Orocos Introduction
Open Robot Control Software

Peter Soetens
Flanders’ Mechatronics Technology Centre
Leuven

1 October 2007
FMTC, Leuven
Outline

1. Introduction
2. Approach
3. Example Application
Section Outline

1. Introduction
   - Examples

2. Approach
   - The Component Model
   - Communication Categories

3. Example Application
Orocos in one-liners

The *Real-Time Toolkit* (RTT):

- *Open Robot Control Software*
  ⇒ *Open Source* ’robot’ control and interfacing
- Real-time Software Toolkits in C++
  ⇒ Developer’s tool
- Tool for developing components for control
  ⇒ Real-time, thread-safe, interactive
- Offers common component implementations
  ⇒ Optional

Freely available on:
http://www.orocos.org
Outline

1 Introduction
   • Examples

2 Approach
   • The Component Model
   • Communication Categories

3 Example Application
Communication and Behaviour

Continuous control: tracking a light source.
Continuous and discrete control: Placing a car window
In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- access the hardware devices
- create components which do all this.
In these examples, Orocos was used to

- do the real-time communications
- define the real-time *behaviour* of machines in response to communication
- access the hardware devices
- create components which do all this.
In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- access the hardware devices
- create components which do all this.
In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- access the hardware devices
- create components which do all this.
1 Introduction
   • Examples

2 Approach
   • The Component Model
   • Communication Categories

3 Example Application
Outline

1 Introduction
   • Examples

2 Approach
   • The Component Model
   • Communication Categories

3 Example Application
A Component Model for Control

Approach

- Create a software component for each ‘task’ within the machine

Control Components
Component Definition

Real-Time Behaviour

Communication Defined by the component interface

Machine

Middleware for Machine Control

Real-Time Communication

Behaviour Defined by real-time state machines
Component Definition

- **Real-Time Behaviour**
  - S1
  - S2
  - S3
  - S4

- **Machine**

- **Communication**
  - Defined by the component interface

- **Middleware for Machine Control**

- **Real-Time Communication**

- **Behaviour**
  - Defined by real-time state machines
Work-flow

Component Model
Real-Time Toolkit to build components

Components
Re-usable part of an application

Applications
‘Deployments’ select and connect Components

Control Components
Build Applications
Control Applications
Build Components

Orocos Real-Time Toolkit
C++ Classes

Real-Time Communication
Real-Time State Machines
Work-flow

Component Model
Real-Time Toolkit to build components

Components
Re-usable part of an application

Applications
‘Deployments’ select and connect Components

Control Components
Build Applications

Control Applications

Build Components

Orocos Real-Time Toolkit C++ Classes

Real-Time State Machines
Real-Time Communication
Work-flow

Component Model
Real-Time Toolkit to build components

Components
Re-usable part of an application

Applications
‘Deployments’ select and connect Components

Control Components
Build Applications
Control Applications
Build Components
Orocot Real-Time Toolkit
C++ Classes

Real-Time Communication
Real-Time State Machines
Outline

1 Introduction
   • Examples

2 Approach
   • The Component Model
   • Communication Categories

3 Example Application
In which ways can components communicate?

- Configuration of parameters
- Exchange data
- Cooperate to achieve a task
Component Implementation

Component

"Peer" Components

Real-Time Communication

S1
S2
S3
S4

Real-Time State Machines
```
State Controlling_P
{
  double error;
  run {
    set error = Ref.Get() - Ist.Get();
    do Out.Set( K * error );
  }
  exit {
    do Out.Set( 0.0 );
  }
  transitions {
    if ( error > MaxError )
      select SignalTrackingError
  }
}

"P Controller Component"
```
1. Introduction
   • Examples

2. Approach
   • The Component Model
   • Communication Categories

3. Example Application
Example Application

How are these communication primitives used?
Example Application

Camera
Component Interface

Camera
- resolution, refresh rate: properties
- moveCamera(p): command

Image: Data Port
- image ready: event
- p getPosition(): method

Image Recognition
- Image: Data Port
- car color: property
Communication: Configuration

Configuration Flow: Properties

Camera

- resolution, refresh rate: properties
- moveCamera(p): command
- image ready: event
- p getPosition(): method

Image: Data Port

Image Recognition

- Image: Data Port
- car color: property

<xml>

<xml>
Data Flow: Port Connections

Camera

- resolution, refresh rate: properties
- Image: Data Port
- Image ready: event
- p getPosition(): method
- moveCamera( p ): command

Connector

Image Recognition

- Image: Data Port
- car color: property

Data Flow: Ports and Connectors
Communication: Data

Data Flow: Ports and Connectors

Component Activity

Camera Component (a)
Write Data: "Image"

Recognition Component (b)

Read Data: "Image"

Data Flow: Ports and Connectors
Communication: Execution

Execution Flow:
- moveCamera(p): command
- Image: Data Port
- Image: Data Port
- Image Recognition
- car color: property
- Execution Flow: events, methods and commands
Communication: Execution

Execution Flow: Commands

Send Command: "moveCamera"

Component Activity

Recognition Component (a)

Check Completion Condition: atPosition()

Check Queue

Execute Command: moveCamera()

Camera Component (b)

Command Queue

Execution Flow: Commands
Communication: Execution

Execution Flow: Events

Component Activities

Asynchronous Reaction to Event

Synchronising Component

Event Queue

Register Reaction to Event

Published Event: "image ready"

Camera Component

Synchronous Reaction to Event

Time
Communication: Complete Picture

**Data Flow:**
*Port Connections*

- Camera
- Image: Data Port
- Image: Data Port
- connector
- Image Recognition
- car color: property

**Execution Flow:**
*Events, methods and commands*

- moveCamera( p ): command
- p getPosition(): method
- image ready: event
- resolution, refresh rate: properties
The following steps lead to a control application design:

- identification of the ‘control tasks’ → components
- defining each component’s interface
- setting up components connections
- defining component or application behaviours