Politechnika Wrocławska

Katedra Cybernetyki i Robotyki

Struktura ramowa OROCOS

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

Zawartość wykładu

Wprowadzenie

Orocos w kilku słowach

- *Open Robot Control Software*
- Struktura ramowa czau rzeczywistego dla języka C++
- Narzędzia służące tworzeniu komponentów do celów sterowania (czasu rzeczywistego, wielowątkowe, interaktywne)
- Implementuje spójny model komponentów

Strona domowa

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

Wprowadzenie

Historia

- 2001 – rozpoczął się jako mały projekt badawczy, sponsorowany przez Prof H. Bruynickx, KU Leuven
- 2001-2005 – rozwijany przez Petera Soetensa w ramach doktoratu, sponsorowany przez EU IST „Orocos", „Ocean" and „Open Machine Controller" projects and FMTC
- 2005 – zarządzany przez FMTC, "Modular Machines Group"

Wprowadzenie

Twardy czas rzeczywisty w Orocos

- Nieblokujące porty danych – faworyzowanie komponentów z wysokimi priorytetami
- Zarządzanie pamięcią uwzględniające ograniczenia czasu rzeczywistego
- Nie zapobiega wykorzystani z naruszeniem ograniczeń czasu rzeczywistego
- Wspiera rozszerzenia czasu rzeczywistego jądra Linux: Xenomai i RTAI

Wspierane systemy operacyjne

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

Wprowadzenie

Elementy składowe Orocos

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

Real-Time Toolkit

Wieloplatformowy, zorientowana na komponenty zestaw narzędzi dla języka C++

- tworzenie dynamicznych i możliwych do rozpraszania komponentów
- gwarantuje bezpieczną, międzywątkową komunikację w czasie rzeczywistym
- maszyny stanu sterowane zdarzeniami, działające w czasie rzeczywistym, definiowane w języku skryptowym
- Komunikacja i podgląd interfejsów w czasie działania komponentu

Real-Time Toolkit

Rozszerzenia RTT

- Log4Cpp biblioteka umożliwiająca tworzenie logów, ze wsparciem dla czasu rzeczywistego
- Wsparcie dla języka skryptowego LUA (implementacja komponentu, uruchamianie aplikacji, nadzór)
- OroGen / ROCK – generowanie kodu komponentów i aplikacji w oparciu o model
- Integracja z ROS – struktura ramowa dla robotów usługowych
- Komunikacja sieciowa komponentów – Message queues, CORBA, Yarp, ZeroMQ (w planach)

Real-Time Toolkit

Pakiet (*Package*)

- katalog na dysku
- zawiera jeden lub więcej komponentów
- wtyczka (*plugin*) lub typekit
- zawiera plik a manifest.xml
- może być instalowany lub użytkowany w miejscu swojego położenia

Komponent (*Component*)

- udostępnia algorytm pozostałej części oprogramowania
- definiuje wejścia, wyjścia i parametry
- wykonuje tzw. działanie (*activity*)
- budowany do postaci biblioteki
- dostarcza i wykorzystuje usługi
- instalowany w katalogu lib/orocos

Real-Time Toolkit

Uruchomienie (*Deployment*)

- opis (częsci) aplikacji
- plik XML lub skrypt (ruby, rtt, Lua)
- tworzy, łączy, konfiguruje i inicjuje komponenty
- przydziela wątki i konfiguruje kanały komunikacji

Porty (*Flow Ports*)

- wysyła i odbiera dane dla algorytmów
- są Wejściowe i Wyjściowe, o ustalonym typie
- Wyjściowe działają w sposób wyślij i zapomnij
- Wejściowe mogą wybudzić komponent (*triggering*)

Real-Time Toolkit

Reguły połączeń (*Connection Policy*)

- definiuje połączenia pomiędzy portami wejściowymi i wyjściowymi
- definiuje sposób buforowania danych, mechanizm blokowania i stan początkowy
- umożliwia wybór warstwy transportowej

Warstwa transportowa (*Transports*)

- łączy komponenty Orocos z innymi robotycznymi strukturami ramowymi lub protokołami
- zarządza typami danych Orocos w rach ustalonego protokołu
- może obsługiwać transmisję strumieniową, komunikację zorientowaną na połączenie lub zorientowaną na usługi
- może lub nie być czasu rzeczywistego

Real-Time Toolkit

Parametry (*Properties*)

- tworzą pary nazwa-wartość
- mogę być definiowane podczas działania
- mogą być zapisane/wczytane do/z pliku XML

Metody (*Operations*)

- zwykłe funkcje C/C++
- mogą być 'wysłane' (*sent*) lub 'wywołane' (*called*)
- mogą być wykonane w ramach wątku komponentu wywołującego lub dostarczającego metodę
- są grupowane w obiekty usług

Usługa (*Service*)

- zbiór portów, parametrów i metod
- jest dostarczany lub wymagany przez innyh
- może być załadowany do komponentu w trakcie jego działania

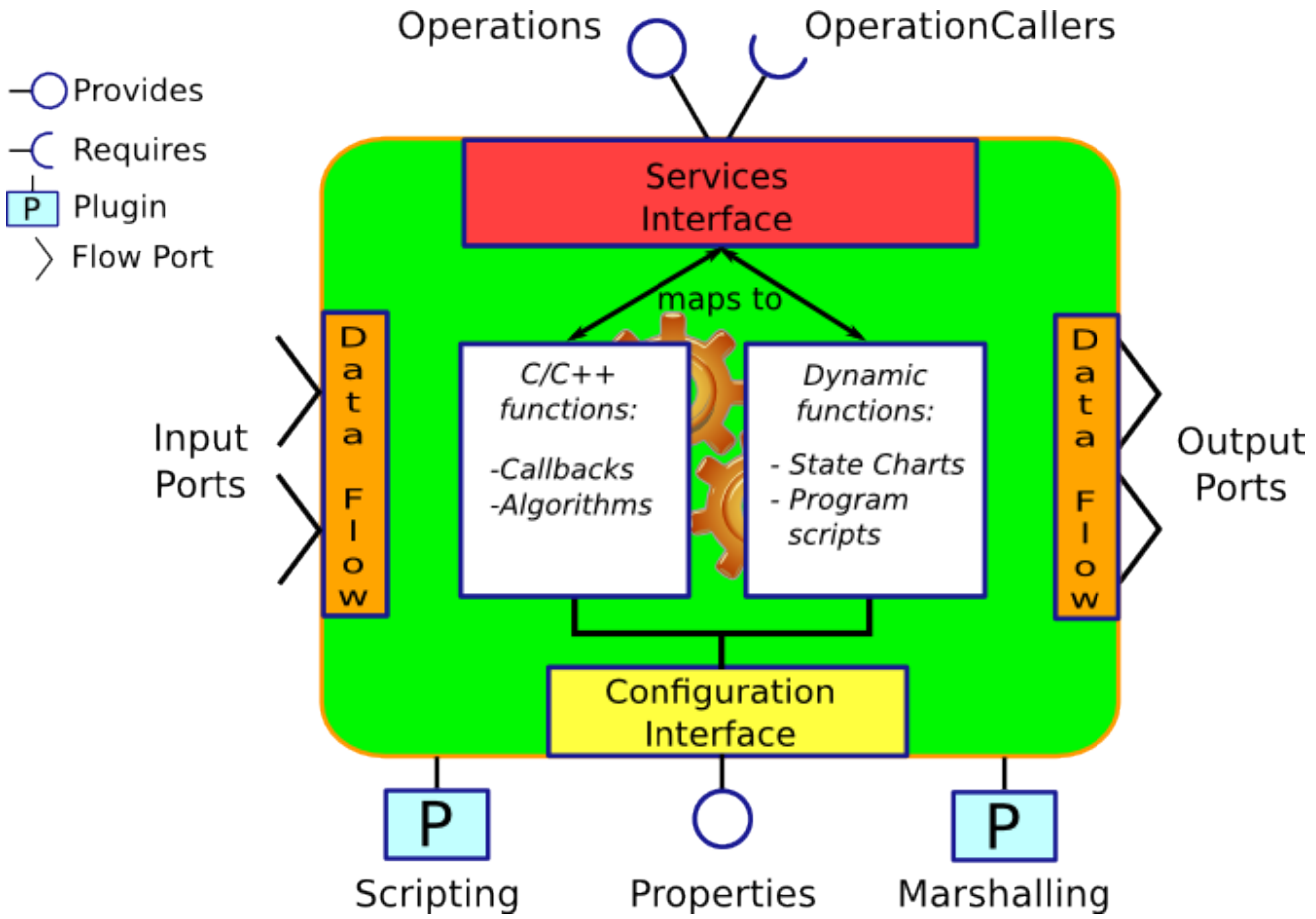
Real-Time Toolkit

Typy danych

- Typy Orocos C++
  - musi posiadać standardowy konstruktor bezparametryczny
  - musi umożliwiać kopiowanie
  - może być typem wbudowanym, strukturą, sekwencją (std::vector lub []) lub dowolną kombinacją
- typegen może wykorzystywać typy C++
  - które mają wszystkie pola publiczne
  - które nie są szablonami
  - które są klasami bazowymi (bez rodziców)
- typekits są
  - wymagane dla każdego typu danych, który ma być używany
  - generowane przez typegen jeżeli to możliwe
  - w innych przypadkach napisane we własnym zakresie

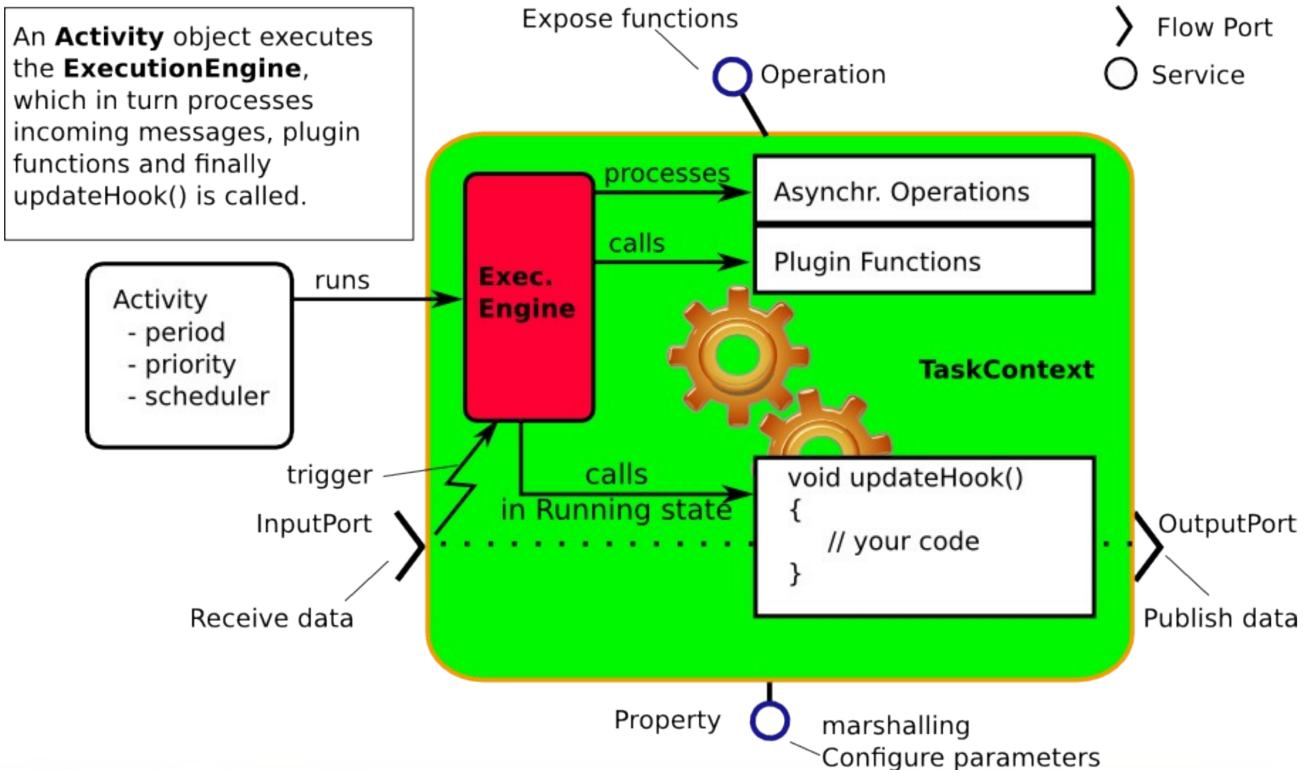
Real-Time Toolkit

Komponent RTT



Real-Time Toolkit

Architektura komponentu



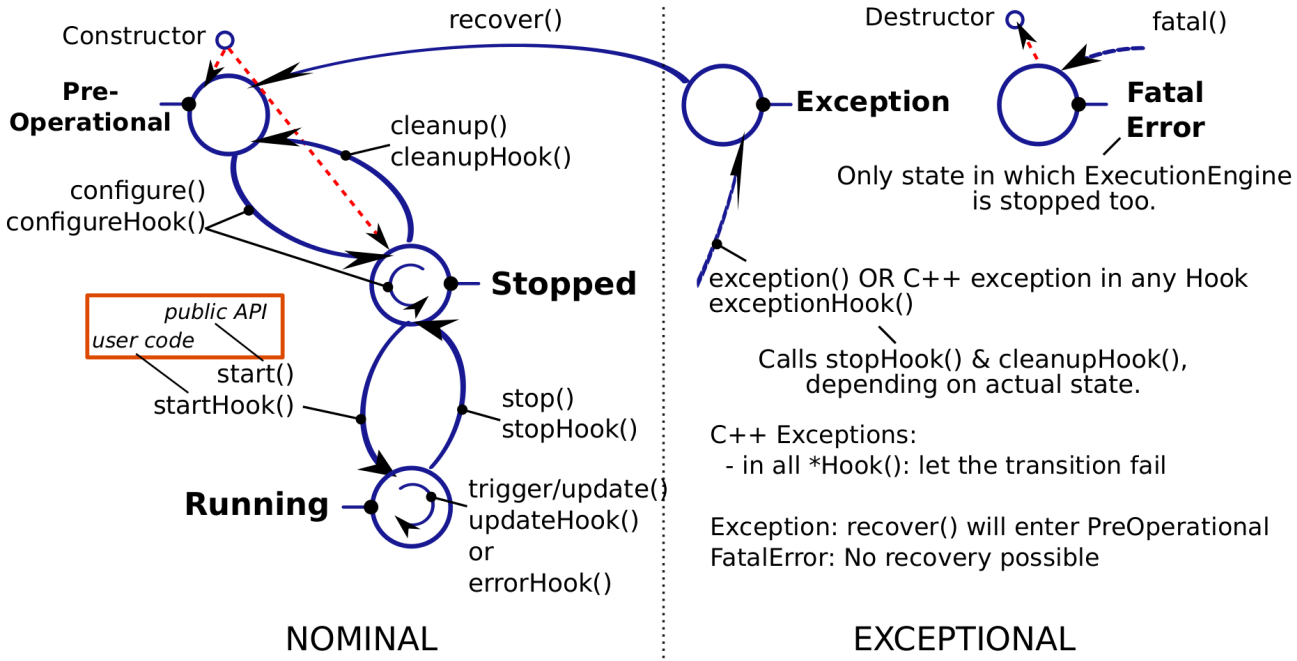
Real-Time Toolkit

Interfejs *activity* RTT

- Może reprezentować wątek (RTT::Activity)
- Może być zależny (RTT::extras::SlaveActivity)
- Może być sekwencyjny (RTT::extras:SequentialActivity)
- Może być czymkolwiek innym (RTT::extras::...)

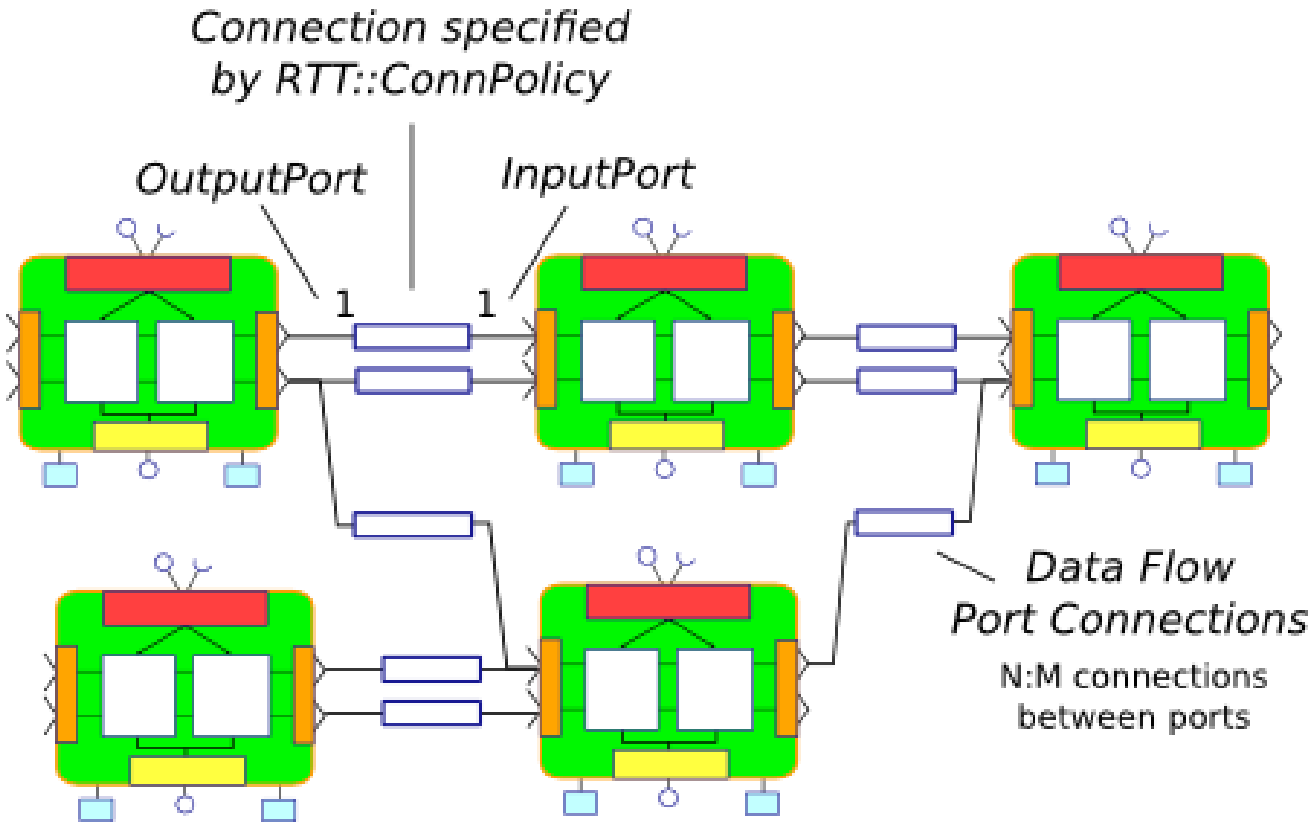
Real-Time Toolkit

Cykl życia komponentu



Real-Time Toolkit

Przepływ danych



Real-Time Toolkit

Plik nagłówkowy komponentu `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

Plik źródłowy komponentu `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)
```

Real-Time Toolkit

- The Orocos Component Builder's Manual

  http://www.orocos.org/stable/documentation/rtt/v2.x/doc-xml/orocos-components-manual.html

- Using orocreate-pkg

  http://www.orocos.org/wiki/orocos/toolchain/getting-started/using-orocreate-pkg

- Configuring and Starting Components from an Orocos Script

  http://www.orocos.org/wiki/orocos/toolchain/getting-started/toolchain-tutorials/configuring-and-starting-components-orocos

- The Deployment Component

  http://www.orocos.org/stable/documentation/ocl/v2.x/doc-xml/orocos-deployment.html

- The TaskBrowser Component

  http://www.orocos.org/stable/documentation/ocl/v2.x/doc-xml/orocos-taskbrowser.html

- LuaCookbook

  http://www.orocos.org/wiki/orocos/toolchain/luacookbook

Koniec

Dziękuję za uwagę

Test

0