Flanders'
MECHATRONICS
Technology Centre
www.fmtc.be

Orocos
Open Robot Control Software

Peter Soetens

Flanders' Mechatronics Technology Centre
Leuven

5 July 2006
V Jornades de Programari Lliure
Barcelona

Outline

Section Outline

Orocos in one-liners

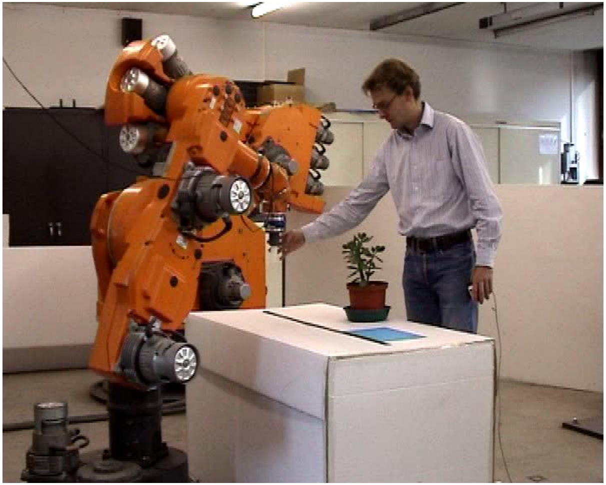
- *Open Robot Control Software*
  $\Rightarrow$ *Open Source* 'robot' control and interfacing
- Real-time Software Toolkits in C++
  $\Rightarrow$ Developer's tool
- Tool for developing components for control
  $\Rightarrow$ Real-time, thread-safe, interactive
- Offers common component implementations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

Freely available on:

http://www.orocos.org

Outline

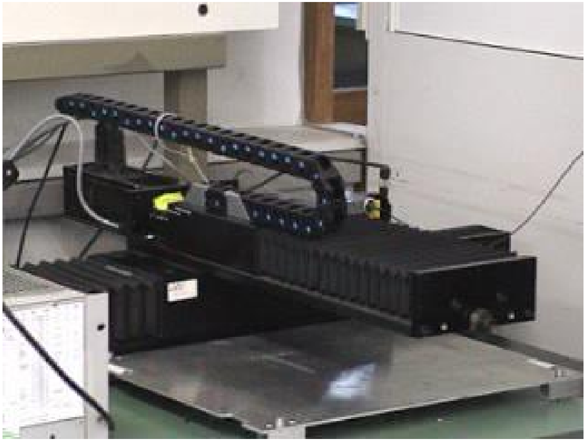
Interaction and Behaviour



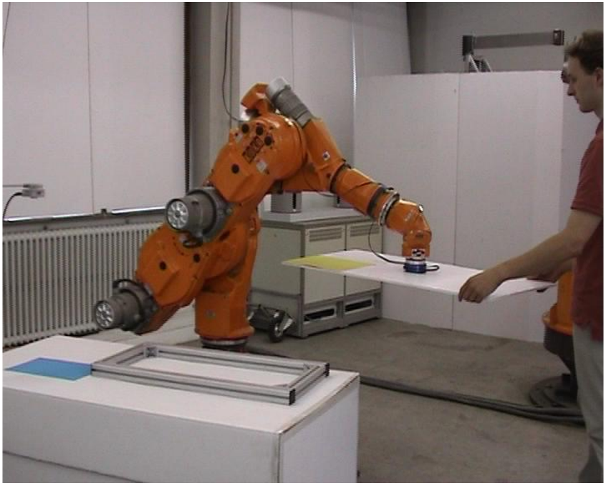
Force control influences behaviour.

Flanders' MECHATRONICS Technology Centre

Interaction and Behaviour


Continuous control: tracking a light source.

Interaction and Behaviour



Continuous and discrete control: Placing a car window

Introduction

In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- calculate real-time kinematics
- access the hardware devices
- create components which do all this.

Introduction

In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- calculate real-time kinematics
- access the hardware devices
- create components which do all this.

Introduction

In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- calculate real-time kinematics
- access the hardware devices
- create components which do all this.

Introduction

In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- calculate real-time kinematics
- access the hardware devices
- create components which do all this.

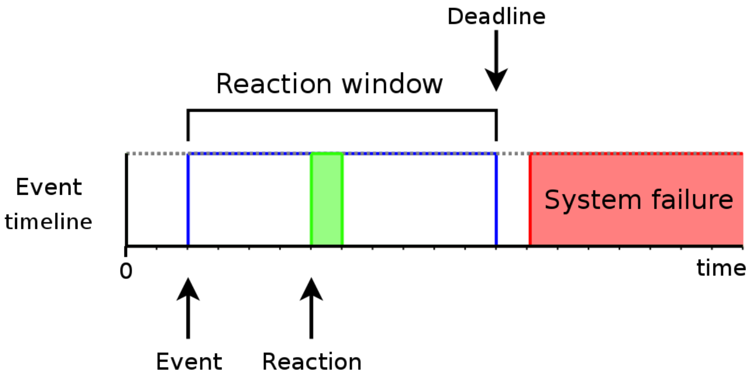
Introduction

In these examples, Orocos was used to

- do the real-time communications
- define the real-time behaviour of machines in response to communication
- calculate real-time kinematics
- access the hardware devices
- create components which do all this.

Outline

What is Real-Time?



- react *always* on time to a given event

Why Real-Time Software?

Consider solving. . .

Robot or machine interaction with the environment

Without guarantees.

- What use is SLAM if your mobile platform bumps into obstacles ?
- What use is a camera if your manipulator crushes your object ?
- What use is controller tuning in MATLAB if the controller fails in practice ?

$\Rightarrow$ They all need real-time control software !

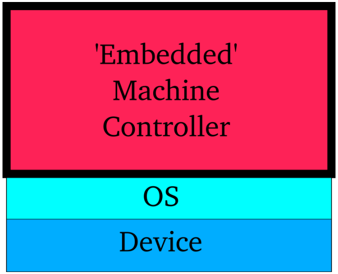
Rapid Software Development

Consider solving...

More hardware ⇒ Much more software

With monolithic software.

- New devices, same problems to solve
- More software and features
- Device connectivity and networking



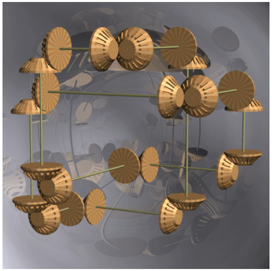
Safe Software Development

Consider solving...

More threads $\Rightarrow$ Much more trouble

With bare threads and locks as tools.

- Deadlocks, thread races, data corruption
- Synchronisation between threads ?
- Communication between threads ?



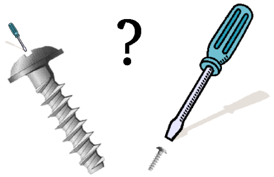
Flexible Software Development

Consider solving…

More layers ⇒ Less control

With closed toolkits.

- ’Solutions’ restrict the solution
- Software interaction ?
- Dead vendor products ?

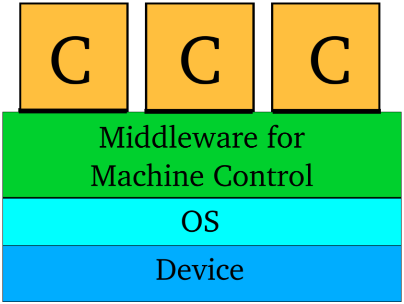


Flanders' MECHATRONICS Technology Centre

Outline

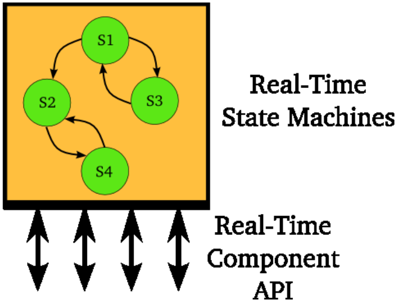
Rapid Software Development

Orocos provides . . .
Middleware for Machine Control
⇒ Software Component deployment *and* interconnection



Safe Software Development

Orocos provides . . .
Tools for Communication ⇒ Thread-safe and Real-Time

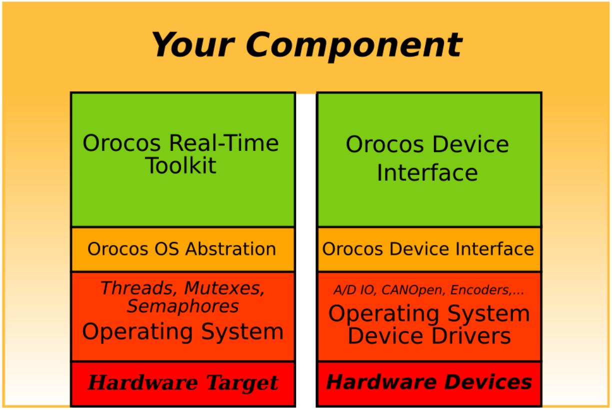


Flanders' MECHATRONICS Technology Centre

Flexible Software Development

KATHOLIEKE UNIVERSITEIT LEUVEN PMA

Orocos is . . .
Free Software ⇒ Open Infrastructure with ∞ lifetime



Orocos Application Stack

Outline

History

- 2001: Started as a ‘small’ research project
  - Founded by Prof H. Bruynickx, KU Leuven
- 2001-2005: Developed during the PhD of Peter Soetens
  - Sponsored by the EU IST “Orocos”, “Ocean” and “Open Machine Controller” projects and FMTC.
- 2005-. . . : Maintained by the FMTC.
  - ‘Modular Machines Group’

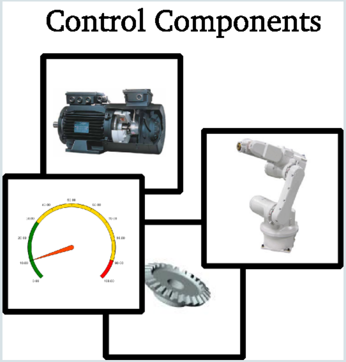
Section Outline

Outline

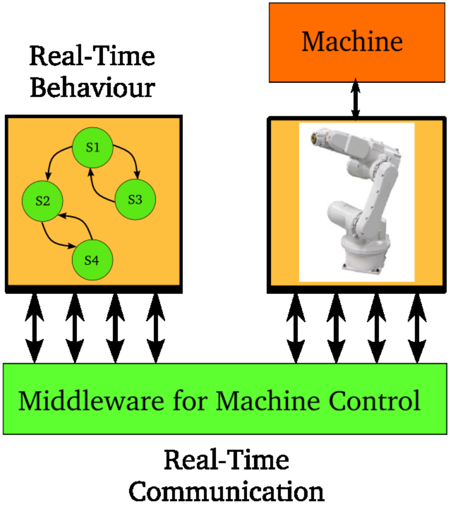
A Software Component Model for Control

Approach

- Create a software component for each 'task' within the machine



Component Definition



Communication

Defined by the component interface

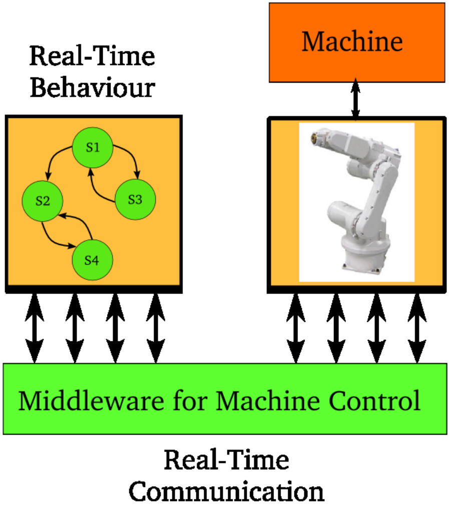
Middleware

Mediates component communication and distribution

Behaviour

Defined by real-time state machines

Component Definition



Communication

Defined by the component interface

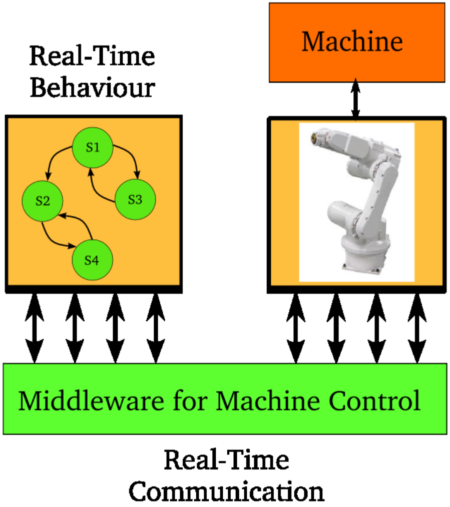
Middleware

Mediates component communication and distribution

Behaviour

Defined by real-time state machines

Component Definition



Communication

Defined by the component interface

Middleware

Mediates component communication and distribution

Behaviour

Defined by real-time state machines

Definitions

What is a Software Component ?

Definition

A modular and replaceable part of a system that encapsulates implementation . . . and exposes a set of interfaces.

What is a Component Model?

Definition

A framework for describing components . . . with the purpose for creating software from re-usable software components.

Definitions

What is a Software Component ?

Definition

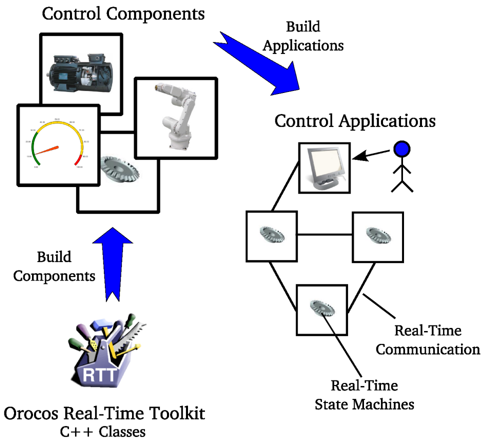
A modular and replaceable part of a system that encapsulates implementation . . . and exposes a set of interfaces.

What is a Component Model?

Definition

A framework for describing components . . . with the purpose for creating software from re-usable software components.

What is a Component Model?



Component Model

Toolkit to describe Real-Time components

Components

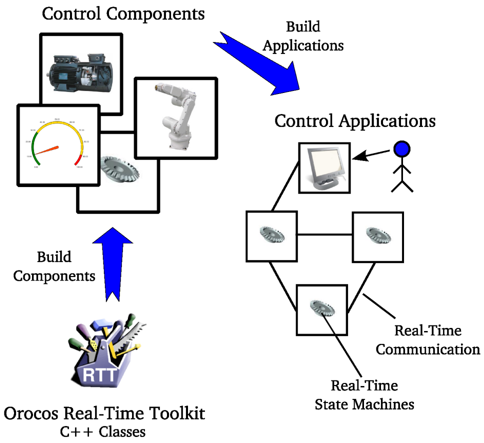
Re-usable part of an application

Applications

'Templates' select and connect Components

Flanders' MECHATRONICS Technology Centre

What is a Component Model?



Component Model

Toolkit to describe Real-Time components

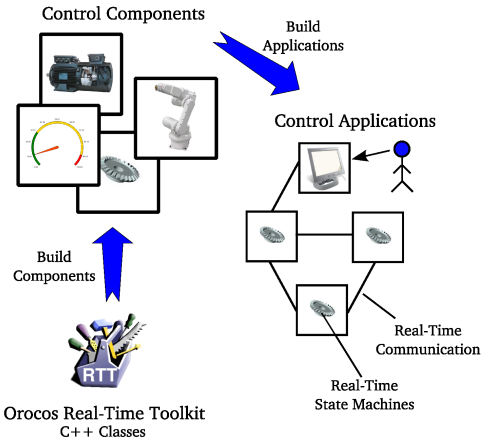
Components

Re-usable part of an application

Applications

'Templates' select and connect Components

What is a Component Model?



Component Model

Toolkit to describe Real-Time components

Components

Re-usable part of an application

Applications

'Templates' select and connect Components

Section Outline

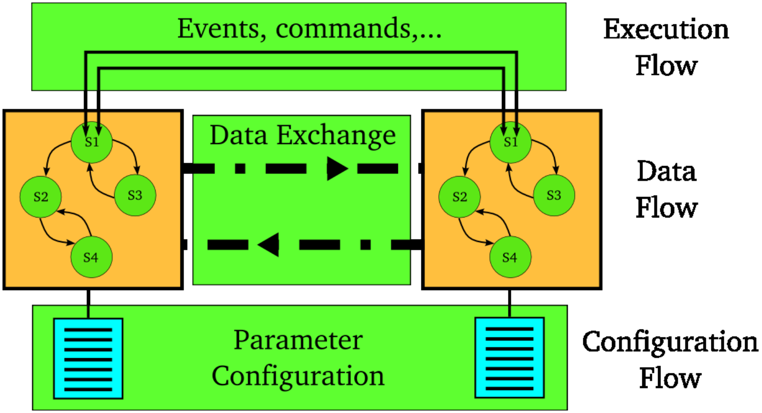
Outline

Component Interaction Patterns

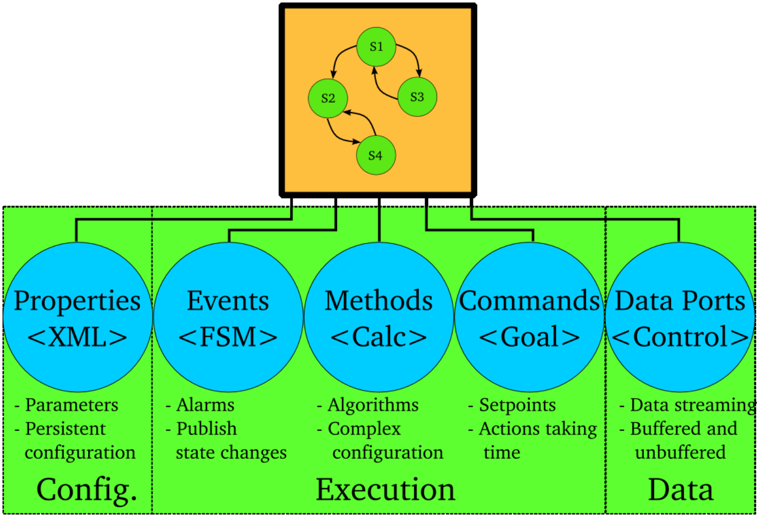
In which ways can components communicate?

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

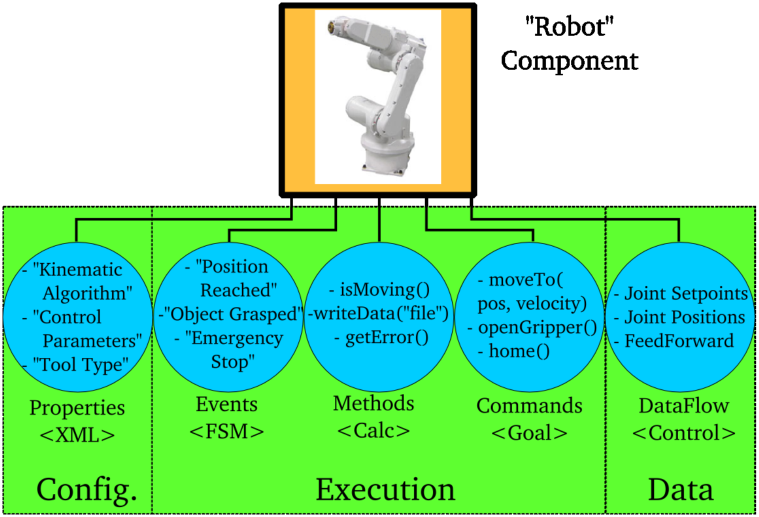
Component Interaction Patterns



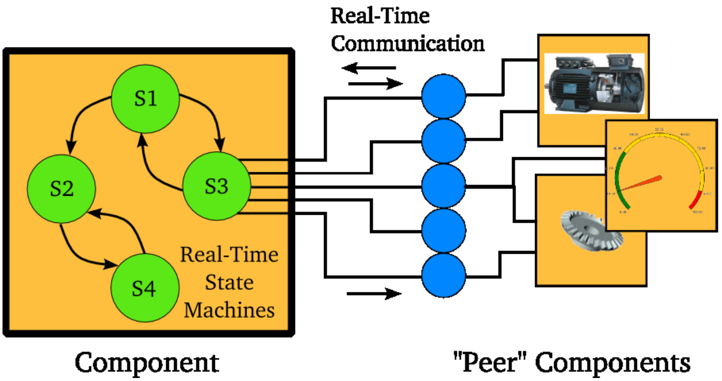
Component Interface



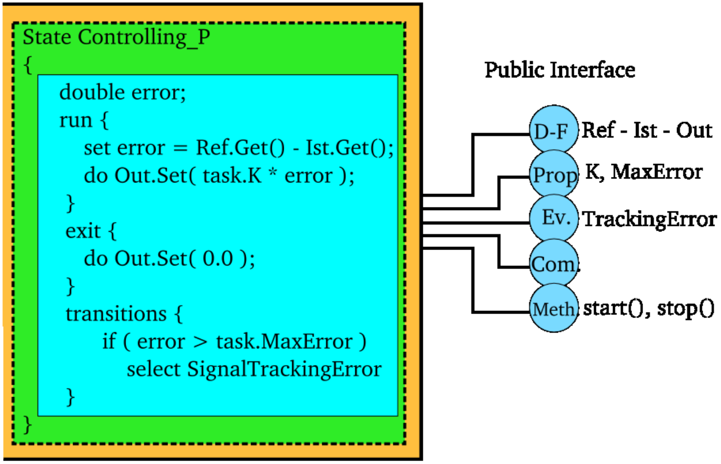
Component Interface



Component Implementation



State Machine Example



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

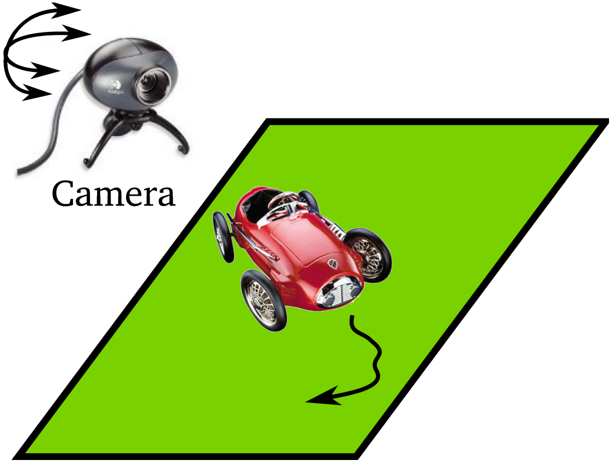
"P Controller Component"

Outline

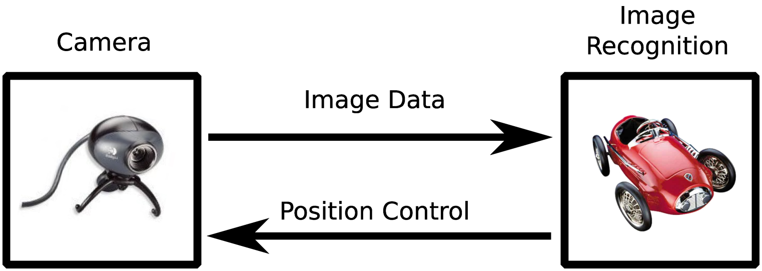
Example Application

How are these communication primitives used ?

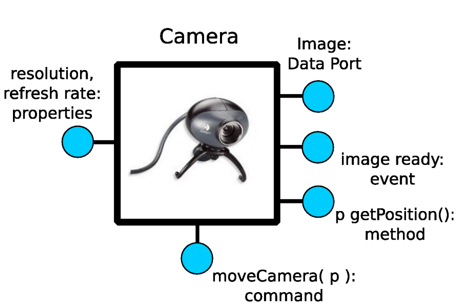
Example Application

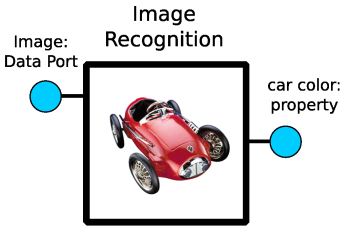


Application Template

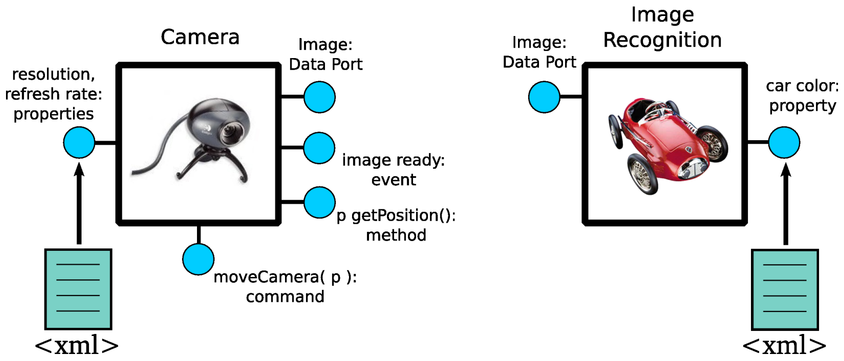


Component Interface



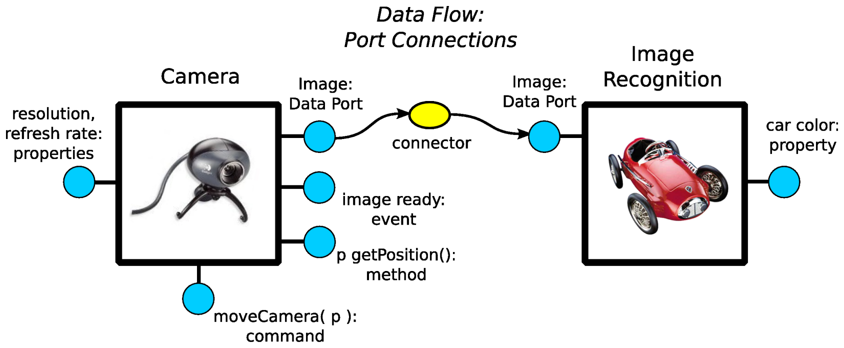


Communication: Configuration



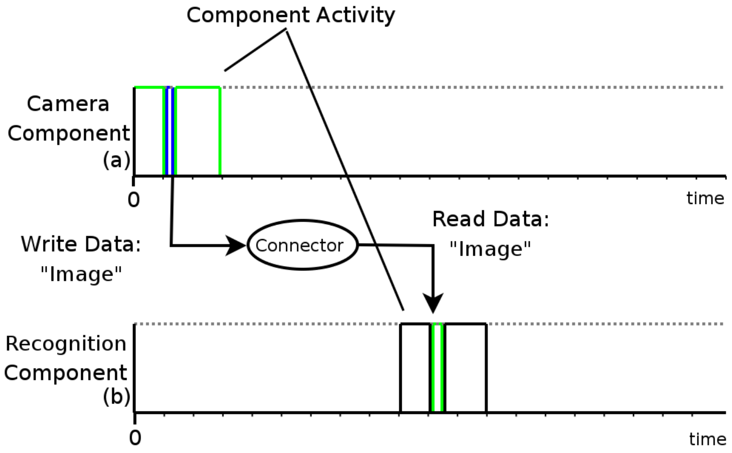
Configuration Flow : Properties

Communication: Data



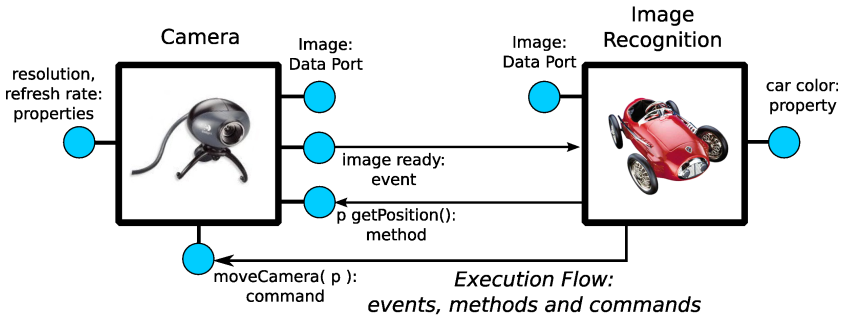
Data Flow : Ports and Connectors

Communication: Data



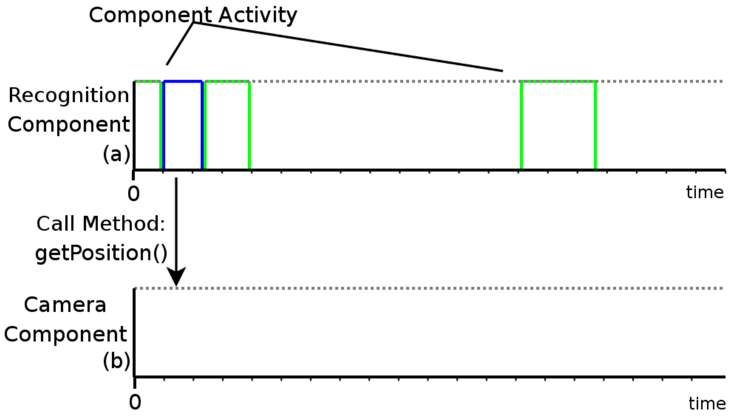
Data Flow : Ports and Connectors

Communication: Execution



Execution Flow

Communication: Execution

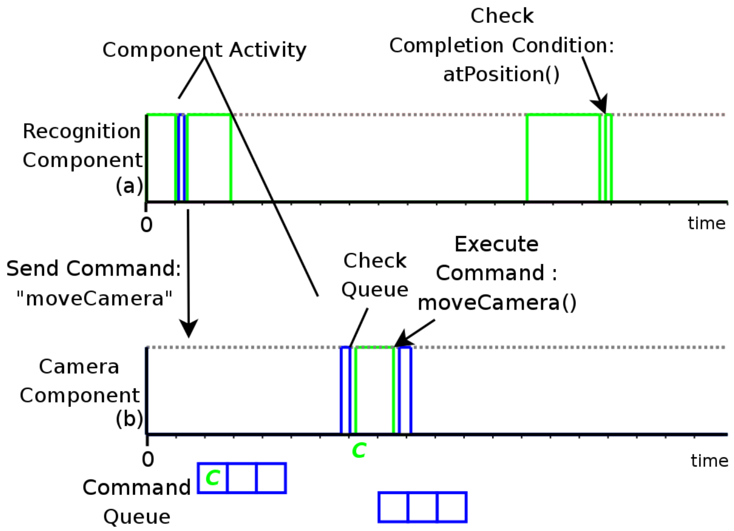


Execution Flow: Methods

Flanders' MECHATRONICS Technology Centre

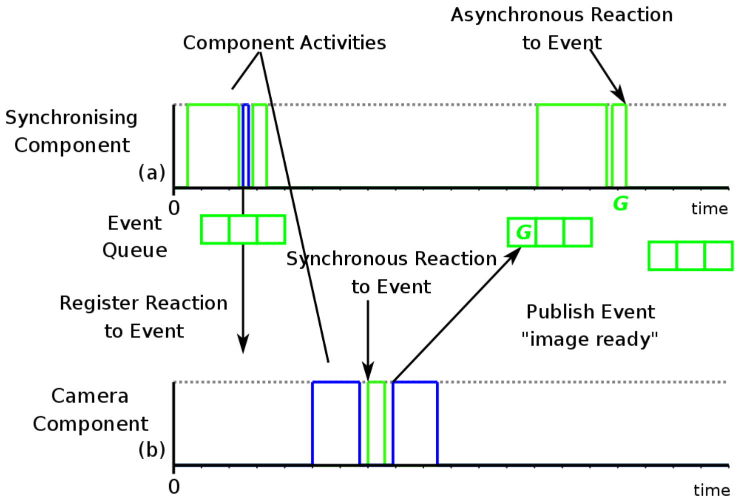
KATHOLIEKE UNIVERSITEIT LEUVEN PMA

Communication: Execution



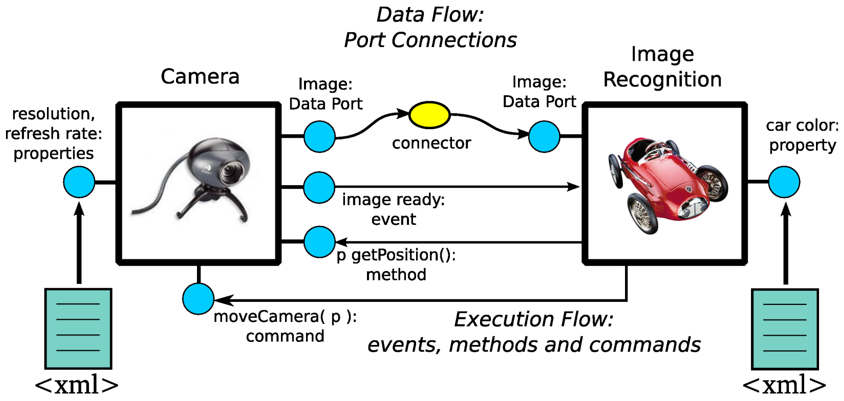
Execution Flow: Commands

Communication: Execution

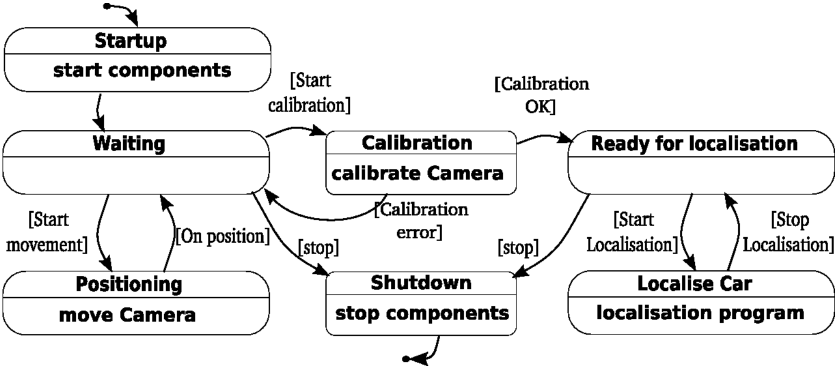

Asynchronous Reaction to Event
Component Activities
Synchronising Component (a)
0
G
time
Event Queue
G
Synchronous Reaction to Event
Register Reaction to Event
Publish Event "image ready"
Camera Component (b)
0
time

Execution Flow: Events

Communication: Complete Picture



Behaviour — State Chart



Example Application Summary

The following steps lead to a control application design:

- identification of the 'control tasks' $\rightarrow$ components
- defining each component's interface
- setting up components connections
- defining component or application behaviours

Outline

The Future of Orocos

Today:

- Feature freeze, focus on usability:
  Components, API, Real-Time Tookit...
- Brand new Kinematics-Dynamics Library (KDL):
  Online this summer.
- Bayesian Filtering Library (BFL)
  `http://people.mech.kuleuven.be/~kgadeyne/bfl.html`

September 2006:

- Orocos 1.0 Release and new web-site

Afterwards;

- Focus on components and kinematics

Conclusion

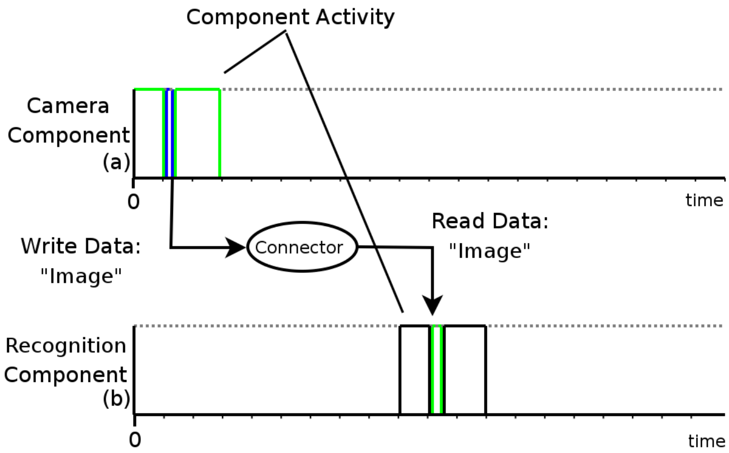
Orocos offers

- a software toolkit for building real-time components
- rich online browsable component interface
- user defined real-time state machines

Further Reference:

http://www.orocos.org

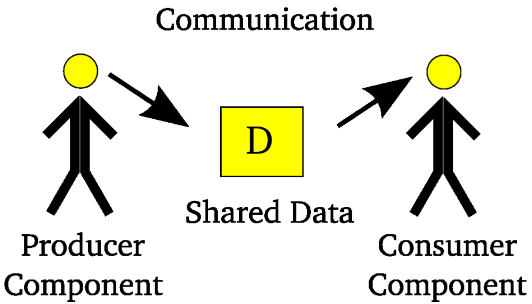
Recap: Data Flow Communication



Measurement results

Measure communication times with

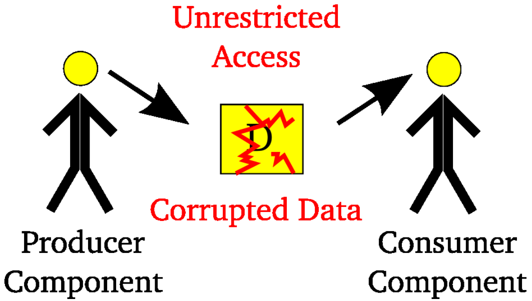
- ideal 'instant' communication
- traditional 'lock-based' communication
- 'lock-free' communication for all communication primitives



Measurement results

Measure communication times with

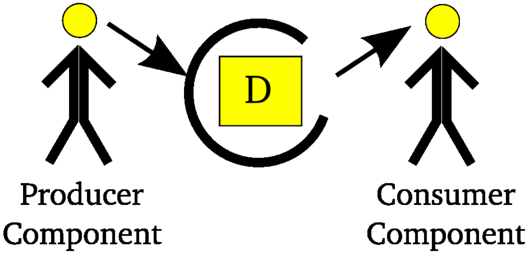
- ideal ‘instant’ communication
- traditional ‘lock-based’ communication
- ‘lock-free’ communication for all communication primitives



Measurement results

Measure communication times with

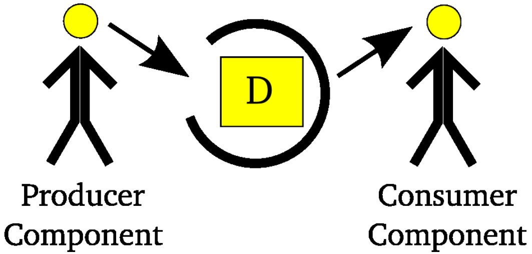
- ideal ‘instant’ communication
- traditional ‘lock-based’ communication
- ‘lock-free’ communication for all communication primitives



Measurement results

Measure communication times with

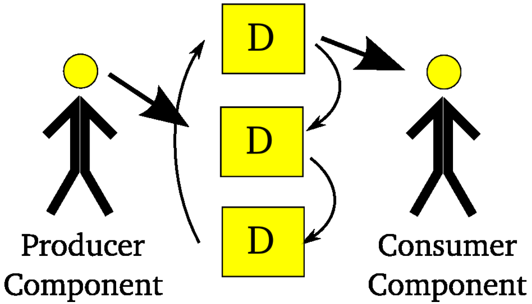
- ideal ‘instant’ communication
- traditional ‘lock-based’ communication
- ‘lock-free’ communication for all communication primitives



Measurement results

Measure communication times with

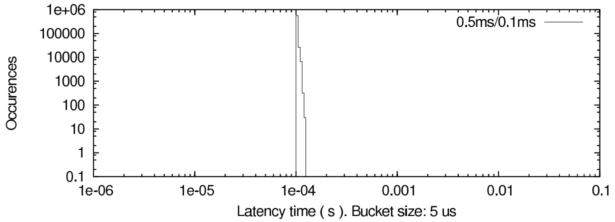
- ideal 'instant' communication
- traditional 'lock-based' communication
- 'lock-free' communication for all communication primitives

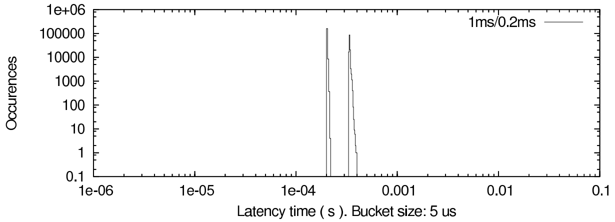


Flanders' MECHATRONICS Technology Centre

KATHOLIEKE UNIVERSITEIT LEUVEN PMA

Ideal 'Instant' Communication

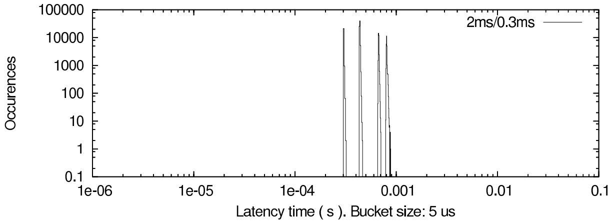




Measured execution latencies: high and low priority.

Flanders' MECHATRONICS Technology Centre

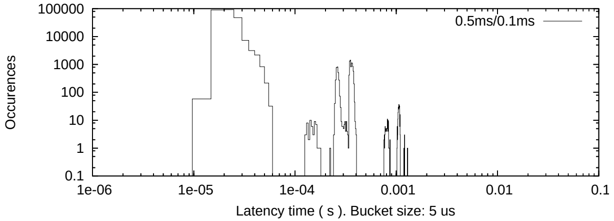
Ideal ‘Instant’ Communication

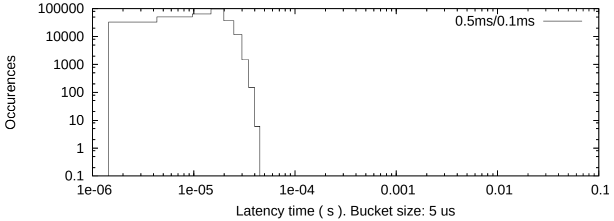


Measured execution latencies: lower priority.

Flanders' MECHATRONICS Technology Centre

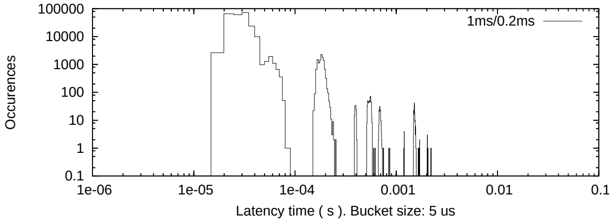
Validation : Data Flow

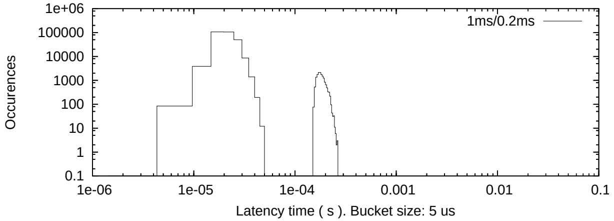




Measured communication latencies: high priority locked and lock-free.

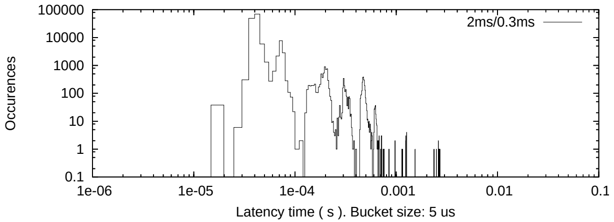
Validation : Data Flow

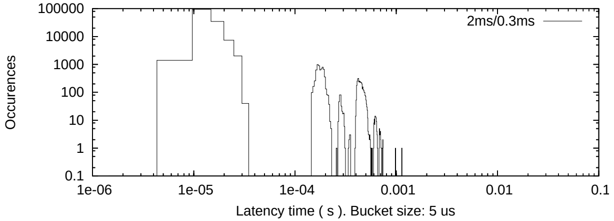




Measured communication latencies: medium priority locked and lock-free.

Validation : Data Flow





Measured communication latencies: low priority locked and lock-free.

Conclusion

Orocos offers

- a software toolkit for building real-time components
- rich online browsable component interface
- user defined real-time state machines

Further Reference:

http://www.orocos.org