

## 5. V E

### Beispiel

package thi.robcon.app.script (Unterverzeichnis im Projekt; Java-Konvention => alles klein geschrieben)  
Eclipse: Verschiebung mit Refactor => Rename

import thi.robcon.ecar | RobScript  
                          ↑  
                          package      Klasse  
                          Eclipse: Strg+Leer  
                          generiert

Name der Klasse (groß in Java)

public class ServiceRoboticsScripts extends IRobConScript {  
    ↳ Oberklasse (nur 1 möglich)

    public ServiceRoboticsScripts () { ... } ← Konstruktor

    protected void script() { patrol(), } ← hier Implementierung virtuelle Methode (Polymorphismus)

    protected void patrol() {

        useRobot (ECarDefines.JSIM-DX1), ← einfacher Simulator

        // useRobot (ECarDefines.DX4), ← realer Pioneer Robot C107  
        for (int i = 0; i < 10; i++) { ← analog C++

            move(1000); ← 1m vorwärts fahren

            if (i % 2 == 0) ← modulo 2 Berechnung

                gripClose(); } In Simulatoren nicht sichtbar

            else gripOpen(); }

            turn(180); ← umdrehen

? } ← In JSIM setPose(x,y);

}

5.

# Objektorientierte Programmierung IAS

## 5.1 Programmierprinzip

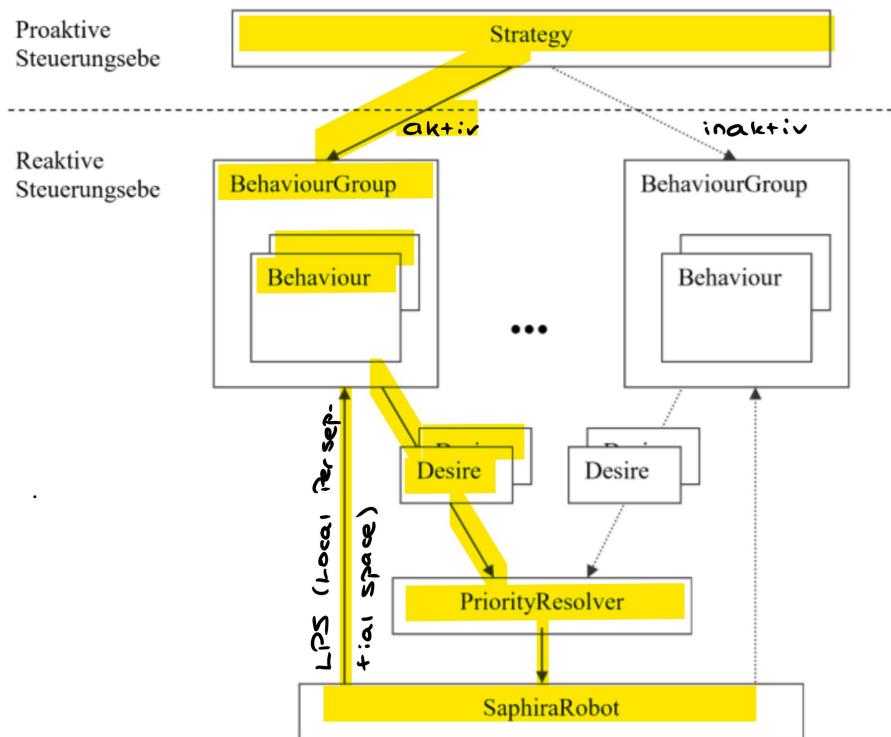
Grundlage: THIRobCon mit Saphira-Architektur

Proaktive Ebene: Endlicher Automat

Reaktive Ebene: Verhaltensmuster mit Resorbierung in Gruppen

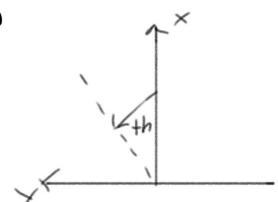
Basisklassen THIRobCon

HANDOUT



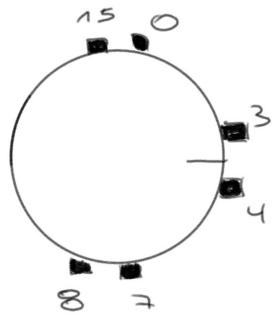
## 5.2 Ausgewählte Methoden Basisklassen

SaphiraRobot



- `Pose getPose()` → Positionskoordinaten Fahrzeug in  $(x, y, \text{th})$
- LPS      {
  - `double getTransVel()`
  - `double getRotVel()`} Aktuelle Geschwindigkeiten Fahrzeug

| • int getSonarRange(int n)  
 | → Entfernungsmessung Sonar n in mm  
 | bezogen auf Fahrzeugmittelpunkt, max. 3m



Fzg-Config { int getRadius() → Abfrage Roboteradius

### Behaviour

Virtuelle Oberklasse Verhaltensmuster mit abstrakter Methode fire(), vom Resolver zyklisch alle 100ms aufgerufen

Erzeugen Instanzen von Unterklassen der Oberklasse Desire für den PriorityResolver als Aktionsvorschläge

### Unterklassen von Desire

- DesTransVel, DesRotVel → Wünsche für Geschwindigkeiten
- DesCanPan, DesCanTilt → Wünsche für Kamerabewegung
- DesGrip, DesLift → Wünsche für Manipulatorpositionen

## 6. VE

### Priority Resolver

Statische Priorität (priority): Generelle Wichtigkeit des Verhaltensmusters; einmalig als Wert  $[0; 100]$  festgelegt.  
niedrigstes ↗ höchstes

Dynamische Priorität (strength): situationsabhängig in jedem Zyklus neu als Wert  $[0; 1.0]$  festgelegt.  
niedr. ↗ hoch.

## Algorithmus:

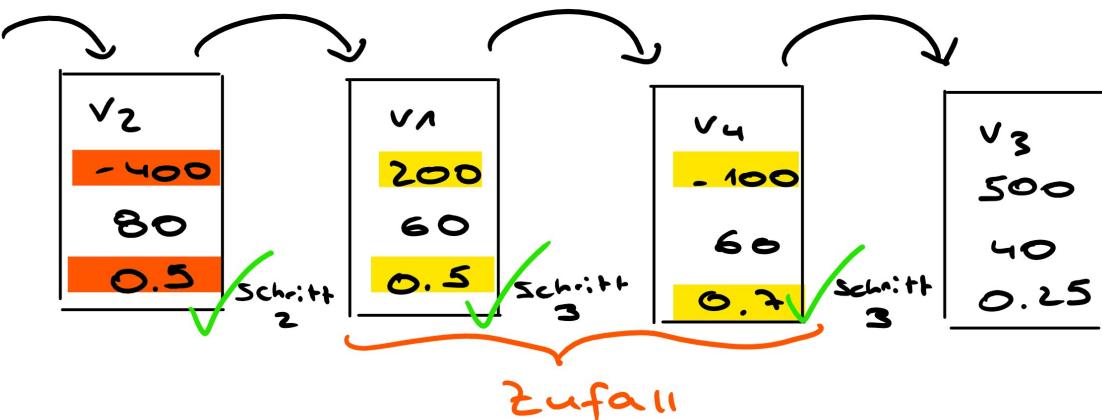
- Ordne Desires nach fallender priority der erzeugenden Verhaltensmuster
- Für alle Steuergeräte s
  - { Desire r, c, r.value = 0, r.strength = 0;
  - Für alle Desire d, solange r.strength < 1
    - { Für alle Desire d<sub>1</sub>, ..., d<sub>n</sub> mit d<sub>i</sub>.priority = d.priority
      - { c.value =  $\sum_{i=1}^n d_i \cdot value \times d_i \cdot strength;$
      - c.strength =  $\frac{1}{n} \sum_{i=1}^n d_i \cdot strength;$
      - r.value += c.value;
      - r.strength += c.strength;
    - s = r.value / r.strength;

## Beispiel: Resolution TransVel

geg:

DesTransVel v1	v2	v3	v4
value 200	-400	500	-100
priority 60	80	40	60
strength 0.5	0.5	0.25	0.7

## Schritt 1:



## Schritt 2:

$$\begin{aligned}c.\text{value} &= -200 \\c.\text{strength} &= 0.5 \\r.\text{value} &= -200 \\r.\text{strength} &= 0.5\end{aligned}$$


## Schritt 3:

$$\begin{aligned}c.\text{value} &= 200 \cdot 0.5 + (-100) \cdot 0.7 = 30 \\c.\text{strength} &= \frac{1}{2}(0.5+0.7) = 0.6 \\r.\text{value} &= -200 + 30 = -170 \\r.\text{strength} &= 0.5 + 0.6 = 1.1\end{aligned}$$

## Schritt 4:

$$s = \frac{-170}{1.1} = -154 \quad (\text{TransVel})$$

## Strategy

Oberklasse endlicher Automat mit abstrakter  
Methode plan() in 100ms Zyklus aufgerufen  
Aktivierung / Deaktivierung Automatenknoten

## 5.2 Basisverhaltensmuster

Behaviour	Beschreibung
Beh. ConstTransVel Beh. Const Red Vel	Fahren mit konstanter Geschwindigkeit
Beh. Lim For Beh. Lim Back	Annähern an Hindernisse mit wechselnden Geschwindigkeiten <small>Lim-Limiter For-Forward Back-Backward</small>
Beh. Cam Init	Initialisierung Kamera
Beh. Move Beh. Turn	Fahre nach Odometriiedaten

## Beispiel: Konstantes Fahren / Andocken

```
1 package thi.irobcon.app.beaviour.constantmotion;
2
3 import thi.irobcon.ecar.ECarDefines;[]
4
5 public class CMMain {
6
7     public static void main(java.lang.String[] args) {
8
9         // SaphiraRobot robot = new SaphiraRobot(ECarDefines.DX3);
10        SaphiraRobot robot = new SaphiraRobot(ECarDefines.JSIM_DX3);
11
12        BehGroup dock = new BehGroup("Dock");
13        BehConstTransVel cv = new BehConstTransVel("ConstVel", 400);
14        dock.add(cv, 50); → statische Priorität
15        robot.add(dock);
16        robot.run();
17    }
18
19 }
```

... main() { ← Generieren  
new SaphiraRobot ← Sim. / Realer Roboter  
Beh. Group ← Behaviour ← Name  
dock ← add (Priority)  
cv (400  $\frac{mm}{s}$ )  
robot.run(), ← Echtzeitzyklus 100ms

```
1 package thi.irobcon.app.beaviour.dock;
2
3 import thi.irobcon.ecar.ECarDefines;[]
4
5 public class DOMain {
6
7     public static void main(java.lang.String[] args) {
8
9         // SaphiraRobot robot = new SaphiraRobot(ECarDefines.DX3);
10        SaphiraRobot robot = new SaphiraRobot(ECarDefines.JSIM_DX3);
11        //robot.addLaser();
12
13        BehGroup dock = new BehGroup("Dock");
14        BehConstTransVel cv = new BehConstTransVel("ConstTransVel", 200);
15        BehLimFor lf = new BehLimFor("LimFor", 1000, 2000, 100);
16        dock.add(lf, 80);
17        dock.add(cv, 50); → Prio höher als konstantes
18        robot.add(dock);
19        robot.run(); Fahren
20    }
21
22 }
```

## 7. VE

### Beispiel: Andocken

```
package ... dock;
```

```
...  
... main (...) {
```

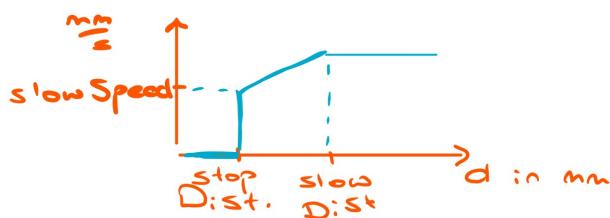
```
...
```

```
BehGroup dock = ...
```

```
BehConstTransVel cv = new BehConstTransVel("CV", 200);
```

```
...
```

```
BehLimFor lf = new BehLimFor("LF", 1000, 2000, 100);
```



```
dock.add (cv, 50); höher als konstant.  
fahren
```

```
dock.add (lf, 80); statische Priorität
```

```
robot.add(dock); „Dem Roboter bekannt machen“
```

```
robot.run(); ← Start des Echtzeitzyklus Resolver
```

```
1 package thi.irobcon.app.behaviour.dock;  
2  
3 import thi.irobcon.ecar.ECarDefines;□  
4  
5 public class DOMain {  
6  
7     public static void main(java.lang.String[] args) {  
8  
9         // SaphiraRobot robot = new SaphiraRobot(ECarDefines.DX3);  
10        SaphiraRobot robot = new SaphiraRobot(ECarDefines.JSIM_DX3);  
11        //robot.addLaser();  
12  
13        BehGroup dock = new BehGroup("Dock");  
14        BehConstTransVel cv = new BehConstTransVel("ConstTransVel", 200);  
15        BehLimFor lf = new BehLimFor("LimFor", 1000, 2000, 100);  
16        dock.add(lf, 80);  
17        dock.add(cv, 50);  
18        robot.add(dock);  
19  
20        robot.run();  
21    }  
22}
```

## 5.3 Programmierung eigener Verhaltensmuster

### Beispiel: Orthogonales Andocken an Wand

```
1 package thi.irobcon.app.behaviour.rightangledock;
2
3 import thi.irobcon.saphira.desire.DesRotVel;
4
5 public class BehAlign extends Behaviour { → Oberklasse aller Verhaltensmuster
6     → Ableiten
7     protected int tolerance; → Konstruktor
8     protected int rotVel;
9
10    public BehAlign(String behName, int tolerance, int rotVel) {
11        super(behName); ( Oberklassenkonstruktor )
12        this.tolerance = tolerance;
13        this.rotVel = rotVel;
14    }
15
16    ↓ virtuell in Oberklasse
17    public void fire() { ↓ id
18        int leftDist, rightDist; → Rücksicht auf andere Verhaltensmuster
19        Pose relObstaclePose = new Pose(); ( Brauchen wir später )
20
21    /*
22     * Mit Sonar explizit
23     * leftDist = robot.getSonarRange(3);
24     * rightDist = robot.getSonarRange(4); bei add Beh Group
25     */
26
27     /*
28     * Mit Kegel
29     * leftDist = (int) (robot.checkPolar(2, 10, relObstaclePose) -
30     *                 robot.getRadius());
31     System.out.println("Position linkes Hindernis: (" + relObstaclePose.getX() + "," +
32     *                     relObstaclePose.getY() + ")");
33     rightDist = (int) (robot.checkPolar(-10, -2, null) -
34     *                     robot.getRadius());
35
36     */
37     /*
38     * Mit Box
39     * leftDist = (int) (robot.checkBox(2500, 500, 100, 100,
40     *                                relObstaclePose) - robot.getRadius());
41
42     System.out.println("Left obstacle at relative position (" + relObstaclePose.getX() +
43     *                     "," + relObstaclePose.getY() +
44     *                     "," + relObstaclePose.getTh() + ") in distance " +
45     *                     (leftDist + robot.getRadius()));
46
47     rightDist = (int) (robot.checkBox(2500, -500, 100, -100, null)
48     *                     - robot.getRadius());
49
50     /*
51     * Mit Positionsabschätzung
52     * Pose robPose = robot.getPose();
53     * System.out.println("Robot at position (" + robPose.getX() +
54     *                     "," + robPose.getY() +
55     *                     "," + robPose.getTh() + ")");
56
57     Pose absObstaclePose = new Pose(robPose.getX() + relObstaclePose.getX(),
58         robPose.getY() + relObstaclePose.getY(),
59         robPose.getTh() + relObstaclePose.getTh());
59
60     System.out.println("Left obstacle at absolute position (" + absObstaclePose.getX() +
61     *                     "," + absObstaclePose.getY() +
62     *                     "," + absObstaclePose.getTh() + ")");
63
63     double distLeft = robPose.findDistanceTo( absObstaclePose );
64     double angleLeft = robPose.findAngleTo( absObstaclePose );
65
66     System.out.println("Validating distance = " + distLeft + " and angle " + angleLeft );
67
68     /*
69     * Kurskorrektur
70     * System.out.println("Left = " + leftDist + " right = " + rightDist);
71
72     if ( leftDist - rightDist > tolerance )
73     { Wunsch an Wunschchart Wunschkwert
74         Resolver.addDesire(new DesRotVel(-rotVel, 1.0)); dynamische Priorität
75         System.out.println("Turn right");
76     }
77     else if ( rightDist - leftDist > tolerance )
78     {
79         Resolver.addDesire(new DesRotVel(rotVel, 1.0));
80         System.out.println("Turn left"); → andere Verhaltensmuster können auch kurs korrigieren
81     }
82     else addDesire(new DesRotVel(0, 0.5));
83
84 }
```

```
public class BehStop extends ... {
```



```
protected int stopDistance;
```

```
public BehStop ... (Konstruktor)
```

```
public void fire() {
```

```
    if (robot.getSonarRange(3) <= stopDistance)  
        success(); ← Erfolgsmeldung an strategische Ebene
```

```
}
```

```
}
```

```
public class RDStrategy extends Strategy {  
    ↑ Oberklasse für  
    ↓ strategy
```

```
protected BehGroup dock;
```

```
public RDStrategy (BehGroup dock) {
```

```
    ... this.dock = dock;
```

```
    dock.activateExclusive();
```

```
}
```

↑ Knotenaktivierung endlicher Automat

alle 100ms  
ausgeführt

```
→ public void plan() {
```

```
    if (dock.isSuccess()) ← Abfrage, ob BehGroup  
        stopRunning();      erfolgreich fertig  
    } ← Beenden Echtzeitzyklus
```

```

public class RDMain {
    ...
    BehAlign al = new BehAlign("AL", 30, 5);
    BehStop st = new BehStop("ST", 800);
    BehLimFor lf = ...;
    ...
    dock.add(al, 75);
    dock.add(st, 80);
    ...
    robot.add(new RDStrategy(dock));
    robot.run();
}

```

tolerance  
↓ ← rotVel  
größer als  
stopDist im  
BehLimFor

```

1 package thi.irobcon.app.behaviour.rightangledock;
2
3 import thi.irobcon.saphira.reactive.Behaviour;
4
5 public class BehStop extends Behaviour {
6     protected int stopDistance;
7
8     public BehStop(String behName, int stopDistance) {
9         super(behName);
10        this.stopDistance = stopDistance;
11    }
12
13     public void fire() {
14         int dist = robot.getSonarRange(3);
15         if (dist <= stopDistance) {
16             System.out.println("Stop");
17             success();
18         }
19     }
20 }
21
22 }
23
24
25

```

```

1 package thi.irobcon.app.behaviour.rightangledock;
2
3 import thi.irobcon.saphira.proactive.Strategy;
4
5 public class RDStrategy extends Strategy {
6     protected BehGroup dock;
7
8     public RDStrategy(BehGroup dock) {
9         super();
10        this.dock = dock;
11        addOutput("Try to dock ...");
12        dock.activateExclusive();
13    }
14
15     public void plan() {
16         if (dock.isSuccess()) {
17             addOutput("done\nSucceed ... stop");
18             stopRunning();
19         }
20     }
21 }
22
23
24
25

```

```

1 package thi.irobcon.app.behaviour.rightangledock;
2
3 import thi.irobcon.ecar.ECarDefines;
4
5 public class RDMain {
6
7     public static void main(java.lang.String[] args) {
8
9         // SaphiraRobot robot = new SaphiraRobot(ECarDefines.DX3);
10        SaphiraRobot robot = new SaphiraRobot(ECarDefines.JSIM_DX3);
11
12        robot.addLaser();
13        robot.setRobPose(new Pose(0, 0, 20));
14
15        BehGroup dock = new BehGroup("Dock");
16        BehConstTransVel cv = new BehConstTransVel("ConstTransVel", 200);
17        BehLimFor lf = new BehLimFor("LimFor", 400, 2000, 100);
18        BehAlign al = new BehAlign("Align", 30, 5);
19        BehStop st = new BehStop("Stop", 800);
20
21        dock.add(lf, 80);
22        dock.add(cv, 50);
23        dock.add(al, 75);
24        dock.add(st, 80);
25        robot.add(dock);
26
27        robot.add(new RDStrategy(dock));
28
29        robot.run();
30
31    }
32
33
34
35
36
37

```

## 5.4 Sensorverarbeitung

### 5.4.1 Odometriedaten und Koordinaten

Standardkoordinatensysteme

ENU ( East - North - Up ) bei Landfahrzeugen

NED ( North - East - Down ) bei Luftfahrzeugen

6 Basen ( $x, y, z$ ) für Raumpunkt

( yaw, pitch, roll ) für Orientierung ( Eulerwinkel )

Drehung um  $z$  um  $y$  um  $x$   
( Höhe )