven
University of Technology

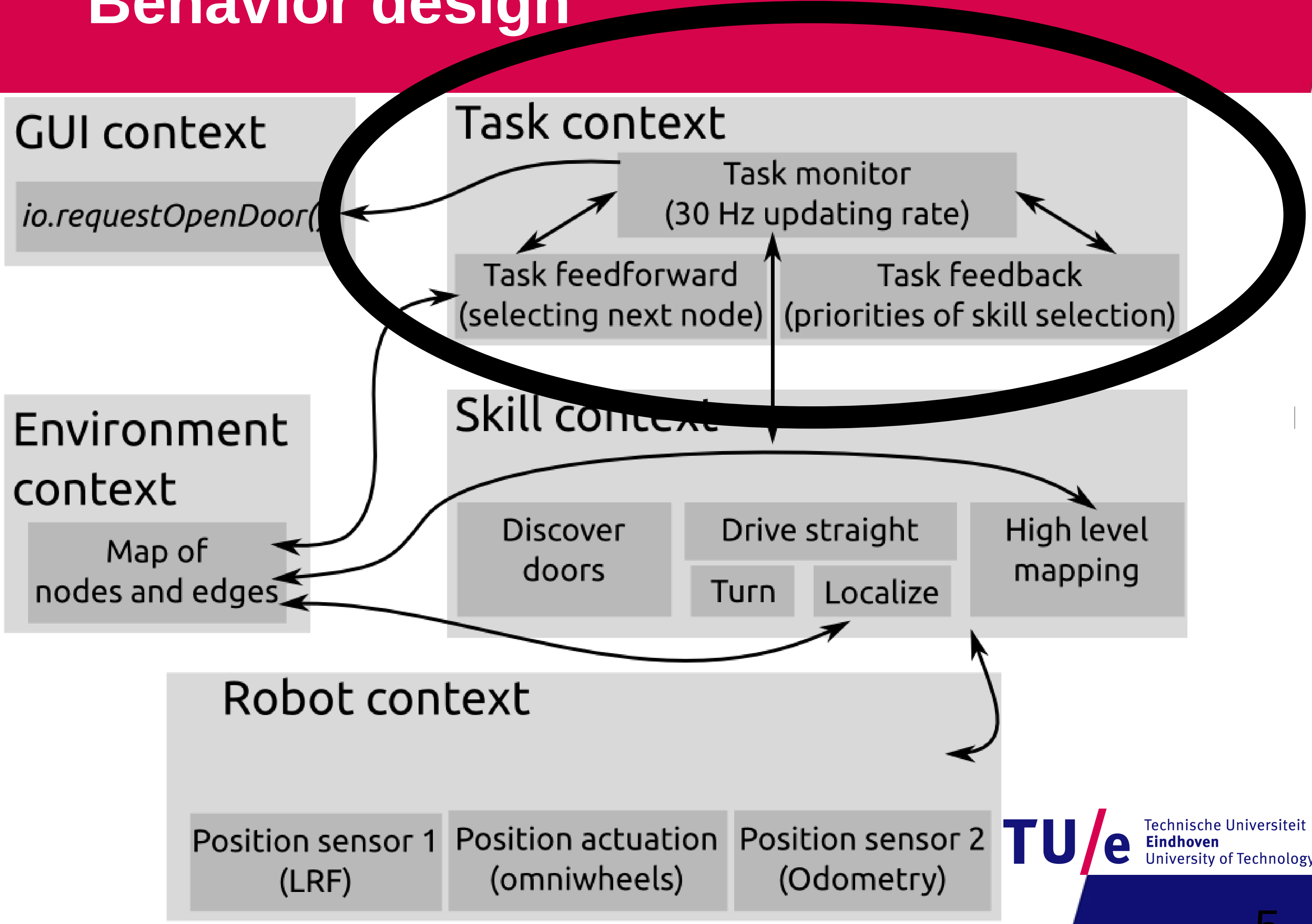
Behavior Design: Context connections



Behavior Design: Implementation



Behavior design



GUI context

io.requestOpenDoor()

Task context

Task monitor
(30 Hz updating rate)

Task feedforward
(selecting next node)

Task feedback
(priorities of skill selection)

Environment context

Map of
nodes and edges

Skill context

Discover
doors

Drive straight

Turn

Localize

High level
mapping

Robot context

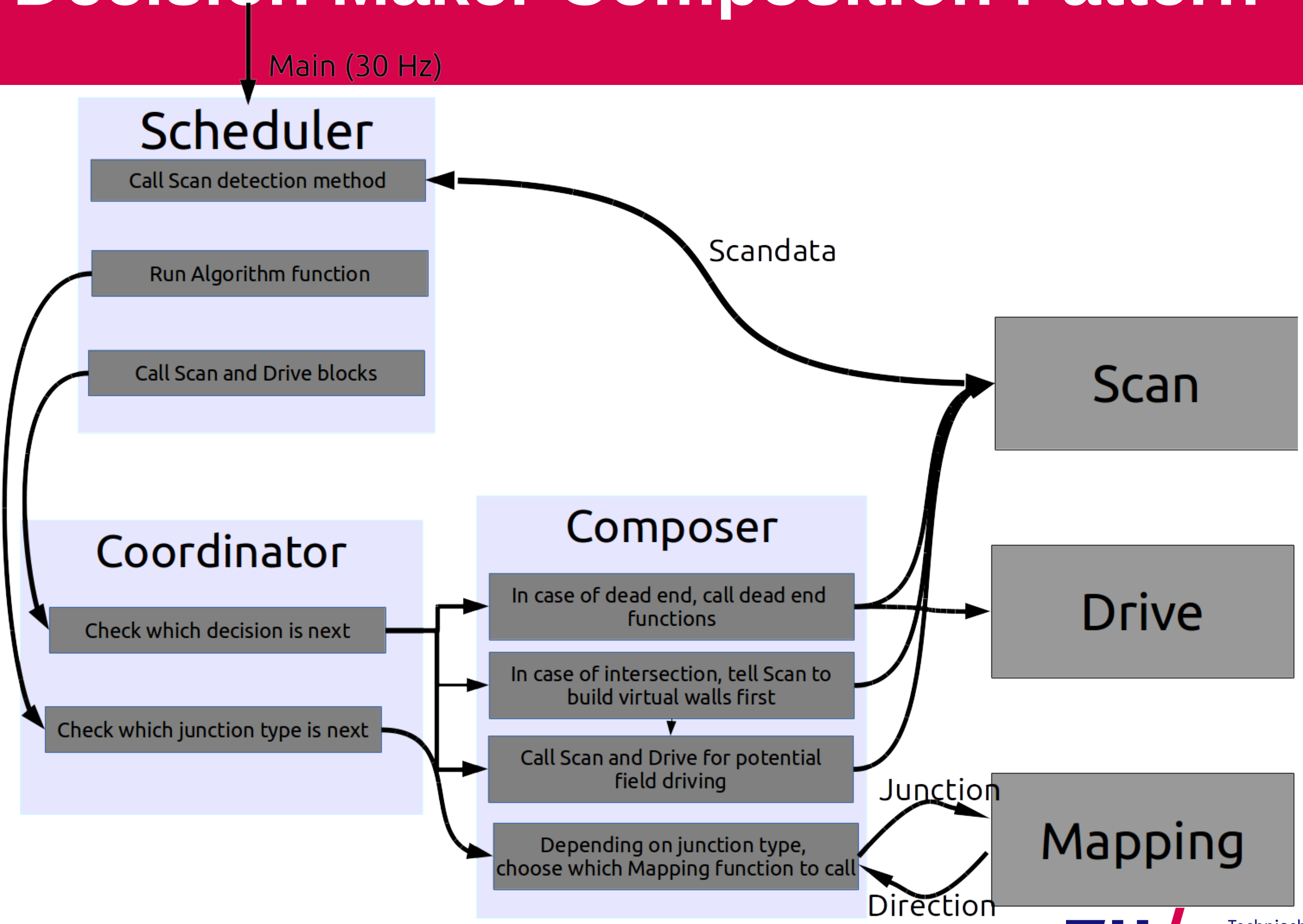
Position sensor 1
(LRF)

Position actuation
(omniwheels)

Position sensor 2
(Odometry)

Technische Universiteit
Eindhoven
University of Technology

Decision Maker Composition Pattern



Main (30 Hz)

Scheduler

Call Scan detection method

Run Algorithm function

Call Scan and Drive blocks

Scandata

Coordinator

Check which decision is next

Check which junction type is next

Composer

In case of dead end, call dead end functions

In case of intersection, tell Scan to build virtual walls first

Call Scan and Drive for potential field driving

Depending on junction type, choose which Mapping function to call

Scan

Drive

Mapping

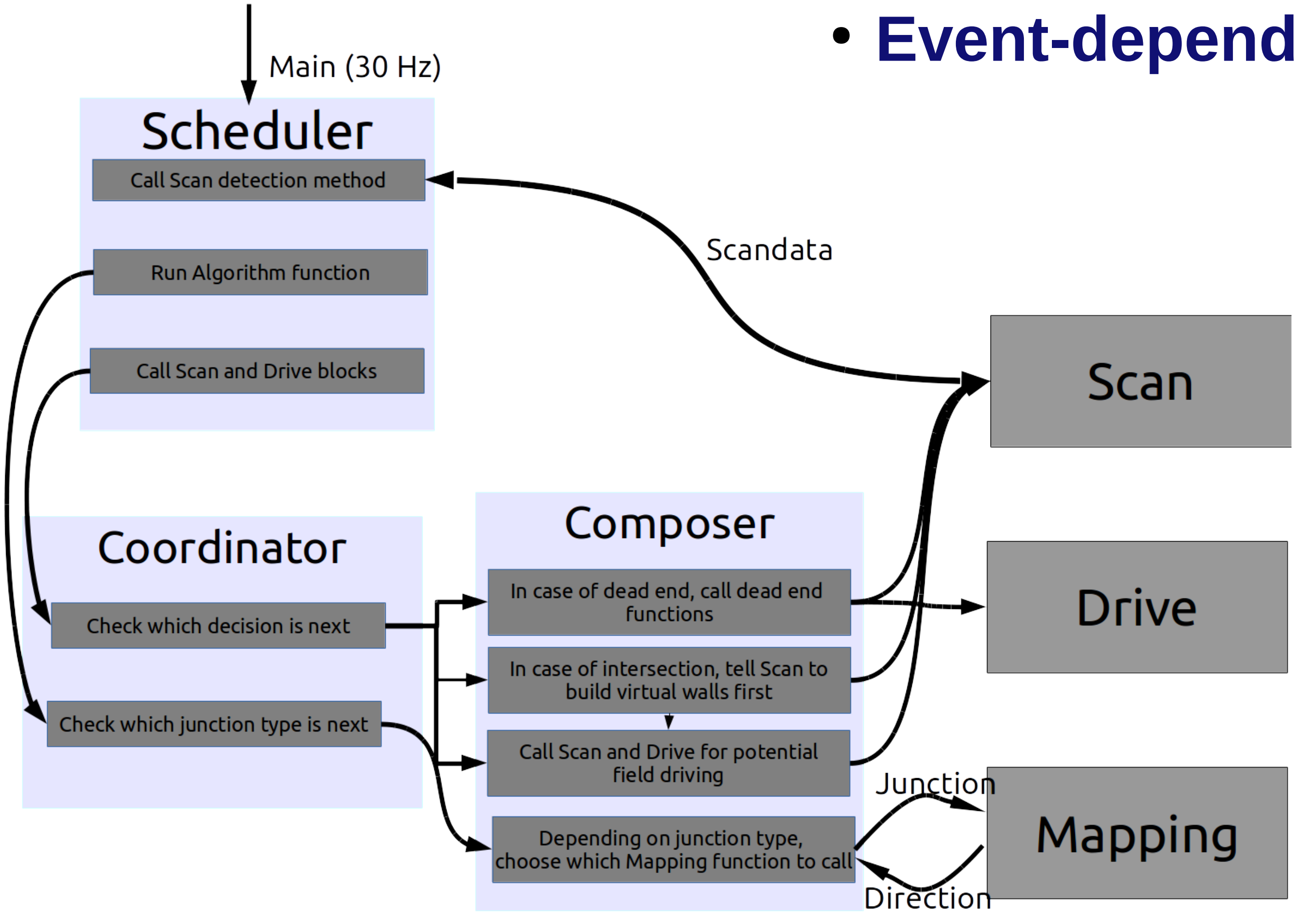
Junction

Direction

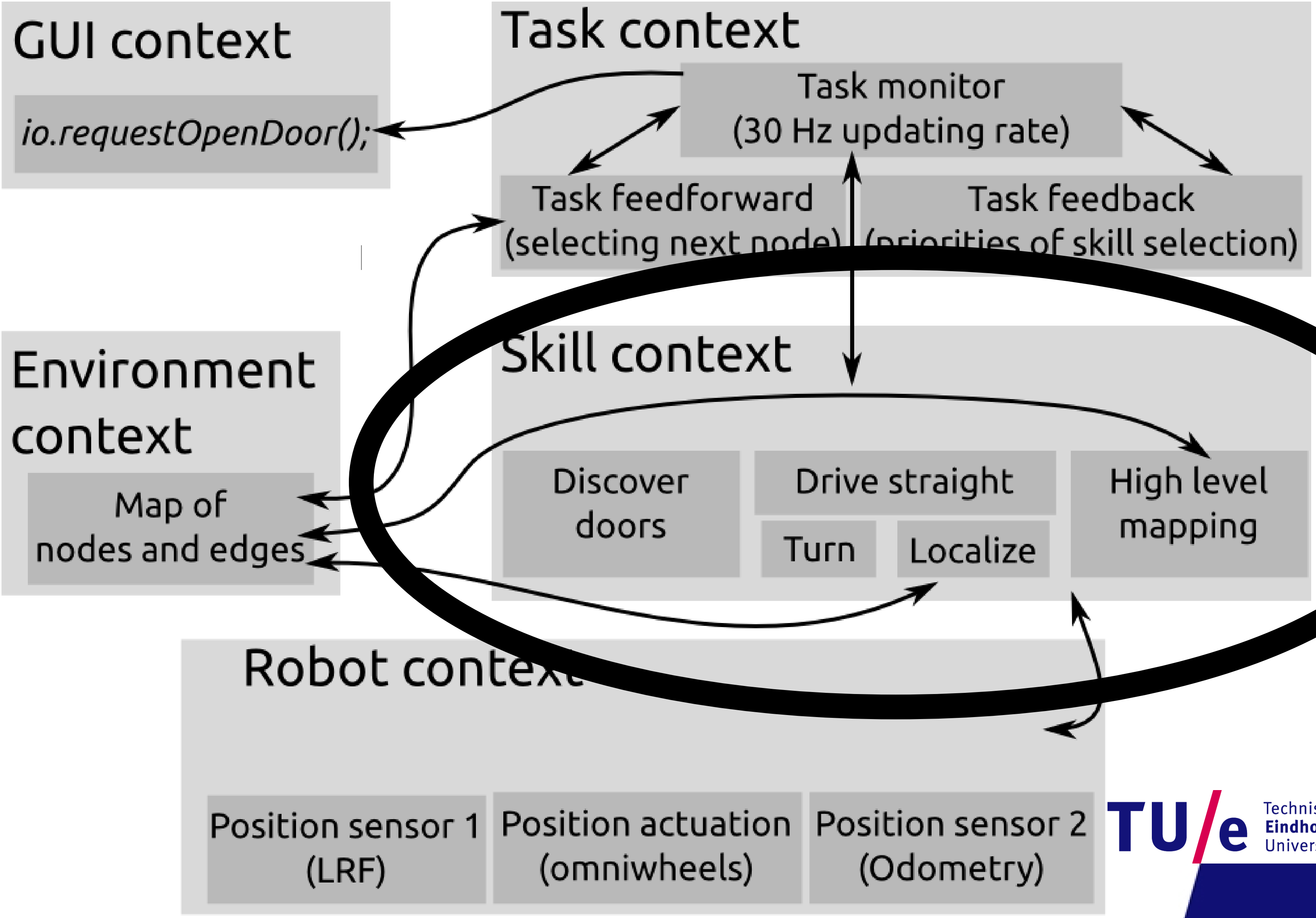
Technische Universiteit
Eindhoven
University of Technology

Decision Maker Composition Pattern

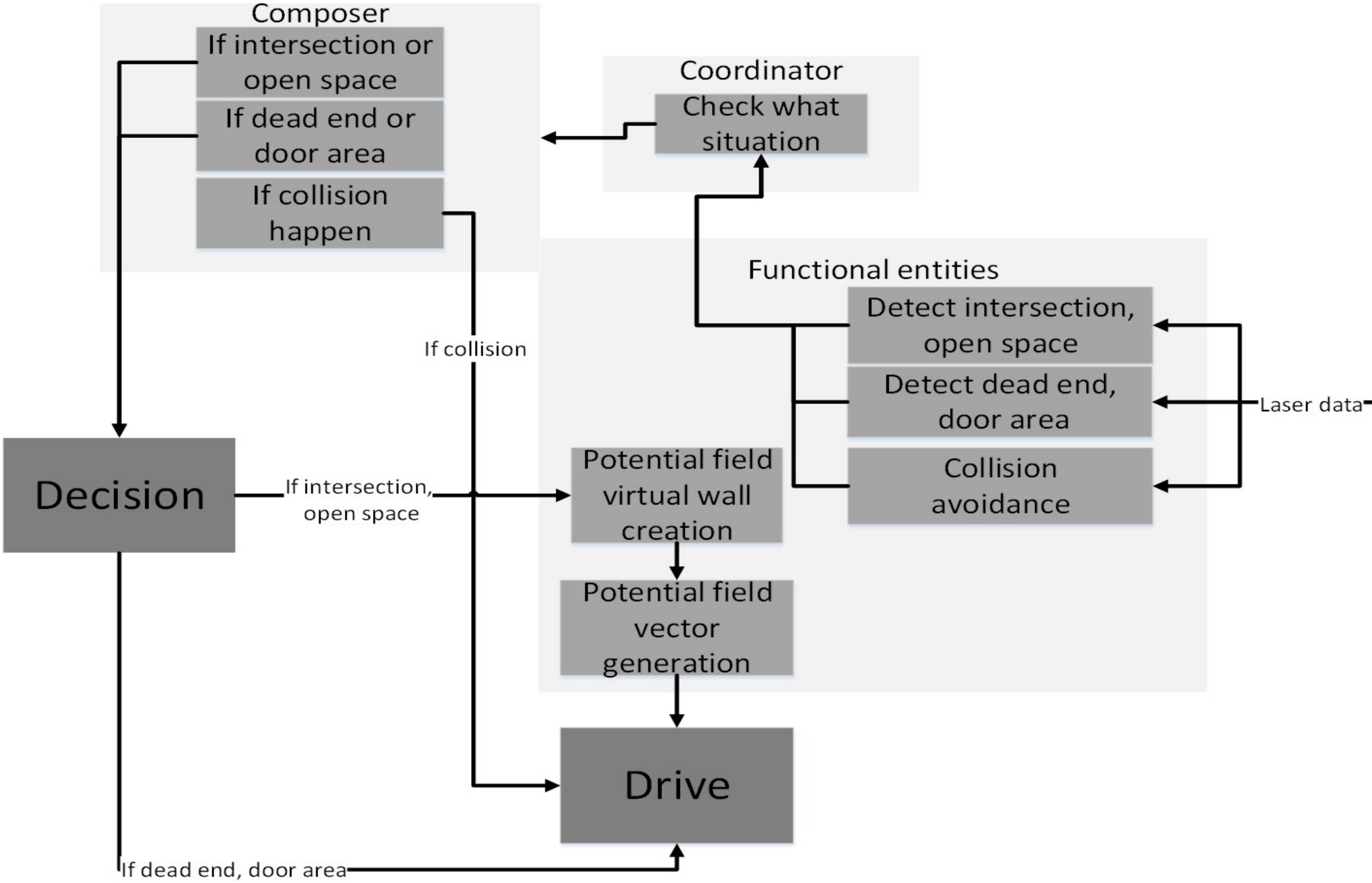
- 'Brains' of the robot
- Fixed updating frequency (30 Hz)
- Event-depended choices


Main (30 Hz)
Scheduler
Call Scan detection method
Run Algorithm function
Call Scan and Drive blocks
Scandata
Scan
Coordinator
Check which decision is next
Check which junction type is next
Composer
In case of dead end, call dead end functions
In case of intersection, tell Scan to build virtual walls first
Call Scan and Drive for potential field driving
Depending on junction type, choose which Mapping function to call
Drive
Junction
Mapping
Direction

Behavior Design



Scan Composition Pattern



Composer

If intersection or open space

If dead end or door area

If collision happen

Coordinator

Check what situation

Functional entities

Detect intersection, open space

Detect dead end, door area

Collision avoidance

Laser data

Potential field virtual wall creation

Potential field vector generation

Decision

If intersection, open space

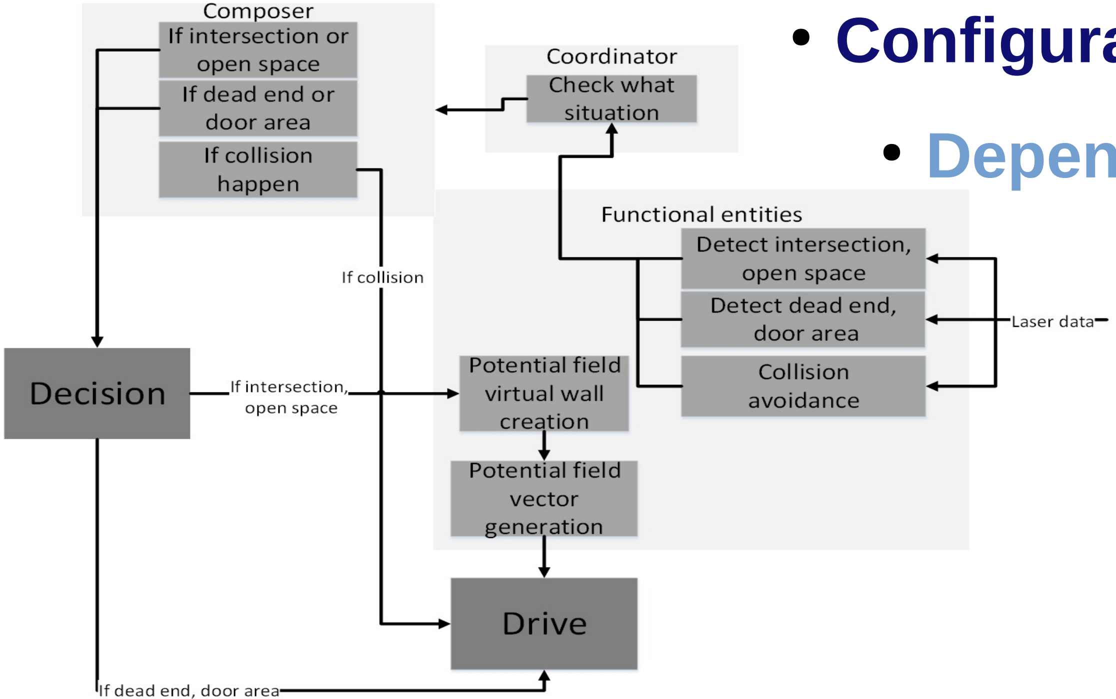
If collision

Drive

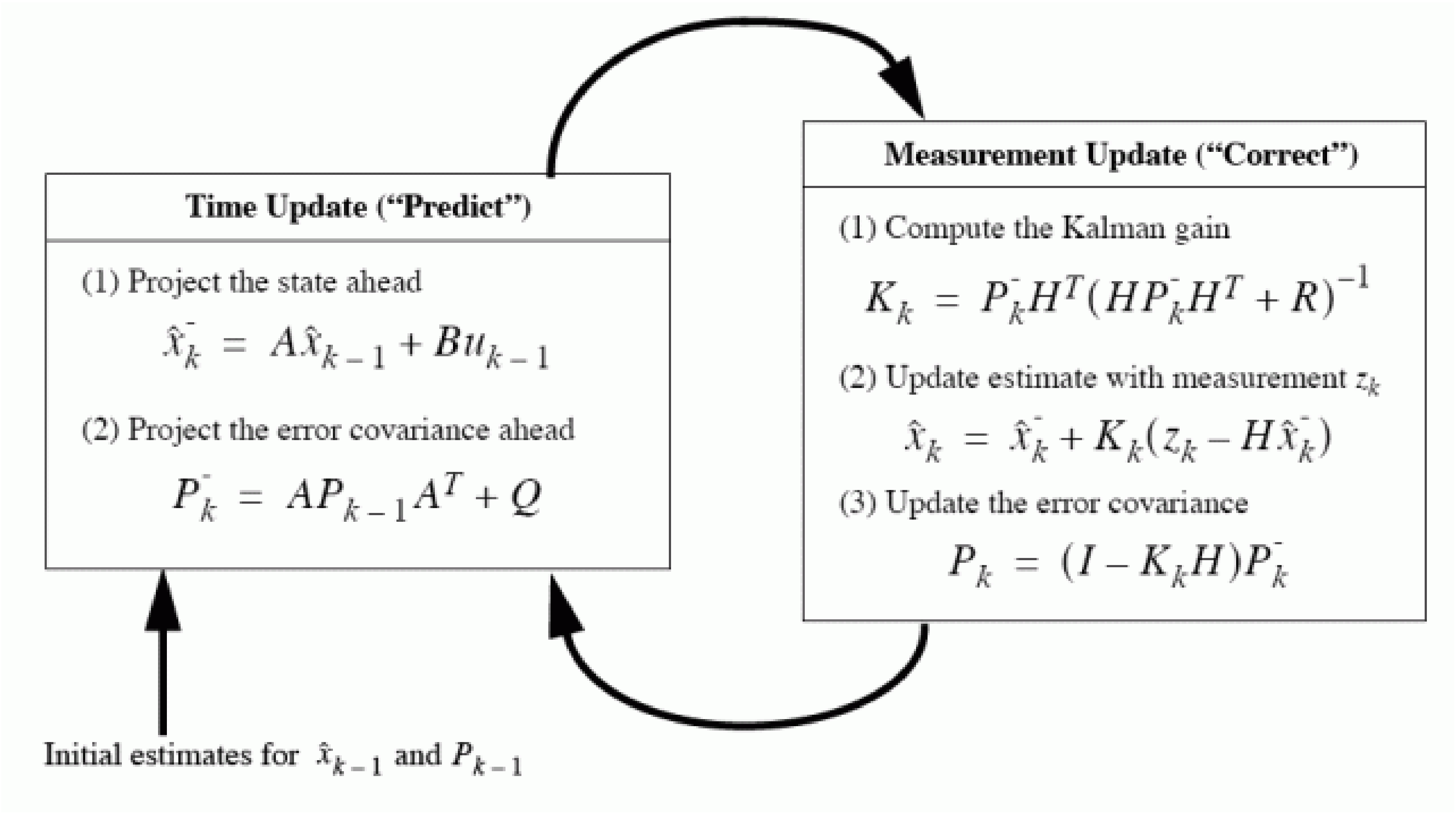
If dead end, door area

Scan Composition Pattern

- **Interprets LRF data**
  - **Low level → High level**
- **Implementation of potential fields**
  - **Cornering with virtual walls**
- **Configurator for 'drive.cpp'**
  - **Dependent on 'Decision Maker'**


Composer
If intersection or open space
If dead end or door area
If collision happen
Coordinator
Check what situation
Functional entities
Detect intersection, open space
Detect dead end, door area
Collision avoidance
Laser data
If collision
Decision
If intersection, open space
Potential field virtual wall creation
Potential field vector generation
Drive
If dead end, door area

Localization



Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

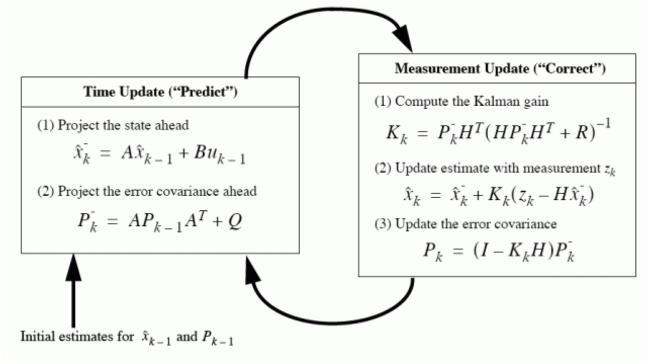
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

(3) Update the error covariance

$$P_k = (I - K_k H)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Localization



Time Update ("Predict")

(1) Project the state ahead

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

(2) Project the error covariance ahead

$$P_k^- = AP_{k-1}A^T + Q$$

Measurement Update ("Correct")

(1) Compute the Kalman gain

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$$

(2) Update estimate with measurement z_k

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

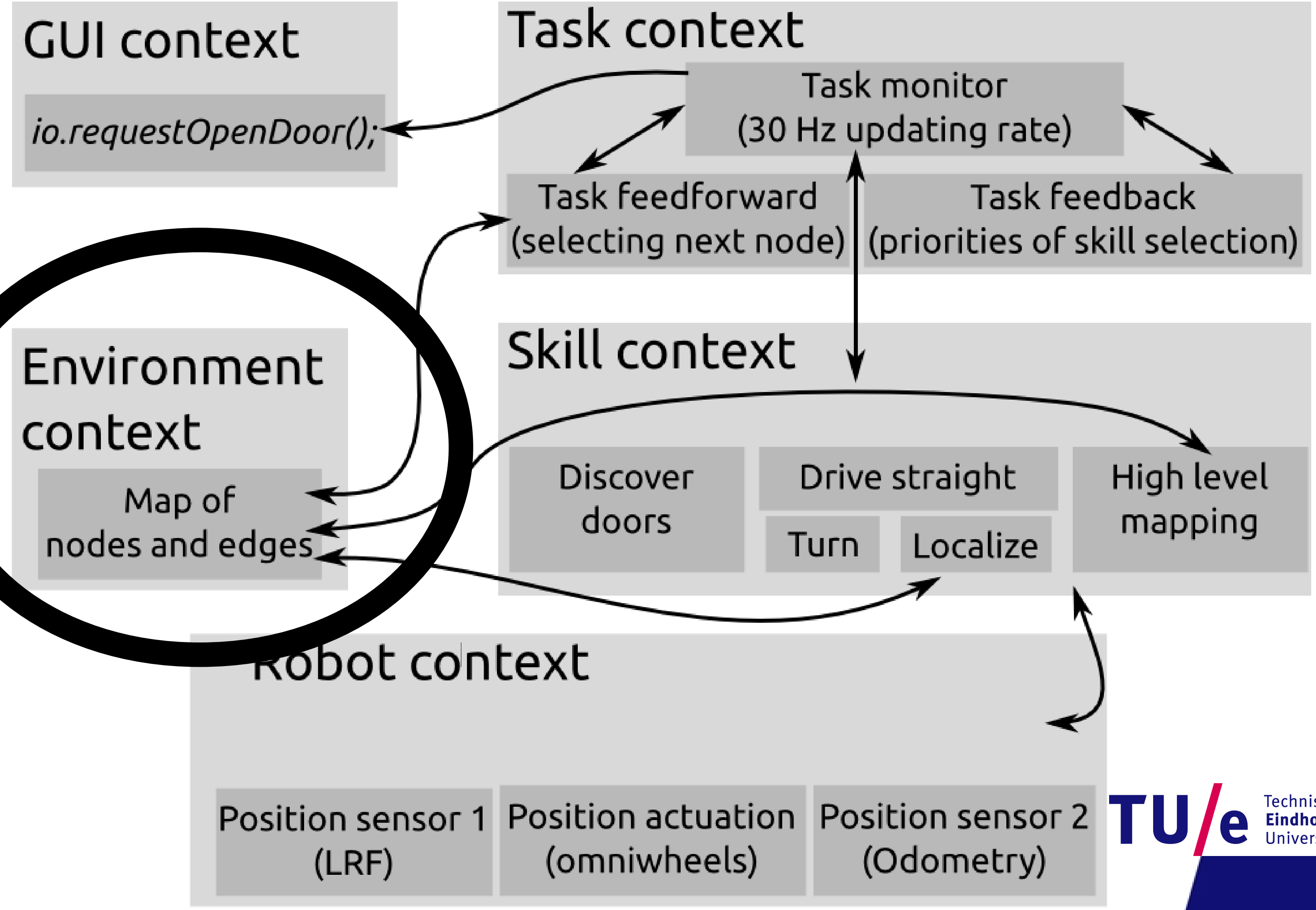
(3) Update the error covariance

$$P_k = (I - K_kH)P_k^-$$

Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

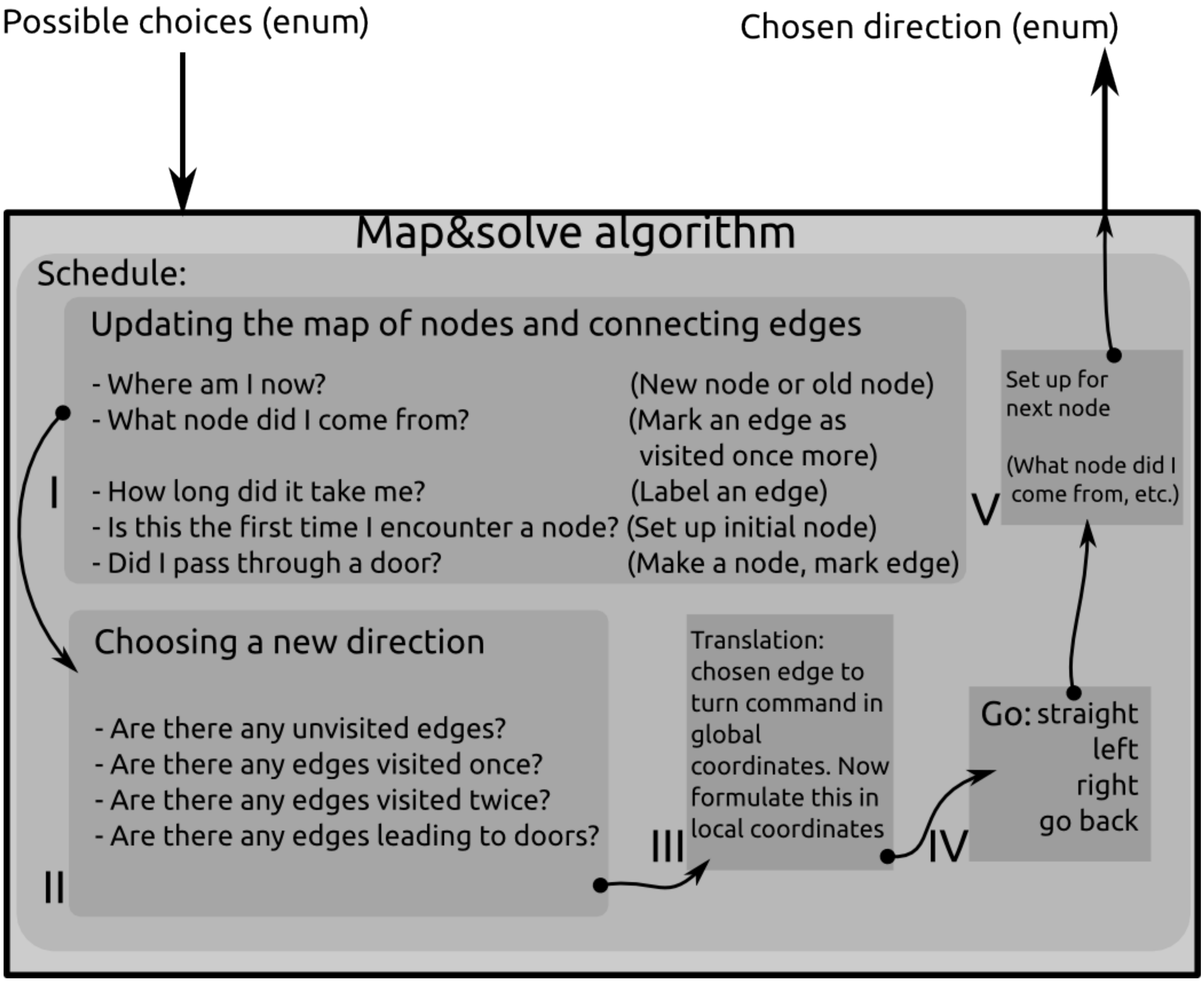
- Combining sensor data
- Returns global coordinates
  - X
  - Y
  - Theta
- Dynamic switching of 'R'-matrix
  - Odometry possibly unreliable
  - LRF might loose track of a wall

Behavior Design



Technische Universiteit
Eindhoven
University of Technology

Mapping and Solving Composition Pattern



Possible choices (enum)

Chosen direction (enum)

Map&solve algorithm

Schedule:

Updating the map of nodes and connecting edges

- Where am I now? (New node or old node)
- What node did I come from? (Mark an edge as visited once more)
- How long did it take me? (Label an edge)
- Is this the first time I encounter a node? (Set up initial node)
- Did I pass through a door? (Make a node, mark edge)

Choosing a new direction

- Are there any unvisited edges?
- Are there any edges visited once?
- Are there any edges visited twice?
- Are there any edges leading to doors?

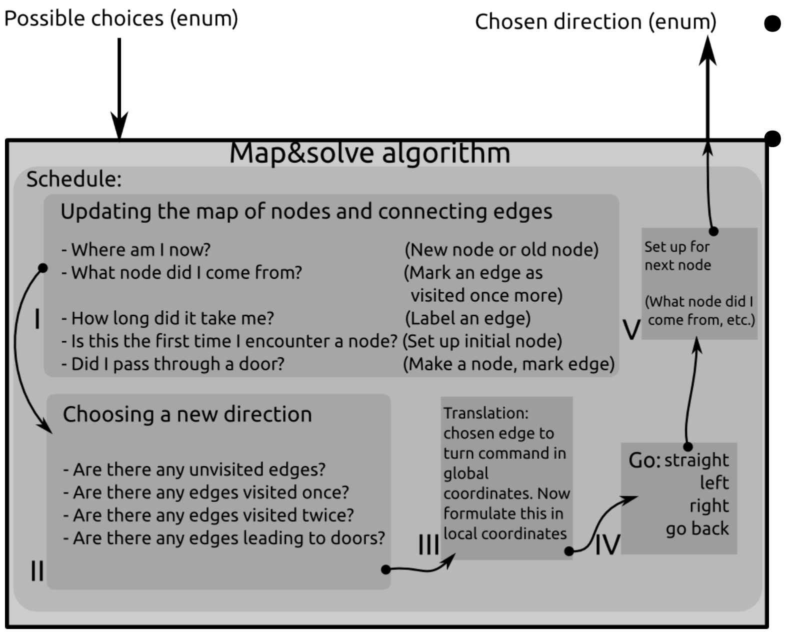
Translation: chosen edge to turn command in global coordinates. Now formulate this in local coordinates

Go: straight
left
right
go back

Set up for next node

(What node did I come from, etc.)

Mapping and Solving Composition Pattern



Possible choices (enum)

Chosen direction (enum)

Map&solve algorithm

Schedule:

Updating the map of nodes and connecting edges

- Where am I now? (New node or old node)
- What node did I come from? (Mark an edge as visited once more)
- How long did it take me? (Label an edge)
- Is this the first time I encounter a node? (Set up initial node)
- Did I pass through a door? (Make a node, mark edge)

I

Choosing a new direction

- Are there any unvisited edges?
- Are there any edges visited once?
- Are there any edges visited twice?
- Are there any edges leading to doors?

II

III

Translation: chosen edge to turn command in global coordinates. Now formulate this in local coordinates

IV

Go: straight
left
right
go back

V

Set up for next node

(What node did I come from, etc.)

- **Tremaux's maze solving algorithm**
- **Mapping a mix of:**
  - Higher level: Graph
  - Lower level: Node position

Summary

- **Behavior design as backbone for entire project**
- **Behavior implementation as guideline for classes/separate *.CPP-files**
- **Separate 'brain' controlling all other functionalities**
- **Implementation of :**
  - **Tremaux's maze-solving algorithm**
  - **Kalman filtering for global coordinates**
  - **Potential field method for basic driving**

Stuff that we expected from EMC

- **Frequency domain motion control in C++**
  - z-domain or s-domain
- **State-space motion control in C++**
- **More low-level lectures, more elaborate examples**

Technische Universiteit
Eindhoven
University of Technology

Stuff that we've learned

- **Top down software design using diagrams**
  - Composition patterns vs behavior diagrams

- **Bottom up implementation**
  - Easier to get working code from scratch
  - Difficulties in integration of the entire software package
  - Valuable information from 'dirty fixes', but should be re-written for the final product
  - *“Shoot first, ask questions later”*-approach

- **Coordinating team work with a group of 10 people**
  - Decoupling problems, mostly trial and error