

# GMD - Forschungszentrum Informationstechnik GmbH



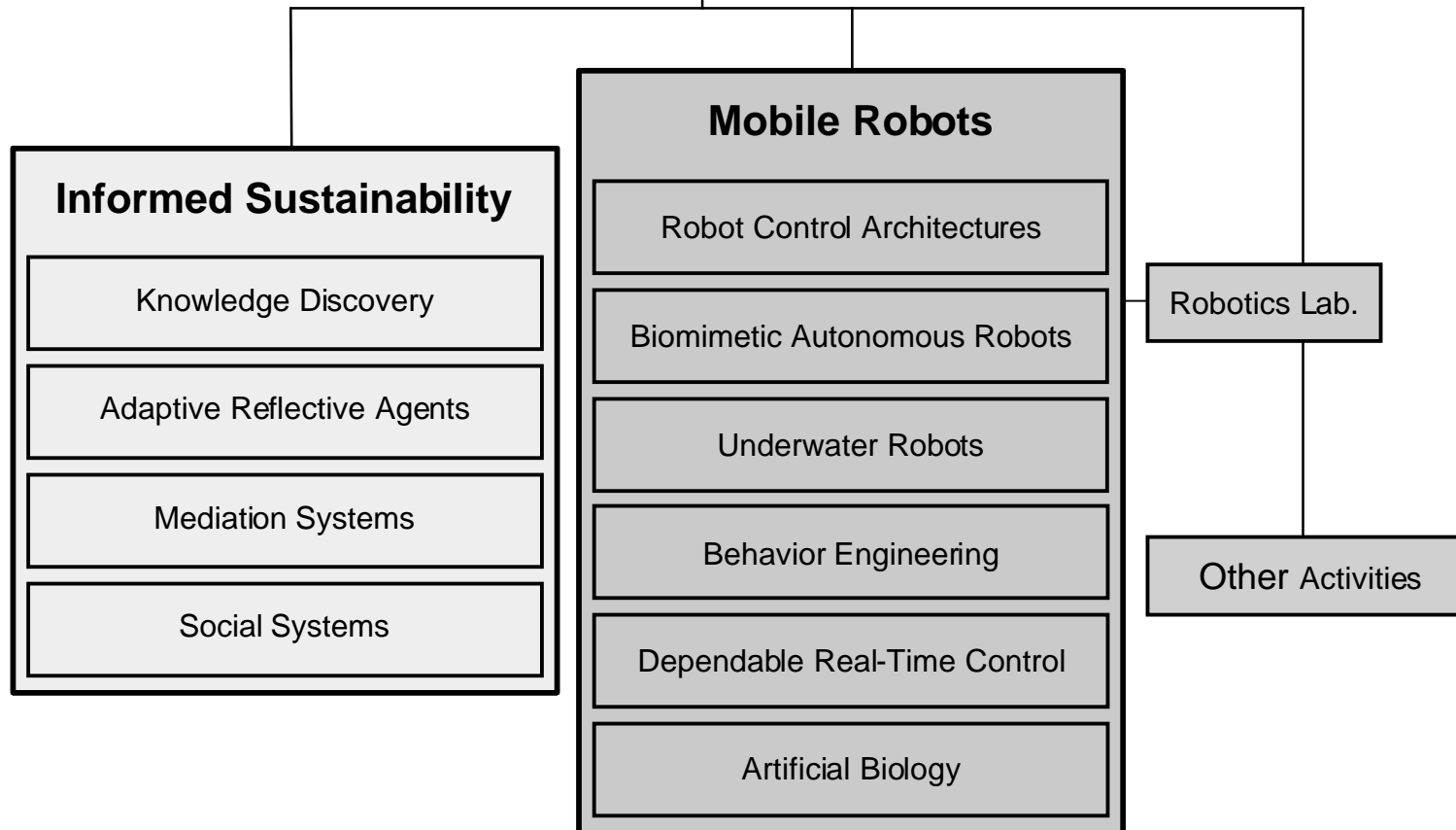
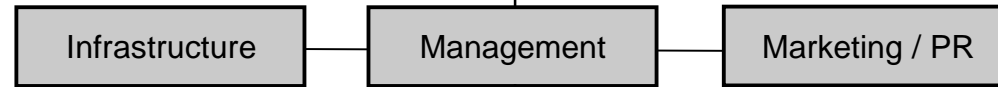
Institute for  
Autonomous intelligent Systems (AiS)  
Schloß Birlinghoven  
D-53754 Sankt Augustin  
<http://ais.gmd.de/>



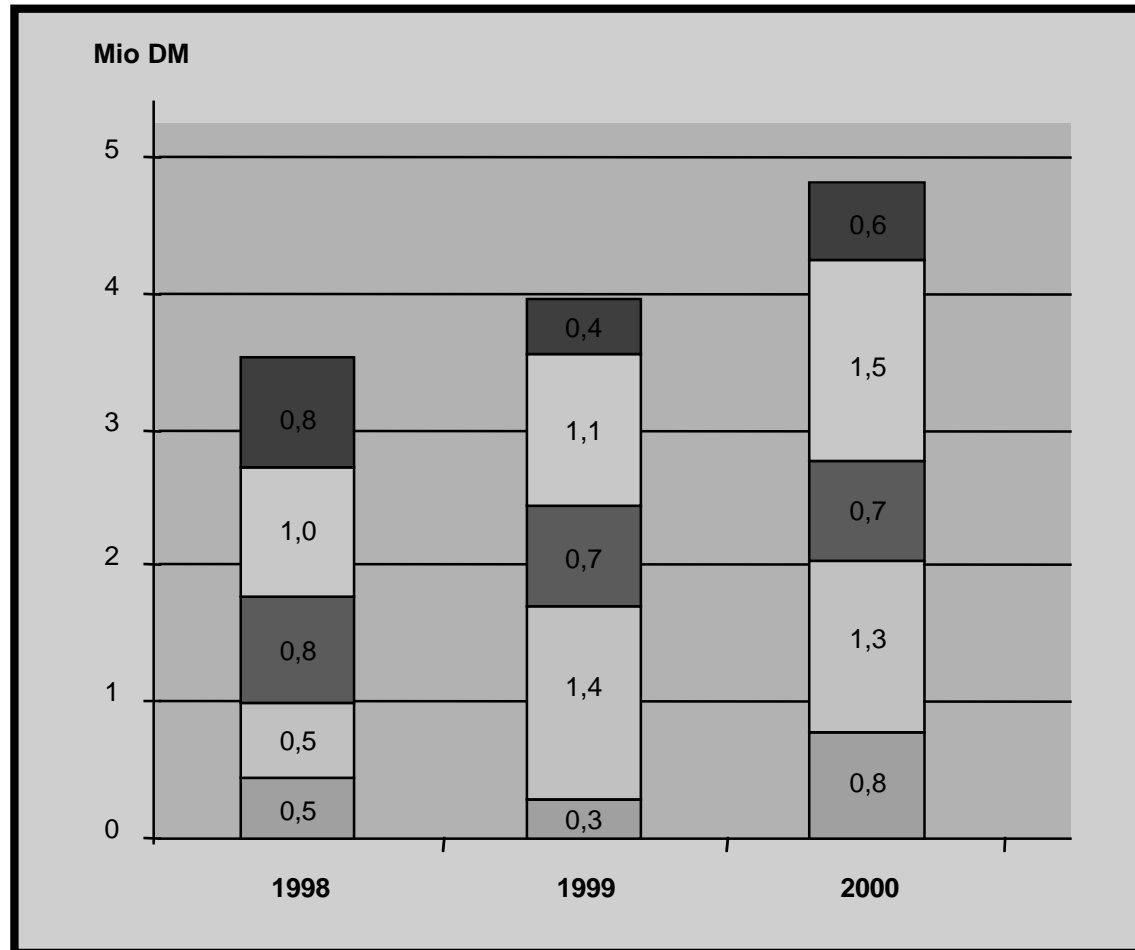
Copyright: GMD-AiS, Sankt Augustin  
Stand: November 2000



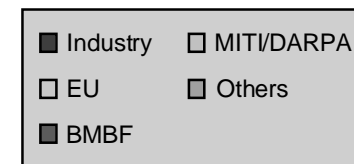
# Institute for Autonomous Intelligent Systems



# External Funding

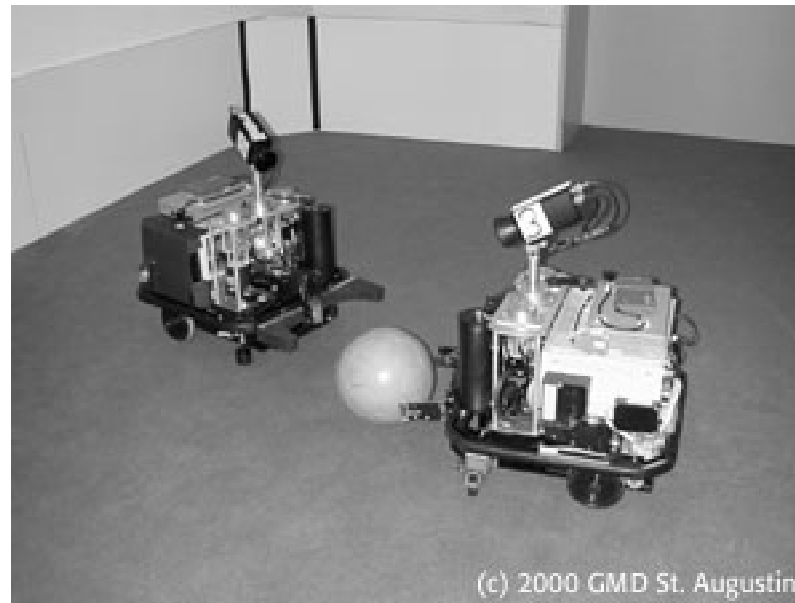


Sources of External Funding



# A Rapid Prototyping Environment for Fast Moving Robots Based Upon Dual Dynamics

Thomas Christaller





# RoboCup



- RoboCup is an international joint project to promote AI, robotics, and related fields.
- It is an attempt to foster AI and intelligent robotics research by providing a standard problem where a wide range of technologies can be integrated and examined.
- Since 1997 annual international competition.
- Today more than 2,500 scientists from 30 countries.

WWW-Address of RoboCup Initiative: <http://www.robocup.org>

## GMD-Robots

Our robot  
must do just  
one thing:

Kick a ball  
into the  
right  
direction.



GMD-Robot



# Outline

- Objectives
- Dual Dynamics
  - behaviors
  - architecture
  - design environment
- Discussion
- Current and future work

# Objectives

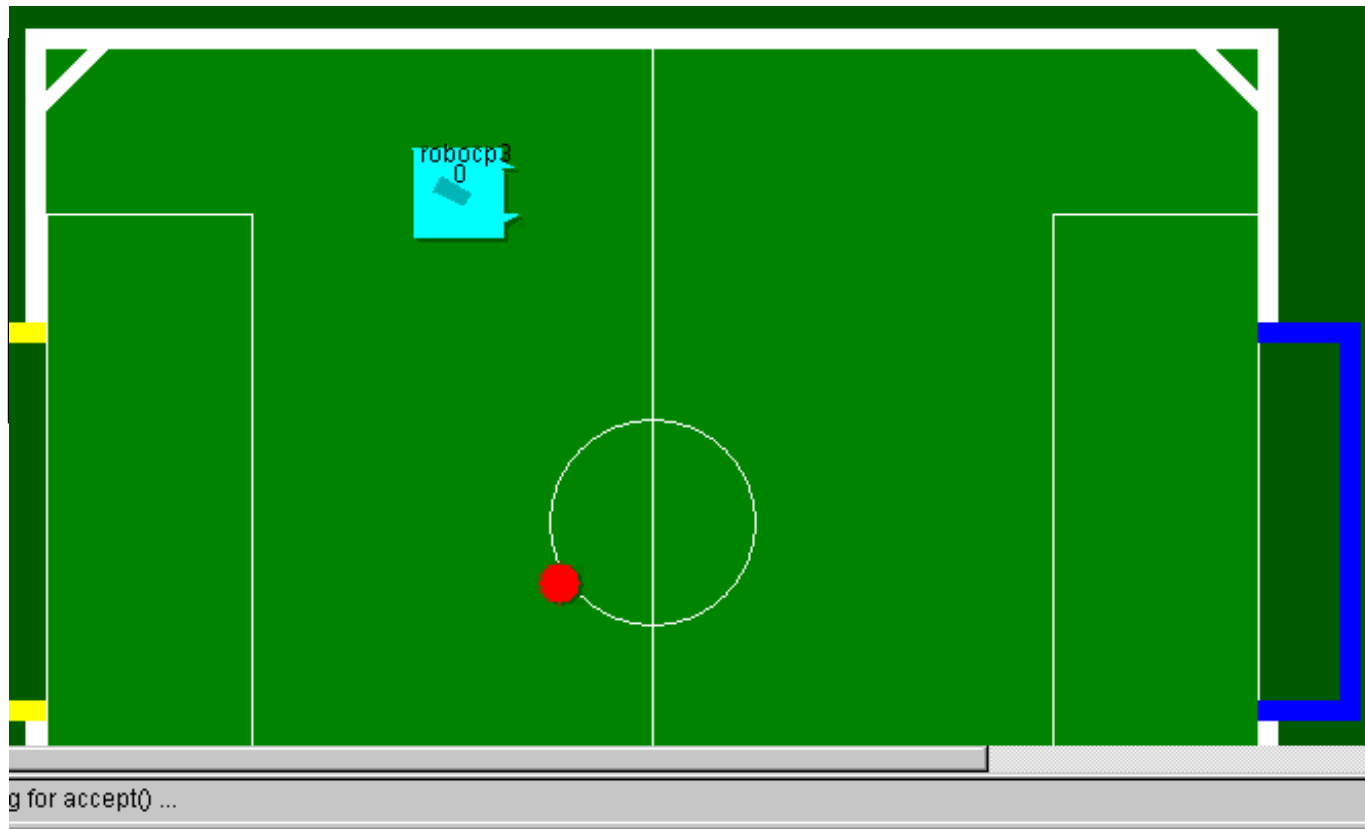
- Combine speed with behavioural complexity
- Rapid design of complex behaviour systems
- RoboCup robots are used as demonstrators

## Fundamental challenge

- *Combine*
- modularity of behaviors, representations, controllers, hardware components,
- *with*
- dynamic unity of resulting action.



*simulated  
GMD-robot*



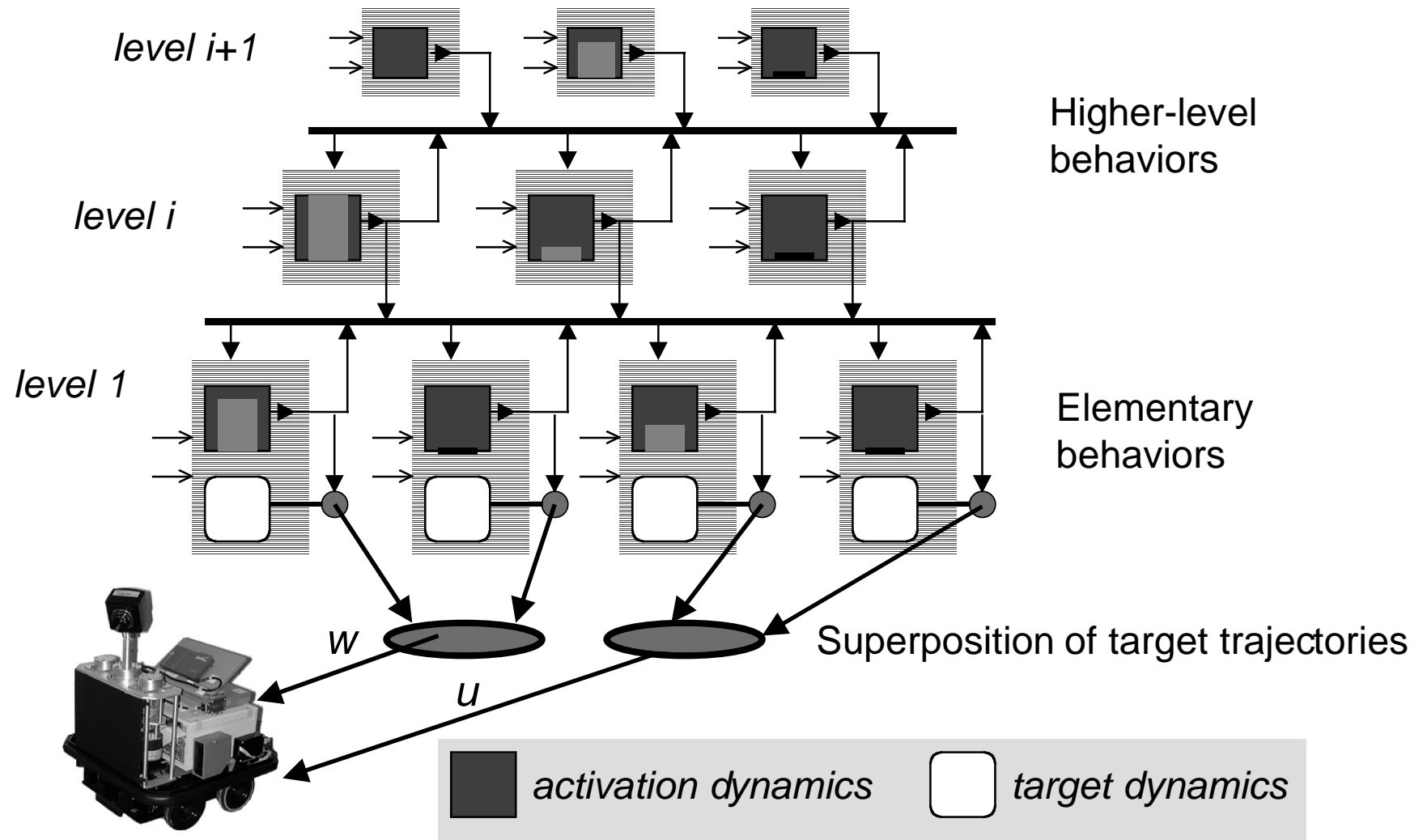
Trace padua3 Robot No.: 0	
BehindBall	
MoveToGoal	
NoSelfGoal	
TurnToBall	

*real-time  
trace of  
behavior  
activations*

# Dual Dynamics

- Roots
  - behaviour-based approach to robotics
  - dynamical systems theory
- Basic idea
  - situated agent operates in modes
  - change of modes are controlled by dynamical systems
  - modes ~ control parameters for self-activation dynamics of behaviours
- Herbert Jaeger, Christian Verbeek, Thomas Christaller

# Dual Dynamics Architecture



# Dual Dynamics Behaviours

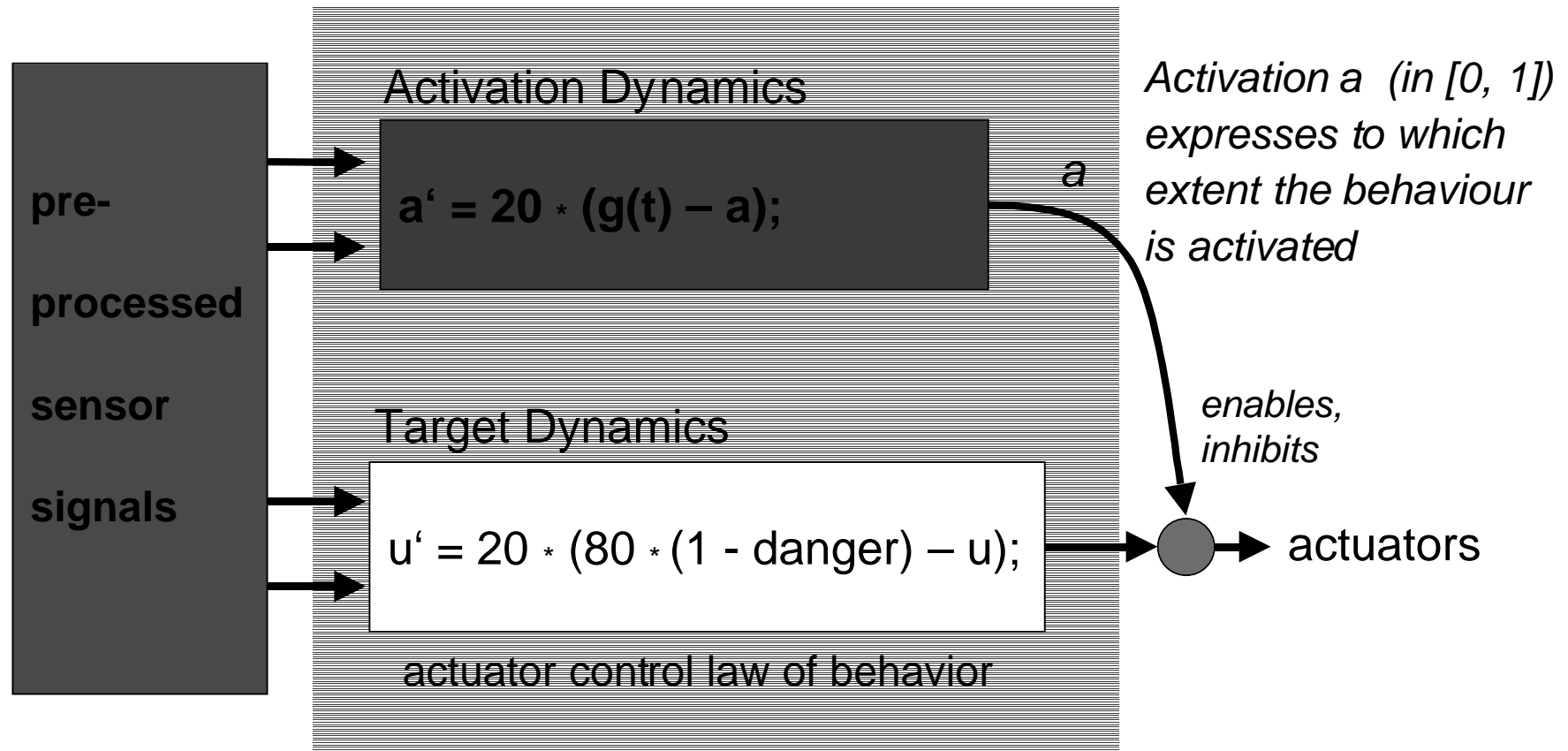
- Behaviors
  - are dynamical systems with an activation dynamics
- Higher-level behaviours:
  - their activations induce bifurcations in lower level activation dynamics („modes“)
  - are behaviours operating on slower time scale
- Elementary behaviours:
  - are behaviours on lowest level of hierarchy
  - they define actuator control law (target dynamics)

**Dual  
Dynamics**

# Activation and Target Dynamics

- Activation Dynamics
  - depend on (pre-processed) sensor input
  - are hand-designed dynamical laws for mode changes
  - are controlled by higher-level activations
  - behaviors are not „called“, they self-activate
- Target Dynamics
  - depend on (pre-processed) sensor input
  - can be any actuator control law
  - generate target trajectories for actuators
  - target trajectories of all behaviors are superimposed

# Elementary Behaviours



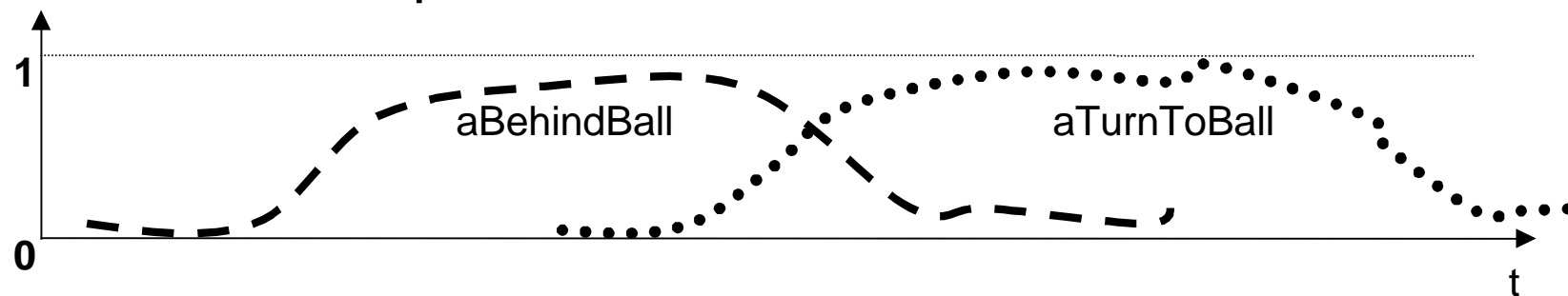
Elementary Behaviour

# Activation Dynamics

- differential equation controls activation

$$a_{\text{BehindBall}}' = 20 * (g(t) - a_{\text{BehindBall}});$$

- activation outputs of behaviours vs. time



- continuous activations smoothly blend behaviours

# Templates for Activation Dynamics

Each behavior activation dynamics has

**// a triggering condition**

**onForce = 20 \* onFlag \* (1 - offFlag);**

**// a disabling condition**

**offForce = 10 \* offFlag;**

**time constant < 40**

in case the control loop runs at 40 Hertz

**// a template differential equation**

**stability = 1;**

**bias = 0.5;**

**decay = 3 \* (1 - aChall1);**

**aBehindBall' = - stability \* aBehindBall**

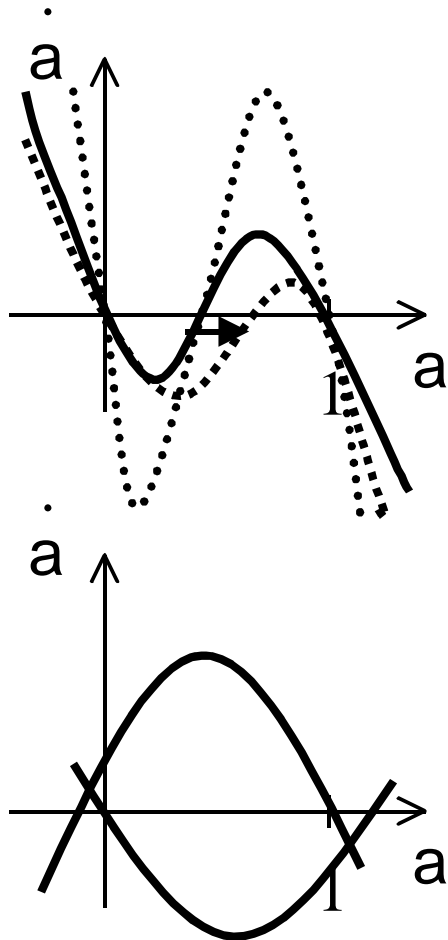
**\* (aBehindBall - 1) \* (aBehindBall + 1 - bias)**

**- onForce \* (aBehindBall + 0.1) \* (aBehindBall - 1)**

**+ offForce \* (aBehindBall - 1.1) \* aBehindBall**

**- decay \* aBehindBall;**

## Cube ODE for bistable, smooth switching of activation



- basic form: cubic ODE  $\dot{a} = f(a)$
- bias toward 0
- greater stability
- Superimpose on-force, off-force  $\dot{a} = g(a)$ , quadratic ODEs
- final form:  $\dot{a} = f(a) + g(a)$

# Target Dynamics

- may be any actuator control law
  - as simple as setting one or more actuator values

```
// drive ahead  
u = 20;  
w = 0;
```

*u is linear velocity of the robot*  
*w is angular velocity of the robot*

```
// accelerate  
u' = 0.1;
```

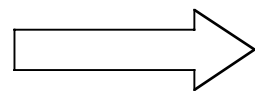
- as complex as a two-dimensional controller as used by the goalie of the GMD-Robots

# Target Dynamics of Goalie

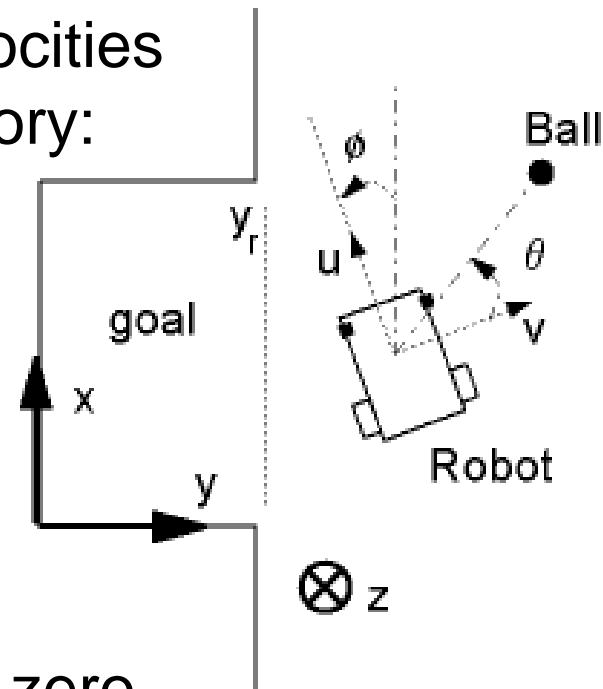
- A nonlinear controller is designed for the linear ( $u$ ) and angular ( $w$ ) velocities based on Lyapunov's stability theory:

$$\omega = -h\phi - (y - y_r)u \frac{\sin \phi}{\phi} : h > 0 \text{ (gain)}$$

$$u = k \sin \theta : k > 0 \text{ (gain)}$$



$(y - y_r)$  and  $\phi$  are globally stabilized to zero



## Target Dynamics of Goalie

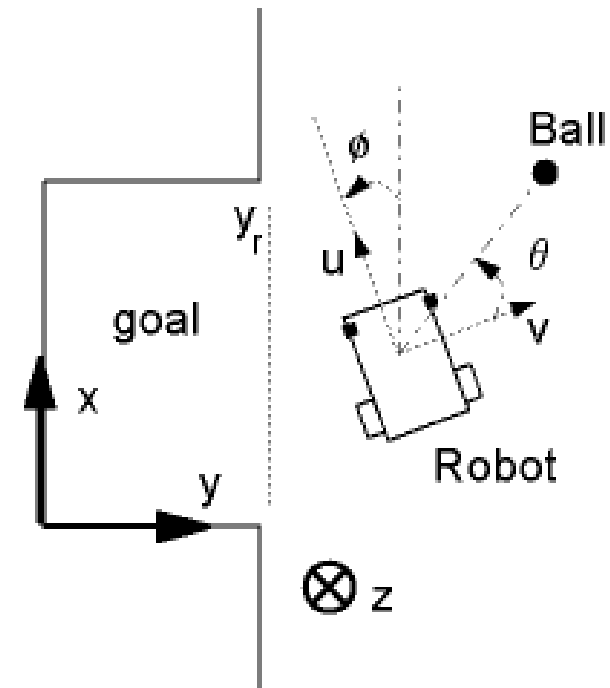
$$\begin{cases} \dot{x} = u \cos \phi \\ \dot{y} = u \sin \phi \\ \dot{\phi} = \omega \end{cases} \quad \begin{cases} V = \frac{1}{2}(\phi^2 + (y - y_r)^2) \text{ (Lyapunov function)} \\ \dot{V} = \phi\omega + (y - y_r)u \sin \phi \end{cases}$$

if  $\omega = -h\phi - (y - y_r)u \frac{\sin \phi}{\phi}$  then

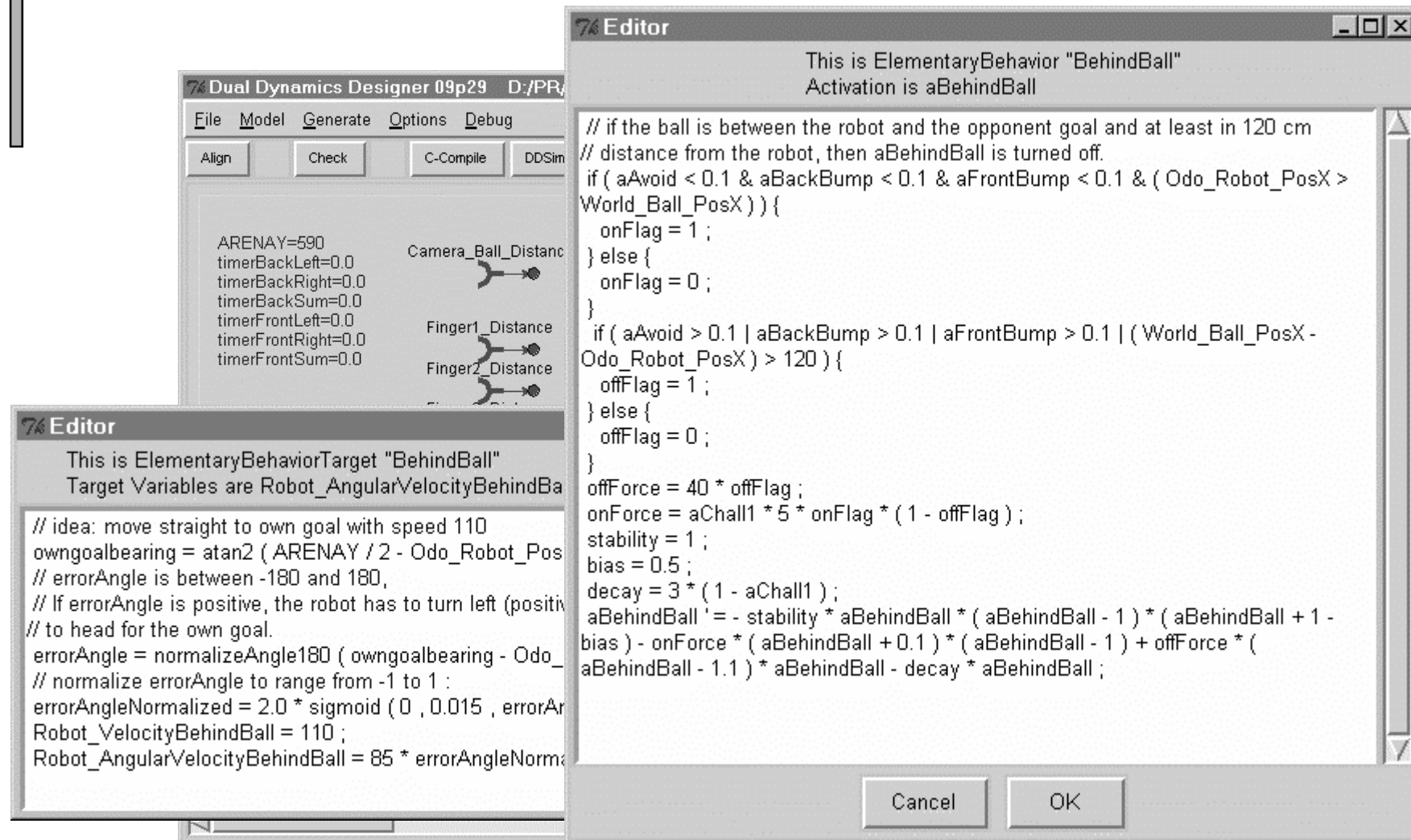
$\dot{V} = -h\phi^2$  is seminegative definite and

by LaSalle's Invariance Principle

$\phi$  and  $(y - y_r)$  converge to zero.



# Dual Dynamics Designer



The image shows the Dual Dynamics Designer software interface. The main window displays a menu bar (File, Model, Generate, Options, Debug) and a toolbar (Align, Check, C-Compile, DDSim). Below the toolbar, there are several numerical parameters and three distance measurement indicators labeled Camera\_Ball\_Distance, Finger1\_Distance, and Finger2\_Distance.

Two 'Editor' windows are open, showing code for elementary behaviors:

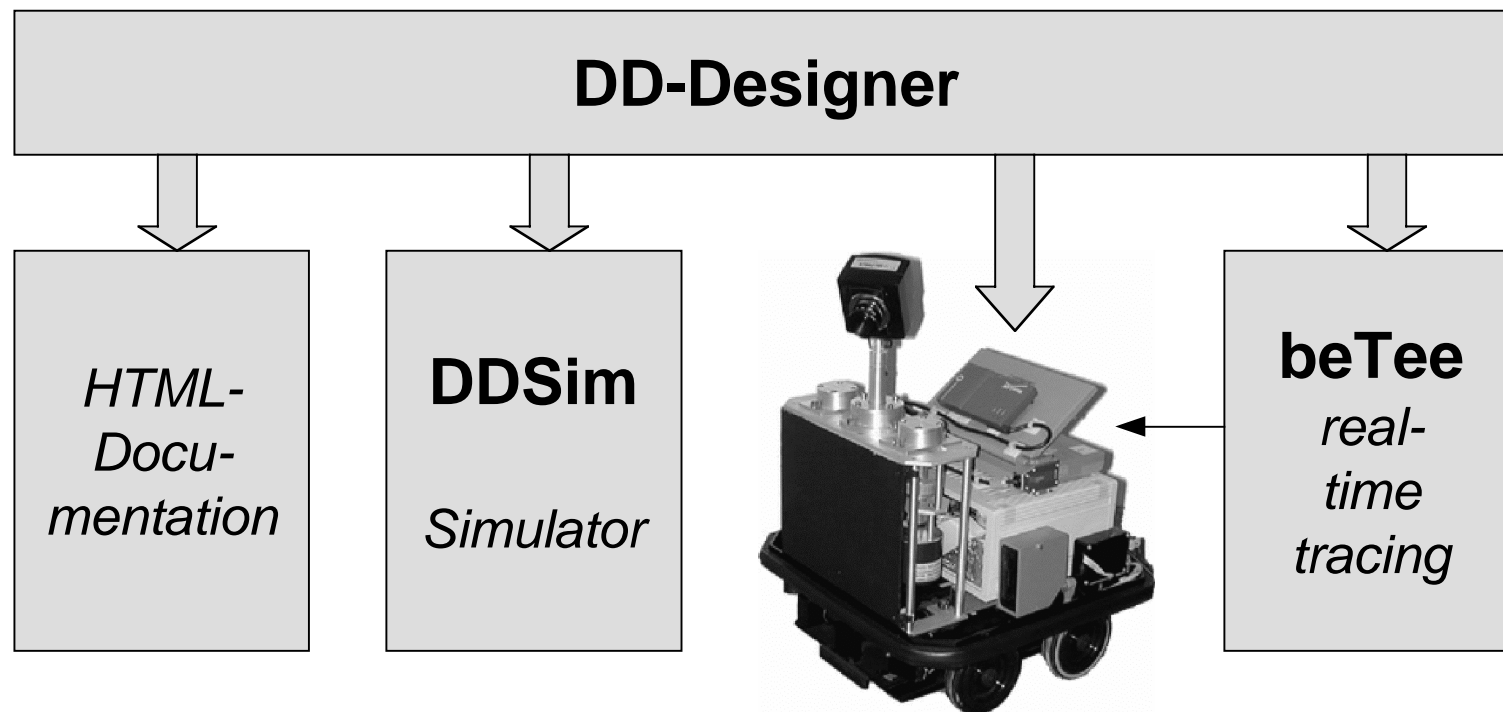
**Editor 1: This is ElementaryBehaviorTarget "BehindBall"**  
 Target Variables are Robot\_AngularVelocityBehindBall

```
// idea: move straight to own goal with speed 110
owngoalbearing = atan2 ( ARENAY / 2 - Odo_Robot_PosX
// errorAngle is between -180 and 180,
// If errorAngle is positive, the robot has to turn left (positiv
// to head for the own goal.
errorAngle = normalizeAngle180 ( owngoalbearing - Odo_
// normalize errorAngle to range from -1 to 1 :
errorAngleNormalized = 2.0 * sigmoid ( 0 , 0.015 , errorAr
Robot_VelocityBehindBall = 110 ;
Robot_AngularVelocityBehindBall = 85 * errorAngleNorma
```

**Editor 2: This is ElementaryBehavior "BehindBall"**  
 Activation is aBehindBall

```
// if the ball is between the robot and the opponent goal and at least in 120 cm
// distance from the robot, then aBehindBall is turned off.
if ( aAvoid < 0.1 & aBackBump < 0.1 & aFrontBump < 0.1 & ( Odo_Robot_PosX >
World_Ball_PosX ) ) {
  onFlag = 1 ;
} else {
  onFlag = 0 ;
}
if ( aAvoid > 0.1 | aBackBump > 0.1 | aFrontBump > 0.1 | ( World_Ball_PosX -
Odo_Robot_PosX ) > 120 ) {
  offFlag = 1 ;
} else {
  offFlag = 0 ;
}
offForce = 40 * offFlag ;
onForce = aChall1 * 5 * onFlag * ( 1 - offFlag ) ;
stability = 1 ;
bias = 0.5 ;
decay = 3 * ( 1 - aChall1 ) ;
aBehindBall = - stability * aBehindBall * ( aBehindBall - 1 ) * ( aBehindBall + 1 -
bias ) - onForce * ( aBehindBall + 0.1 ) * ( aBehindBall - 1 ) + offForce * (
aBehindBall - 1.1 ) * aBehindBall - decay * aBehindBall ;
```

# Dual Dynamics Design Environment

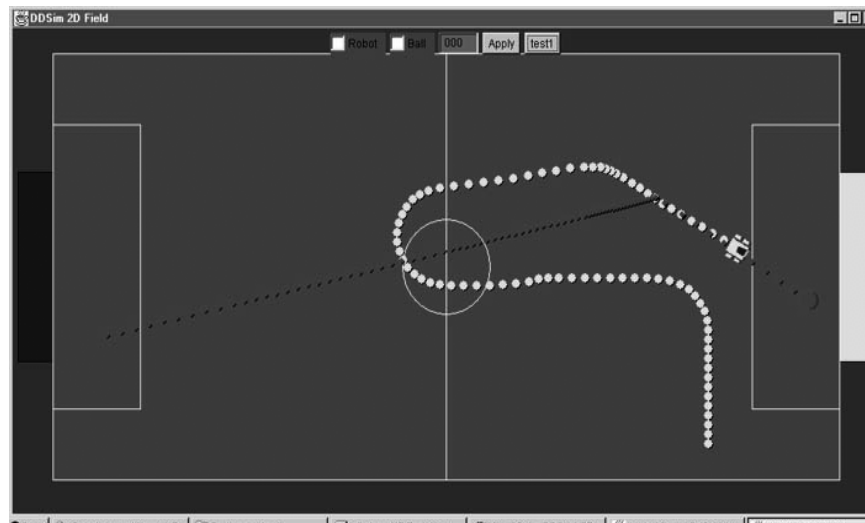


document ➔ simulate ➔ execute ➔ test

- Bredenfeld, Kobialka, Schöll

## Kick a moving ball: a case study

- First task of RoboCup-99 qualification: single robot must find resting ball in empty arena and kick it into goal
- We made it a bit more difficult: ball may move



Applet Viewer: DdSi... [-] [□] [X]

Applet

Start  

Config

2D Hide

Trace Off

Reset

Quit

DDSim  
April 13, 1999

Position

BehindBall

Avoid

Whirl

TurnToBall

NoSelfGoal

Kick

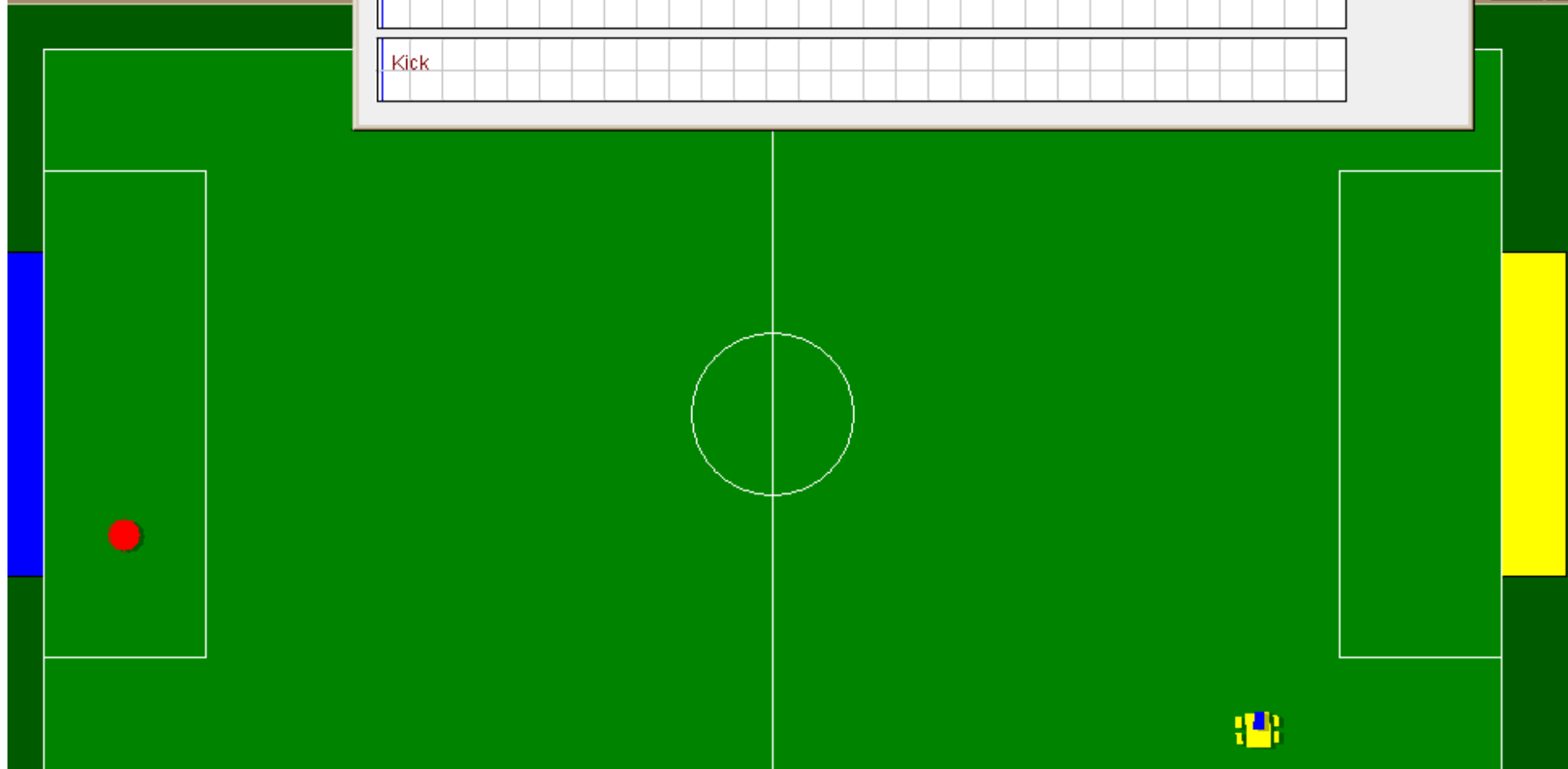
Up

Down

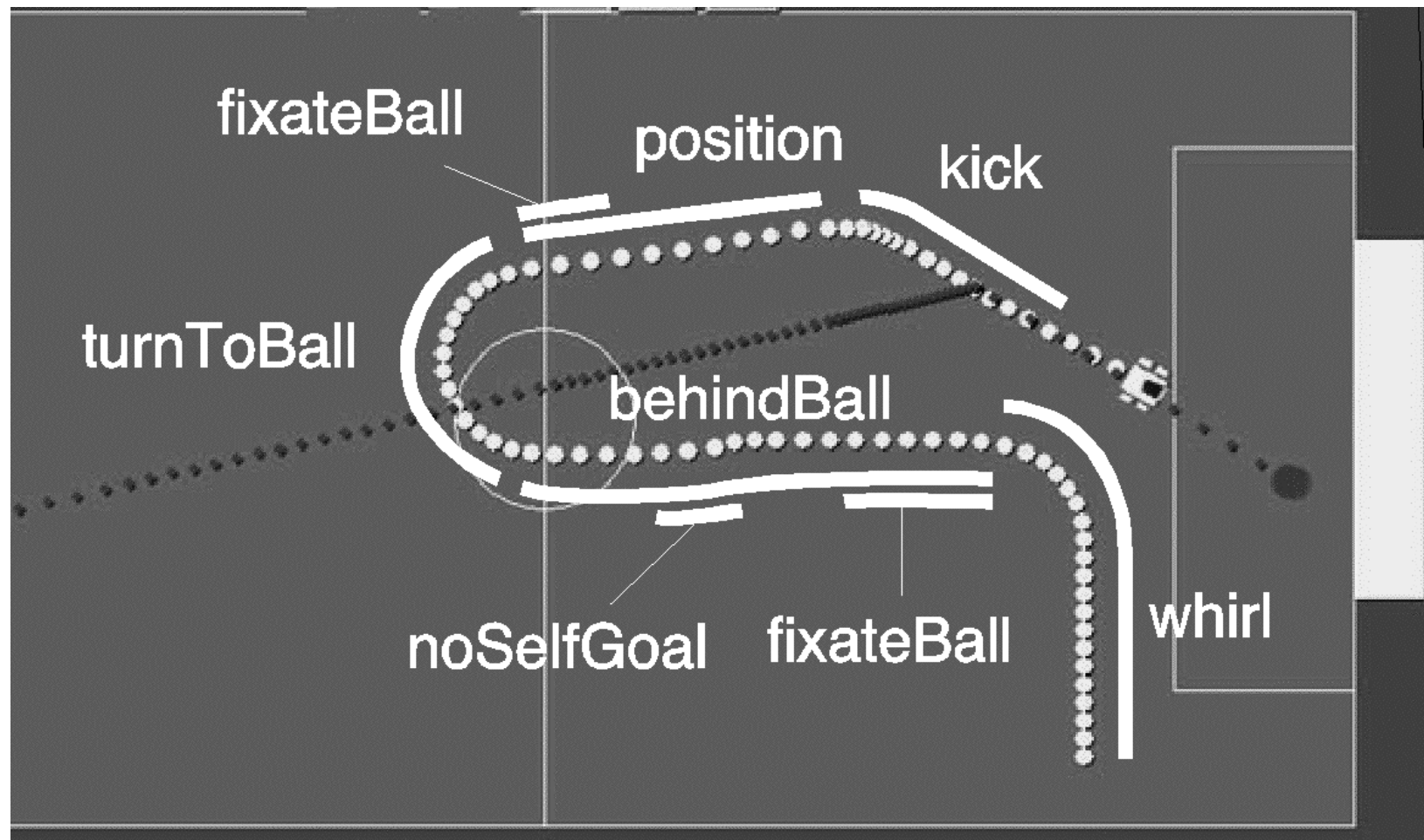
Play

Step

DDSim 2D Field



# RoboCup Behavior System



# Real Experiment



The screenshot displays a real experiment interface with three main windows:

- Field:** A green soccer field with a red ball and a robot (labeled '3') near the center. A blue goal is on the left and a yellow goal is on the right.
- Trace 3. Robot:** A window showing a list of variables being traced over time. The variables include: Camera\_BallVisible, Finger1\_Distance, Finger3\_Distance, Finger4\_Distance, Robot\_AngularVelocitySensed, aAvoid, aBehindBall, aGoHome, aKick, aNoSelfGoal, aPosition, aTurnToBall, and Robot\_AngularVelocity. Each variable has a corresponding colored horizontal bar indicating its active period.
- Configure 3. Robot:** A window for configuring the robot's behavior. It features two lists: "traced Variables" and "Not traced". The "traced Variables" list includes: BlueGoalHeighth, YellowGoalHeighth, Camera\_BallVisible, Camera\_CurrentVideoL, Finger1\_Distance, Finger2\_Distance, Finger3\_Distance, Finger4\_Distance, Robot\_AngularVel, aAvoid, aBehindBall, aGoHome, aKick, aNoSelfGoal, aPosition, and aTurnToBall. The "Not traced" list includes: BlockWaitTimer, RequiredLeftSpeed, RequiredRightSpeed, time, errt/sec, and time\_raster. There are navigation buttons (<<, >>, clear) and a "Start at" field set to 0.543.

An "Analyze Tr..." window is also visible, showing a file path of C:/temp/traceDD.tt and a "Start at" field set to 0.543.



# Discussion



- Pros
  - behaviors extremely robust
  - integrated design environment
  - simulated behaviors directly run on robot
  - architecture open for hybrid controllers
- Cons
  - procedural control flow has to be „emulated“ in DD
  - debugging difficult, but our debug tools help a lot
  - parameters have to be tuned in real experiments, but parameters to be tuned are known in DD

## Current and Future Work

- Behaviors
  - templates for DD activation dynamics
  - complex superposition of target trajectories
  - automatic system identification with OOMs
- Design Environment
  - adaptive sensor simulation models
  - DD-Designer generator for Matlab
- Robots
  - optical flow sensors for fast obstacle avoidance
- DFG Schwerpunktprogramm starting in 2001

## Conclusion

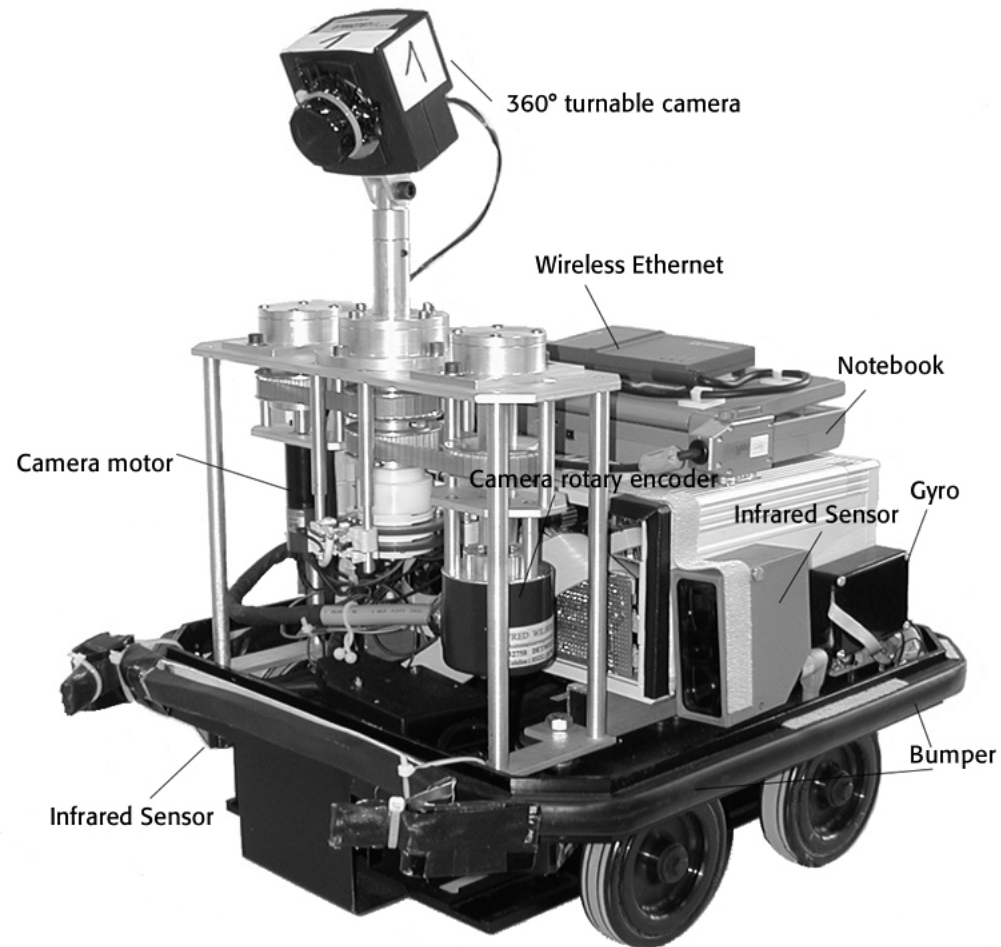
- Challenge: implement many, fast behaviors yet achieve „dynamic unity“
- Approach: transparent mathematical model + single, comprehensive design tool
- DD model rests on dynamical systems theory. Dynamic unity of behavior captured through modes
- DDD tool rests on single internal representation, from which simulation code, robot code, and web documentation is derived

Wissenschaftsfestival 9/2000  
„Stein der Weisen“, Bonn  
GMD vs. University Ulm



# GMD-Robots

<http://ais.gmd.de/BE>



## Research question: standardizing dynamics

- *Goal: identify standard types of activation dynamics and provide equation templates. Three types presently found sufficient.*
- Steep-flanked, instantaneous onset/offset of 0-1-valued activation with inhibition of other behaviours
- (example: *bumpRetract*)



- Slow (exponential) onset/offset of essentially 0-1-valued activation with inhibition of other behaviours
- (example: *whirl*)



- Activation freely varying within 0-1 interval, superimposed on other behaviours

