ETH

HOCHSCHULE LUZERN

Roboscoop!

Project presentation
Spiez, 13 March 2013

Partners

ETH Zurich, Chair of Software Engineering

Chair of Software Engineering

Hochschule Luzern, iHome Lab

i Home Lab
HOCHSCHULE LUZERN

ETH Zurich, Autonomous Systems Lab

2

Roboscoop project members

Chair of Software Engineering

- Bertrand Meyer
- Benjamin Morandi
- Sebastian Nanz
- Andrey Rusakov
- Jiwon Shin

i|Home|Lab HOCHSCHULE LUZERN

- Alexander Klapproth
- Dieter von Arx
- Martin Biallas
- Rolf Kistler
- Marcel Mathis
- Andreas Rumsch

3

Goals

Develop framework for **concurrent** robotics programming

Produce prototype of SmartWalker for elderly people

Learn about software engineering for robotics

4

Achievements so far

First version of Roboscoop framework

First version of SmartWalker

Setting up a course to teach this stuff!

5

Software architecture

Roboscoop

SCOOP

ROS

Integration with other frameworks, external calls

O-O structure, coordination, concurrency (wheels, navigation system, GUI, etc.)

Serial communication with hardware
Coordinate systems
Image processing
Navigation
…

6

SCOOP

Simple Concurrent Object-Oriented Programming

- Easy parallelization
- One more keyword in Eiffel (**separate**)
- Coordination is easy to express: close correspondence with behavioral specification
- Natural addition to O-O framework
- Retains natural modes of reasoning about programs

7

SmartWalker

Smart assistant for elderly people

Hi-tech extension of the regular walker

Autonomous robot with sensors and actuators

Possible functionalities:

- Support while going uphill/downhill
- Navigation during shopping
- Finding a charging station
- Fall detection
- …

8

SmartWalker

9

SmartWalker: Hardware

Single-board computer (BeagleBone)

- Low-cost
- Credit-card-sized
- 720MHz ARM processor
- Operating system: Linux computer
- Connectivity: 2 x 46 pin headers

10

More about the hardware

Motor controller: controls motors up to 350 W

- Controls force of motors according to given voltage & direction signal
- Optimized for controlling e-bike motors

Motor:

- Integrated Hall sensor to determine position
- Accumulator
- 36 V / 12 Ah with electronic management

11

SmartWalker: Software

Single-board computer

- Measures 2D position
- Speed control loop for each wheel
- Controls wheels concurrently

Tablet PC:

- User interface
- High-level control
- Roboscoop

Communication over ROS

12

Roboscoop

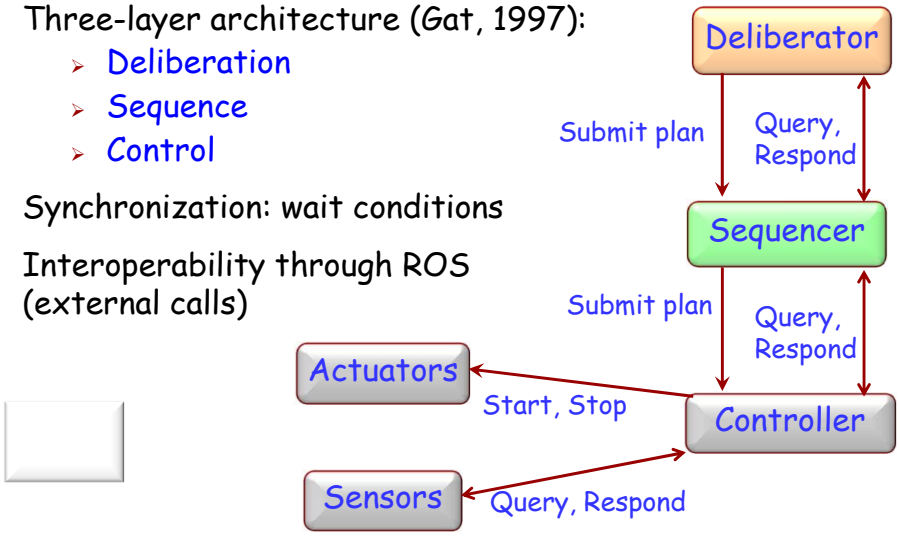
Coordination layer above SCOOP (and hence Eiffel)

Three-layer architecture (Gat, 1997):

- Deliberation
- Sequence
- Control

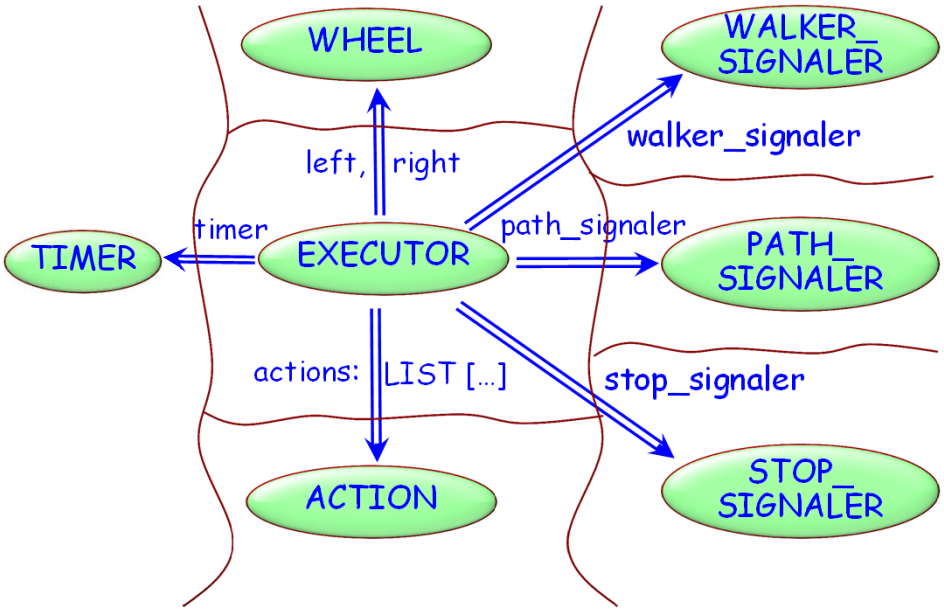
Synchronization: wait conditions

Interoperability through ROS (external calls)



13

Object & processor architecture



14

SCOOP: separate calls ("embarrassingly parallel")

```
walker_signaler: separate WALKER_SIGNALER
                                    -- From sensors: position, orientation…,
stop_signaler: separate STOP_SIGNALER
path_signaler: separate PATH_SIGNALER
actions: LIST [separate ACTION]

start_path (left, right: separate WHEEL)
        -- Perform sequence given by actions.
    local
        i: INTEGER
    do
        across actions as a until
                stop_requested (stop_signaler)
        loop
                execute (a, path_signaler, stop_signaler, left, right)
                wait (a, path_signaler, stop_signaler, left, right)
        end
    end
```

15

SCOOP: synchronization through preconditions

```
to_next  (a: ACTION
          left, right: separate WHEEL
          ps: separate PATH_SIGNALER
          ss: separate STOP_SIGNALER
          ws: separate WALKER_SIGNALER)
            --Unless stop requested, complete a and enable next action.
    require
        ss.stop_requested or (ps.state = a.index and a.done (ws))
    do
        left.stop
        right.stop
        if not ss.stop_requested then
                ps.set_state (a.index + 1)
        end
    end
```

16

SCOOP: wait by necessity

```
path: LIST [separate ACTION]
executor: EXECUTOR
        -- To obtain actions from a script:
path := parser.item (script)

        -- To execute sequence of actions:
across path as p loop add_action (executor, p.item) end

add_action (e: separate EXECUTOR; a: separate ACTION)
          --Add a to action executor.
      local
        s: BOOLEAN
      do
        e.add_action (a)
        s := a.done                    -- The order matters!
      end
```

17

Application to teaching

New interdisciplinary course at ETH, Fall 2013:

Robot Programming Laboratory

Open to CS and ME students

Combines software engineering, concurrency & robotics:

- How software engineering applies to robotics
- Main architectures, coordination, synchronization methods
- Experience in programming a small robotic system
- Sensing, planning and control
- ROS and Roboscoop

18

Things we learned

Concurrency is great for robotics

A SmartWalker would be truly useful

This stuff is tough

19

Roboscoop: what's next?

Gain more experience through course

Continue enhancing the Roboscoop concurrency framework

Add sensors to SmartWalker!

Implement SmartWalker scenarios

Evaluate applicability to other robots

Perform evaluation of SCOOP for robotics

20