Flanders'
MECHATRONICS
Technology Centre
www.fmtc.be

Orocos Introduction
Open Robot Control Software

Peter Soetens

Flanders' Mechatronics Technology Centre
Leuven

1 October 2007
FMTC, Leuven

Outline

Flanders' MECHATRONICS Technology Centre

Section Outline

Orocos in one-liners

The *Real-Time Toolkit* (RTT):

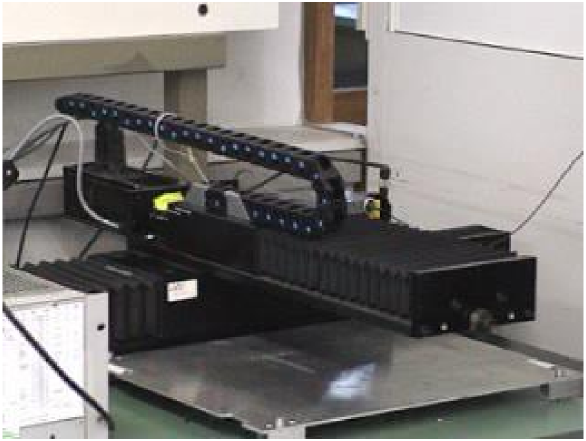
- *Open Robot Control Software*
  $\Rightarrow$ *Open Source* 'robot' control and interfacing
- Real-time Software Toolkits in C++
  $\Rightarrow$ Developer's tool
- Tool for developing components for control
  $\Rightarrow$ Real-time, thread-safe, interactive
- Offers common component implementations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

Freely available on:

http://www.orocos.org

Outline

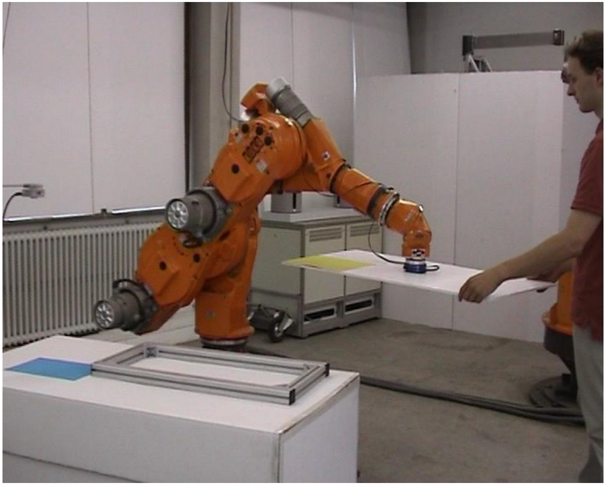
Communication and Behaviour



Continuous control: tracking a light source.

Flanders' MECHATRONICS Technology Centre

Communication and Behaviour


Continuous and discrete control: Placing a car window

Flanders' MECHATRONICS Technology Centre

Introduction

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.

Introduction

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.

Introduction

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.

Introduction

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.

Flanders' MECHATRONICS Technology Centre

Section Outline

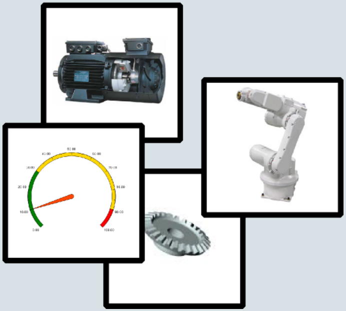
Outline

A Component Model for Control

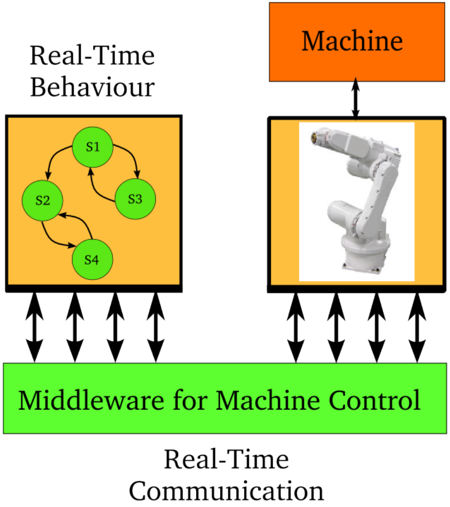
Approach

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

Control Components



Component Definition



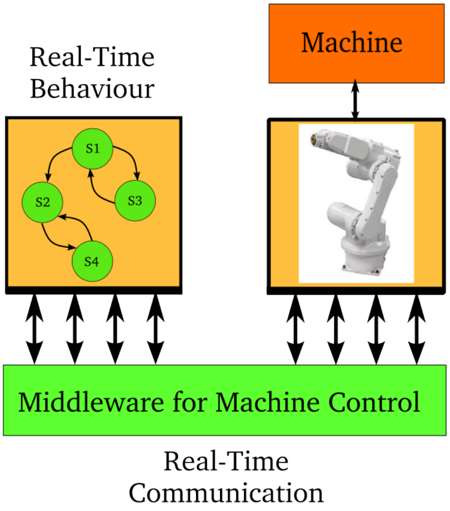
Communication

Defined by the component interface

Behaviour

Defined by real-time state machines

Component Definition



Communication

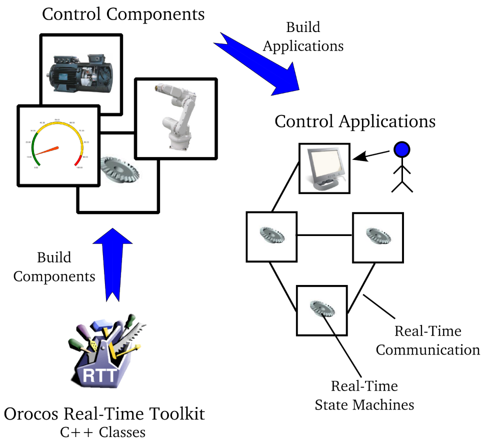
Defined by the component interface

Behaviour

Defined by real-time state machines

Flanders' MECHATRONICS Technology Centre

Work-flow



Component Model

Real-Time Toolkit to build components

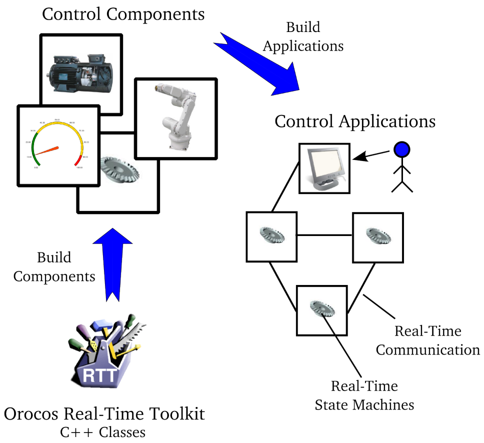
Components

Re-usable part of an application

Applications

'Deployments' select and connect Components

Work-flow



Component Model

Real-Time Toolkit to build components

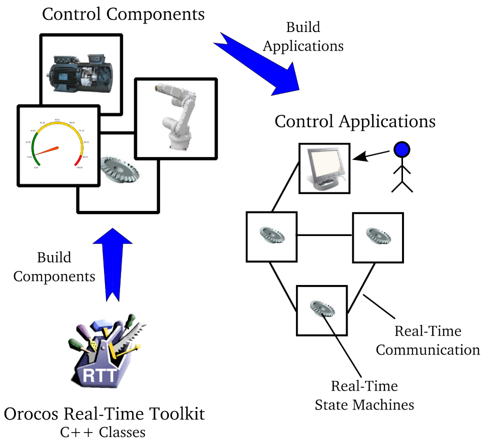
Components

Re-usable part of an application

Applications

'Deployments' select and connect Components

Work-flow



Component Model

Real-Time Toolkit to build components

Components

Re-usable part of an application

Applications

'Deployments' select and connect Components

Flanders' MECHATRONICS Technology Centre

Outline

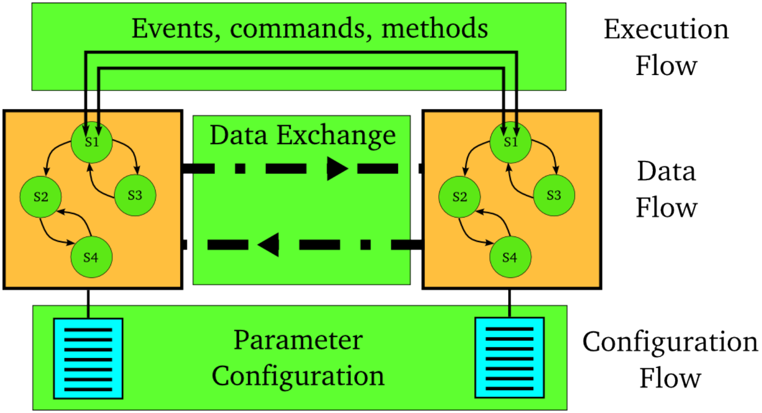
Component Communication Patterns

In which ways can components communicate?

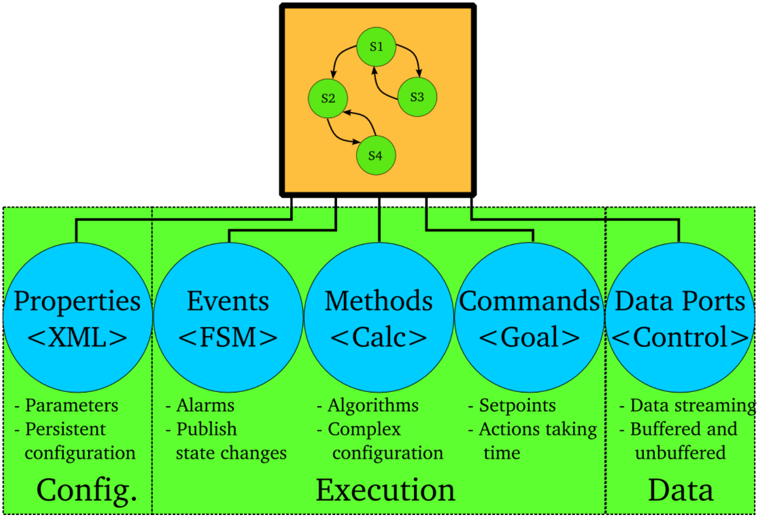
- Configuration of parameters
- Exchange data
- Cooperate to achieve a task

Flanders' MECHATRONICS Technology Centre

Component Communication Patterns

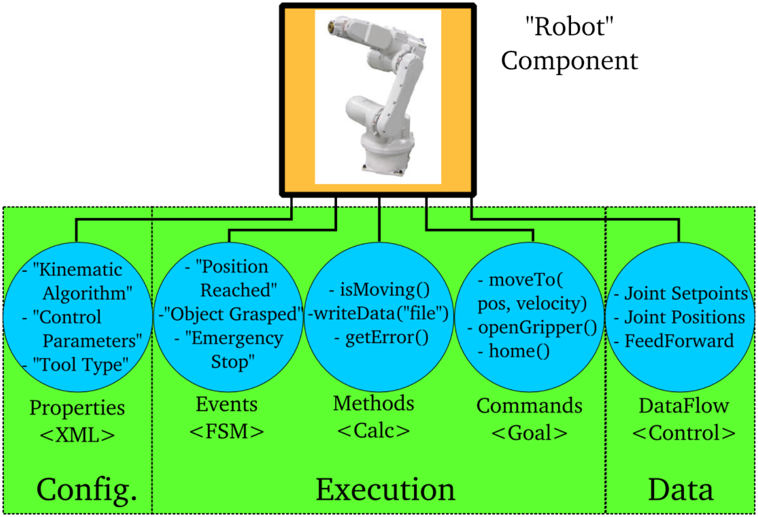


Component Interface

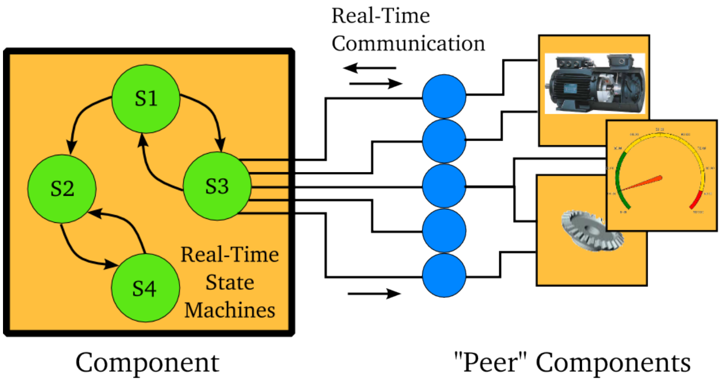


FMTC
Flanders' MECHATRONICS Technology Centre

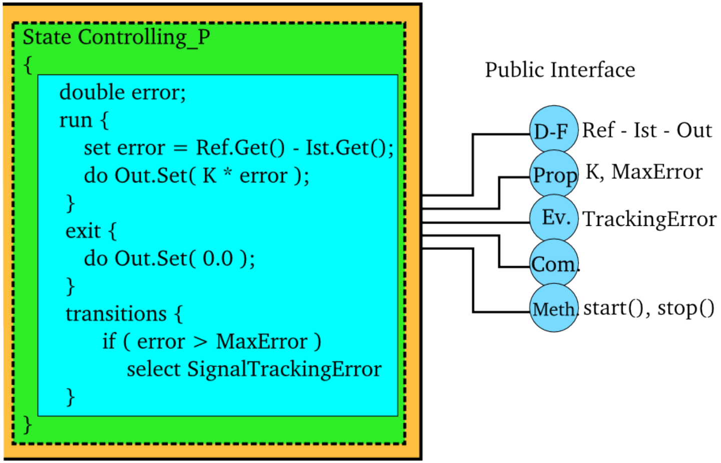
Component Interface



Component Implementation



State Machine Example



```
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
    }
}
```

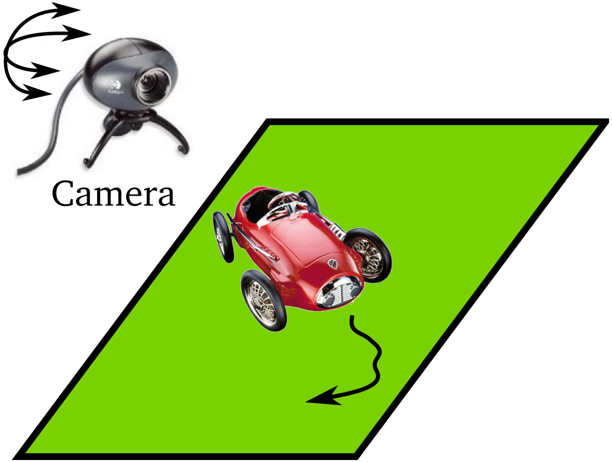
"P Controller Component"

Section Outline

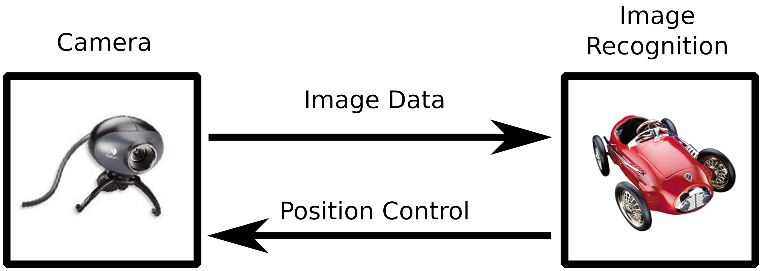
Example Application

How are these communication primitives used ?

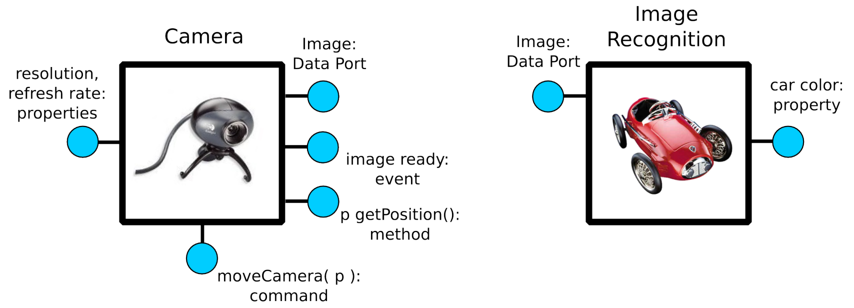
Example Application



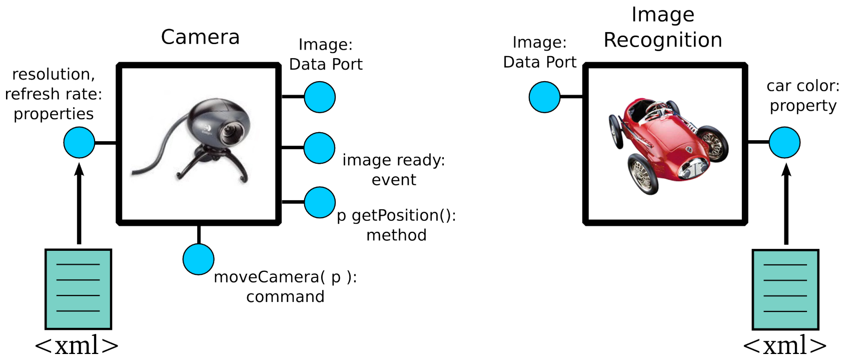
Deployment Configuration



Component Interface

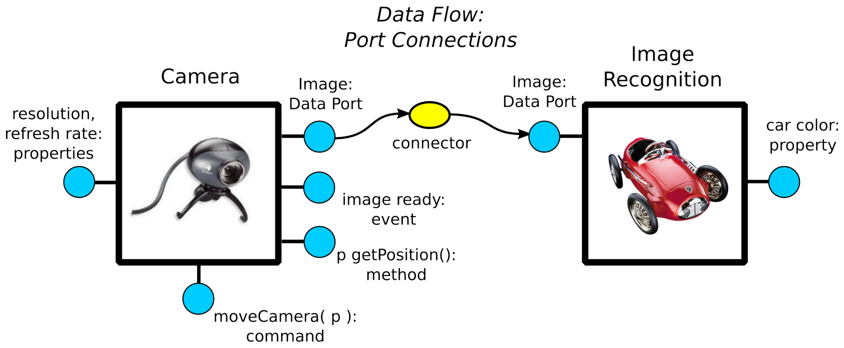


Communication: Configuration



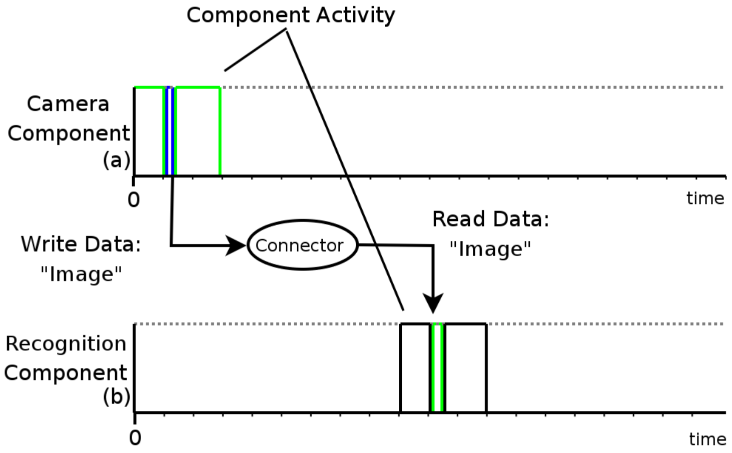
Configuration Flow : Properties

Communication: Data



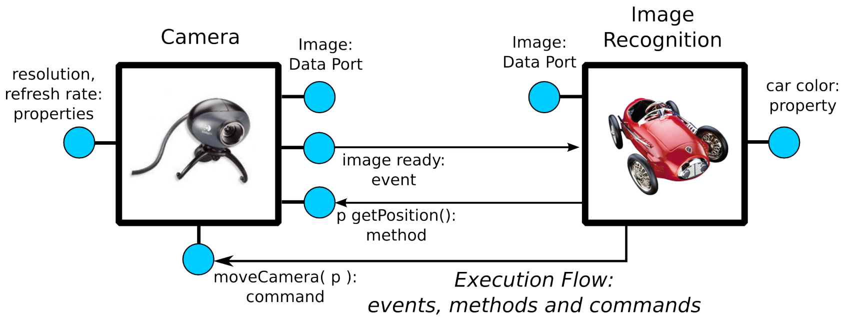
Data Flow : Ports and Connectors

Communication: Data



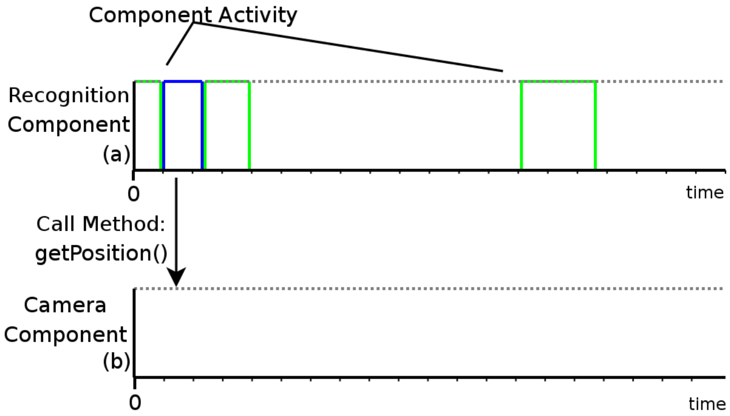
Data Flow : Ports and Connectors

Communication: Execution



Execution Flow

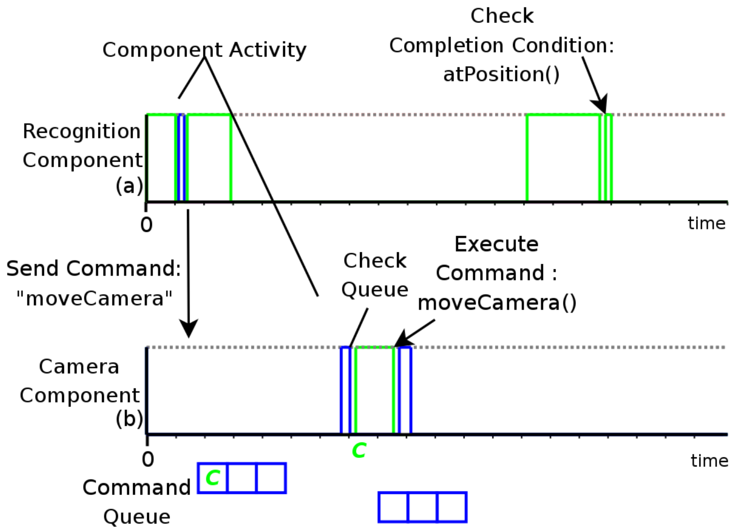
Communication: Execution



Execution Flow: Methods

Flanders' MECHATRONICS Technology Centre

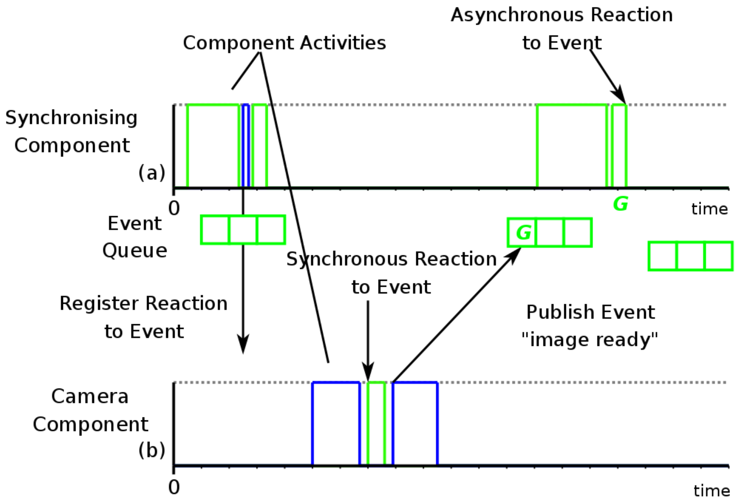
Communication: Execution



Execution Flow: Commands

Flanders' MECHATRONICS Technology Centre

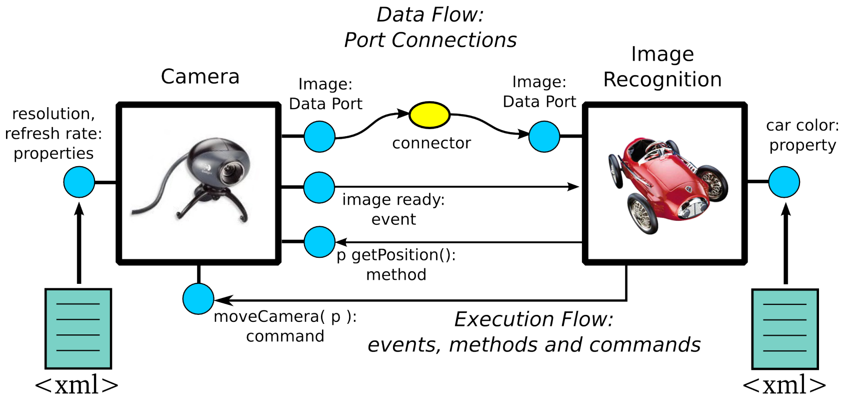
Communication: Execution



Component Activities
Asynchronous Reaction to Event
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



Flanders' MECHATRONICS Technology Centre

Example Application Summary

The following steps lead to a control application design:

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