Wrocław University of Technology

Chair of Cybernetics and Robotics

OROCOS framework

Mariusz Janiak
p. 331 C-3, 71 320 26 44

Contents

Introduction

Orocos in one-liners

- *Open Robot Control Software*
- Real-time Software Toolkits in C++
- Tool for developing components for control (real-time, thread-safe, interactive)
- Offers common component implementations

Web page

- http://www.orocos.org
- https://github.com/orocos-toolchain

Introduction

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"

Introduction

Hard realtime is Orocos

- Lock-free data ports favour highest priority component activity
- Realtime-aware memory management
- Does not prevent non-realtime use
- Support real-time Linux extensions: Xenomai and RTAI

Supported OS

- Linux 32/64bit (GNU,clang,Intel)
- Mac OS-X (GNU)
- Windows (XP $\rightarrow$ 7) 32/64bit (MSVS2005-2010)
- QNX (GNU) – beta

Introduction

Orocos sub-projects

- RTT – Real-Time Toolkit ("Run-Time Toolkit"!)
- KDL – Kinematics & Dynamics Library
- BFL – Bayesian Filtering Library
- OCL – Orocos Components Library

Real-Time Toolkit

A cross-platform, component framework in C++

- creating dynamic loadable and distributable components
- guarantee real-time, thread-safe communication
- real-time, event driven state machine scripts
- run-time interface inspection and communication

Real-Time Toolkit

Extensions to RTT

- Log4Cpp logging framework, with real-time logging support
- Lua scripting support, with application deployment and supervision
- OroGen / ROCK – model based code generation of components and applications
- ROS integration – open source framework for service robotics
- Networked component communication – Message queues, CORBA, Yarp, ZeroMQ (planned)

Real-Time Toolkit

Package

- is a directory on your filesystem
- contains one or more component
- plugin or typekit libraries
- contains a manifest.xml file
- can be installed or used in-place

Component

- exposes an algorithm to the rest of the software
- defines inputs, outputs and parameters
- is run by an activity
- is compiled into a library
- offers and uses services
- installs in lib/orocos

Real-Time Toolkit

Deployment

- description of a (part of) an application
- in an XML or script (ruby, rtt, Lua) file
- creates, connects, configures and starts components
- allocates threads and sets connection policies

Flow Ports

- publish and receive data for algorithms
- are In or Out and of a given data type
- Outs are send-and-forget
- Ins can wake us up (triggering)

Real-Time Toolkit

Connection Policy

- defines the connection between an Input and Output port
- defines data buffering, locking mechanism, and initial state
- allows to specify a transport

Transports

- connect Orocos components to other robotics frameworks or protocols
- handle Orocos data types over a given protocol
- can support streaming, connection-oriented or service-oriented communication
- may or may not be hard real-time

Real-Time Toolkit

Properties

- are structured name-value pairs
- are the run-time parameters
- can be serialized to XML

Operations

- are plain C/C++ functions
- are 'sent' or 'called'
- run in the caller's thread or the component's thread
- are grouped into service objects

Service

- a collection of flow ports, properties and operations
- is provided to and required by others
- can be loaded at run-time in a component

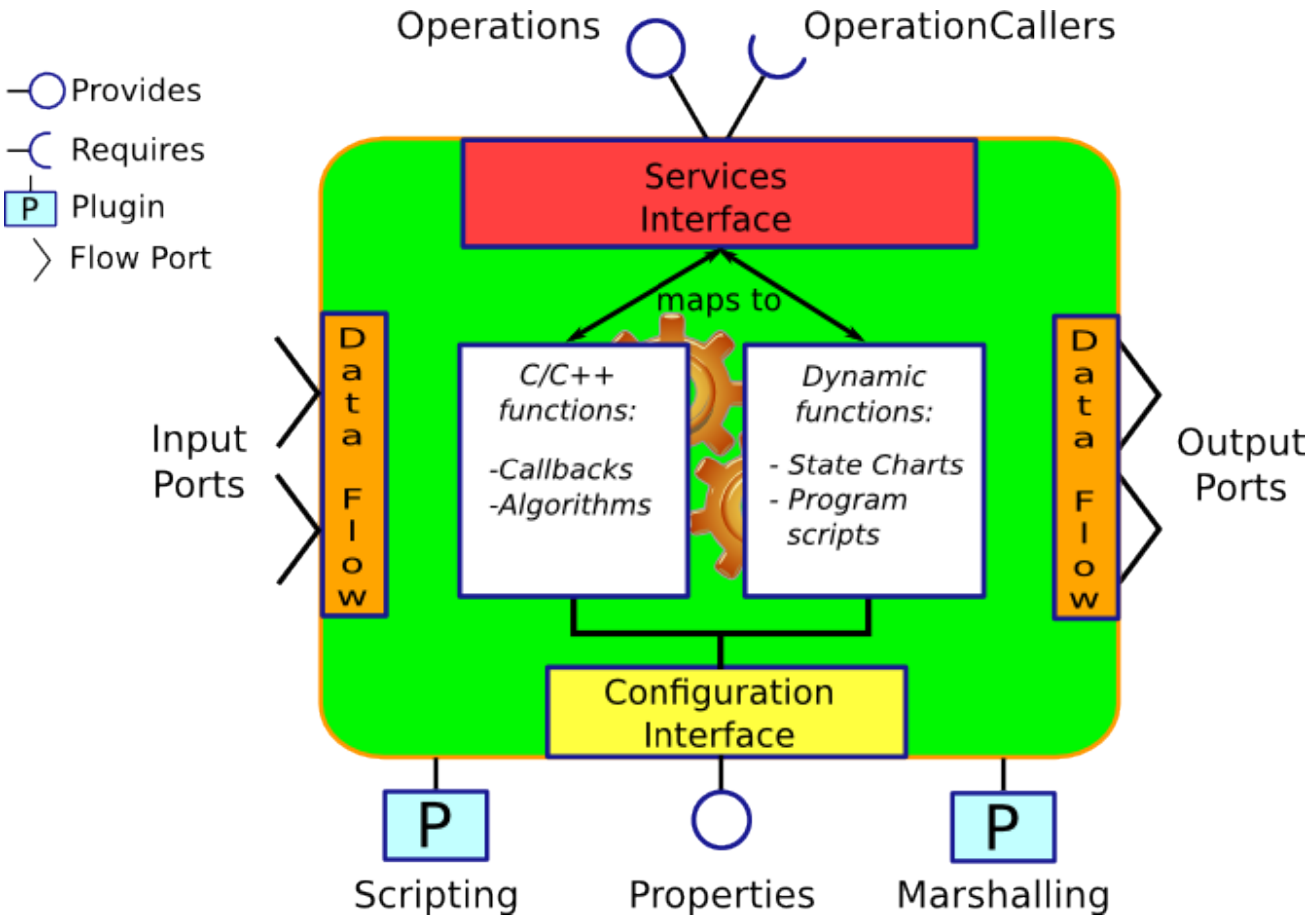
Real-Time Toolkit

Data Types

- Orocos C++ types
  - must be default constructible
  - must be copy-able
  - may be primitive types, structs, sequences (std::vector or []) or any combination
- typegen can use C++ types
  - that have all members as public
  - that are not templated
  - that have no parent class
- typekits are
  - required for each data type to be usable
  - generated by typegen if possible
  - hand-written in other cases

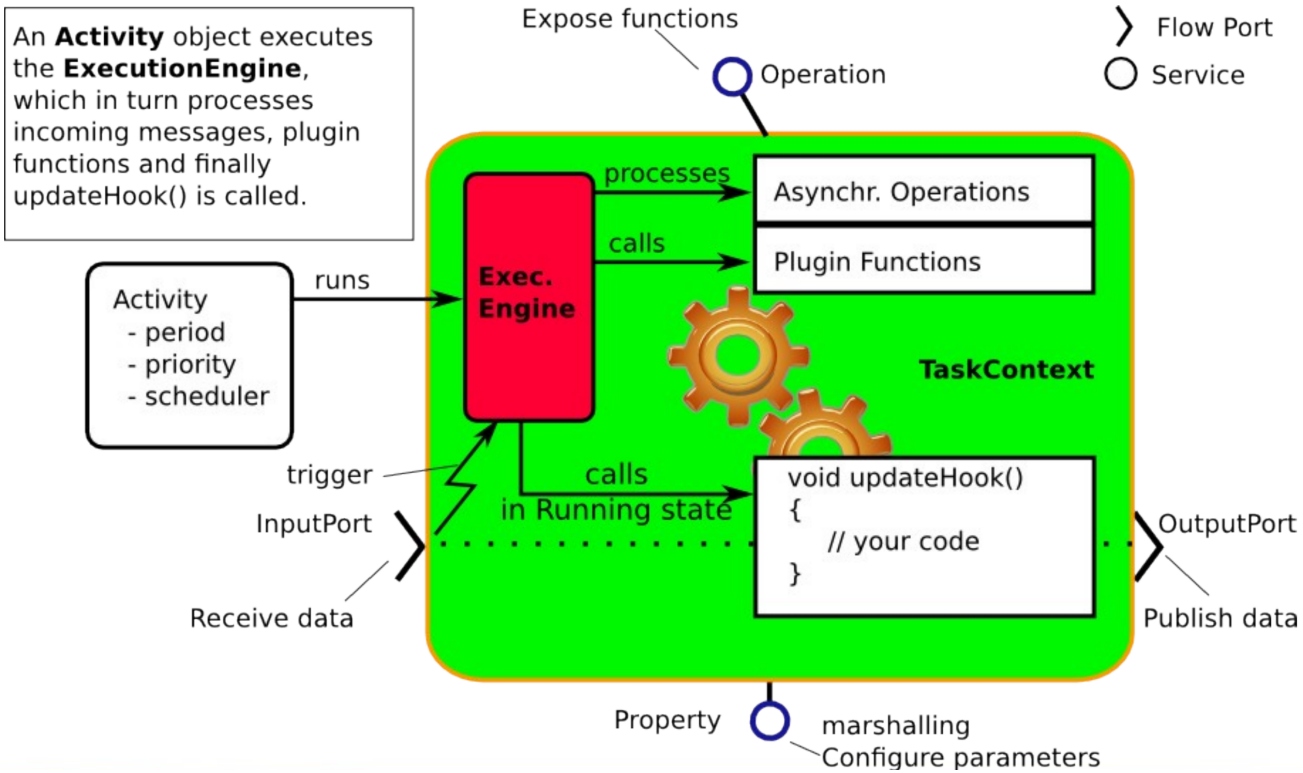
Real-Time Toolkit

RTT component



Real-Time Toolkit

Component architecture



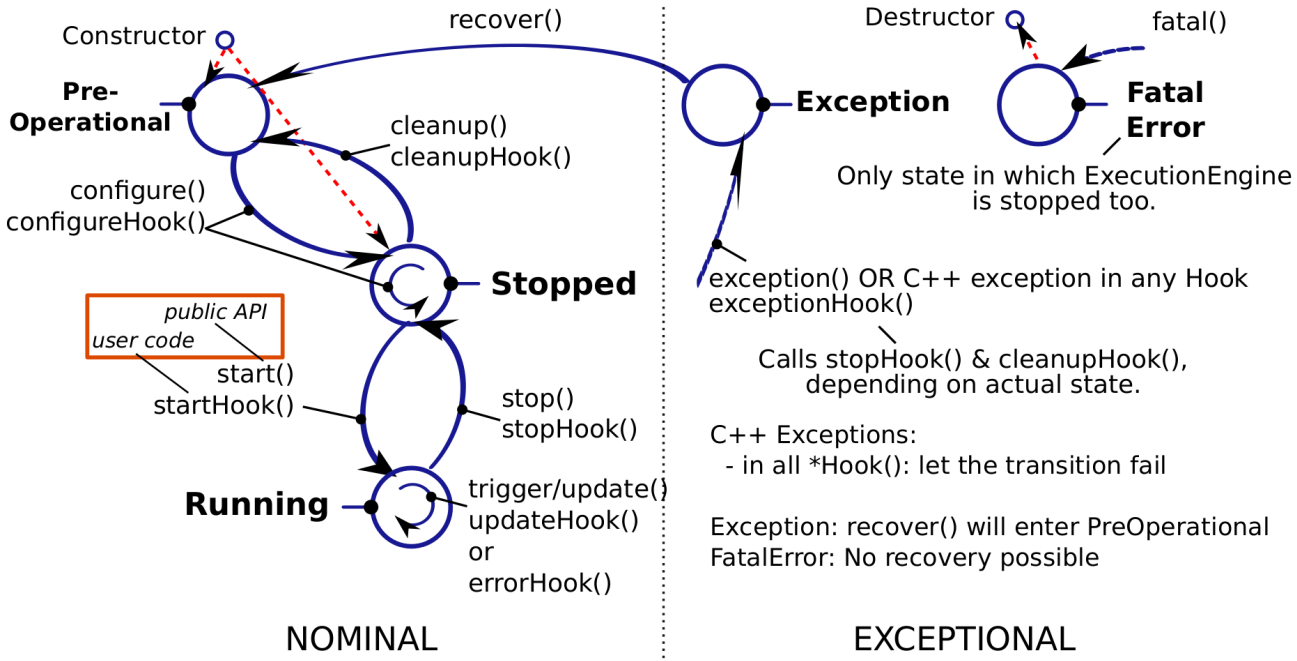
Real-Time Toolkit

RTT activity interface

- May map to a thread (RTT::Activity)
- May be a slave (RTT::extras::SlaveActivity)
- May be sequential (RTT::extras:SequentialActivity)
- May be anything else (RTT::extras::...)

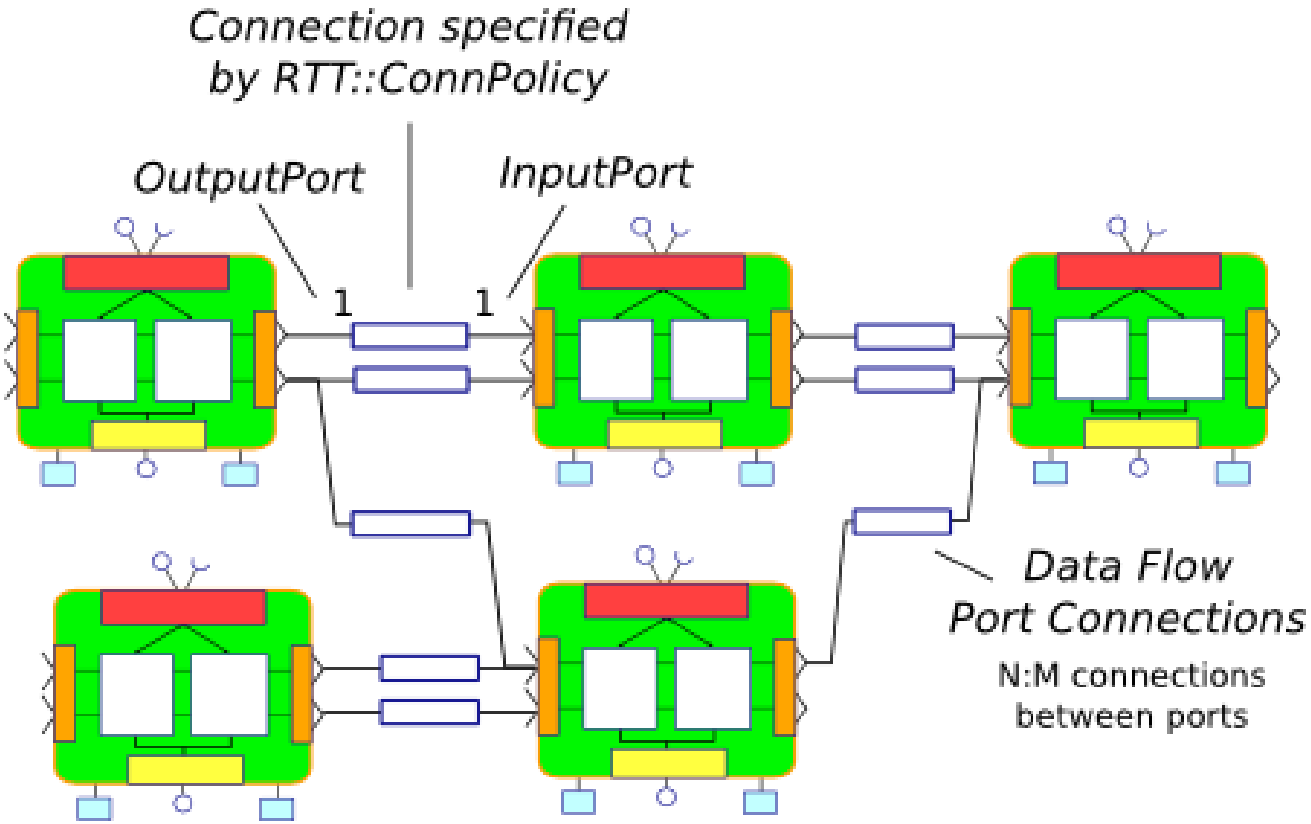
Real-Time Toolkit

Component lifecycle statemachine



Real-Time Toolkit

Data flow



Real-Time Toolkit

Component header file `Oro_test.hpp`

```
 1  #ifndef OROCOS_ORO_TEST_COMPONENT_HPP
 2  #define OROCOS_ORO_TEST_COMPONENT_HPP
 3
 4  #include <rtt/RTT.hpp>
 5
 6   class Oro_test : public RTT::TaskContext{
 7     public:
 8     Oro_test(std::string const& name);
 9     bool configureHook();
10     bool startHook();
11     void updateHook();
12     void stopHook();
13     void cleanupHook();
14   };
15  #endif
```

Real-Time Toolkit

Component source file `Oro_test.cpp`

```
 1  #include "oro_test-component.hpp"
 2  #include <rtt/Component.hpp>
 3
 4  Oro_test::Oro_test(std::string const& name) : TaskContext(name){;}
 5
 6  bool Oro_test::configureHook(){return true;}
 7
 8  bool Oro_test::startHook(){return true;}
 9
10  void Oro_test::updateHook(){;}
11
12  void Oro_test::stopHook() {;}
13
14  void Oro_test::cleanupHook() {;}
15
16  ORO_CREATE_COMPONENT(Oro_test)
```

The End

Thank you for your kind attention.

Test

0