An introduction to the Orocos Real-Time Toolkit

rFSM

and

iTaSC

Real-time and Portable

- **Hard realtime is Orocos' competitive advantage:**
  - **Lock-free data ports favour highest priority component activity.**
  - **Realtime-aware memory management.**
  - **Does not prevent non-realtime use!**
- **... in an extremely flexible and extendible programming environment**

Real-time and Portable

- Orocos Toolchain is supported on:
  - Linux 32/64bit (GNU,clang,Intel)
  - Real-Time Linux Extensions
    - Xenomai
    - Real-Time Application Interface (RTAI)
  - Mac OS-X (GNU)
  - Windows (XP->7) 32/64bit (MSVS2005-2010)
  - QNX (GNU) – beta

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)

Extensions to RTT

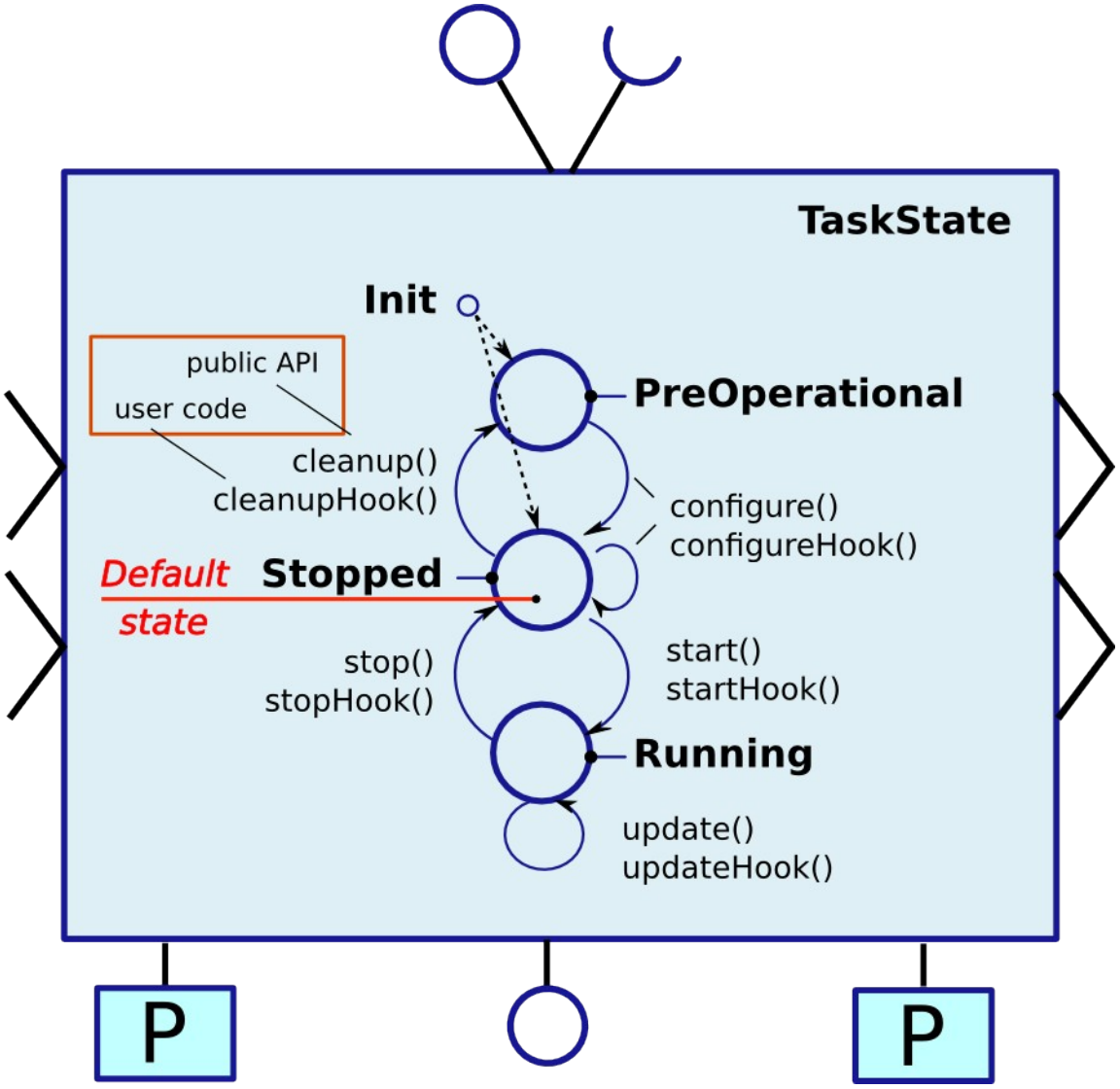
- **iTaSC**
  - **instantaneous Task Specification using Constraints**
- **rFSM**
  - **(independent) hierarchical state machines**

Who’s using Orocos. . . ?

- **Institutes:**
  - **K.U.Leuven (B), FU Berlin (D), Polytechnic University of Catalonia (ES), University of Florida (US), German research centre for Artificial Intelligence (DFKI, Bremen, D), University of New South Wales (Sydney, AU), University of Maryland (USA), Polytechnic University of Milan (I), University of Southern Denmark, Onera (F), Irisa (F), Cea (F)**
- **Companies:**
  - **three FMTC member companies (with already one product several years on the market!); Willow Garage; and some other machine or robotics builder companies in Belgium, France, Spain, Canada, USA (NASA) and the Netherlands, with several products on the market**

A Component's Life Cycle

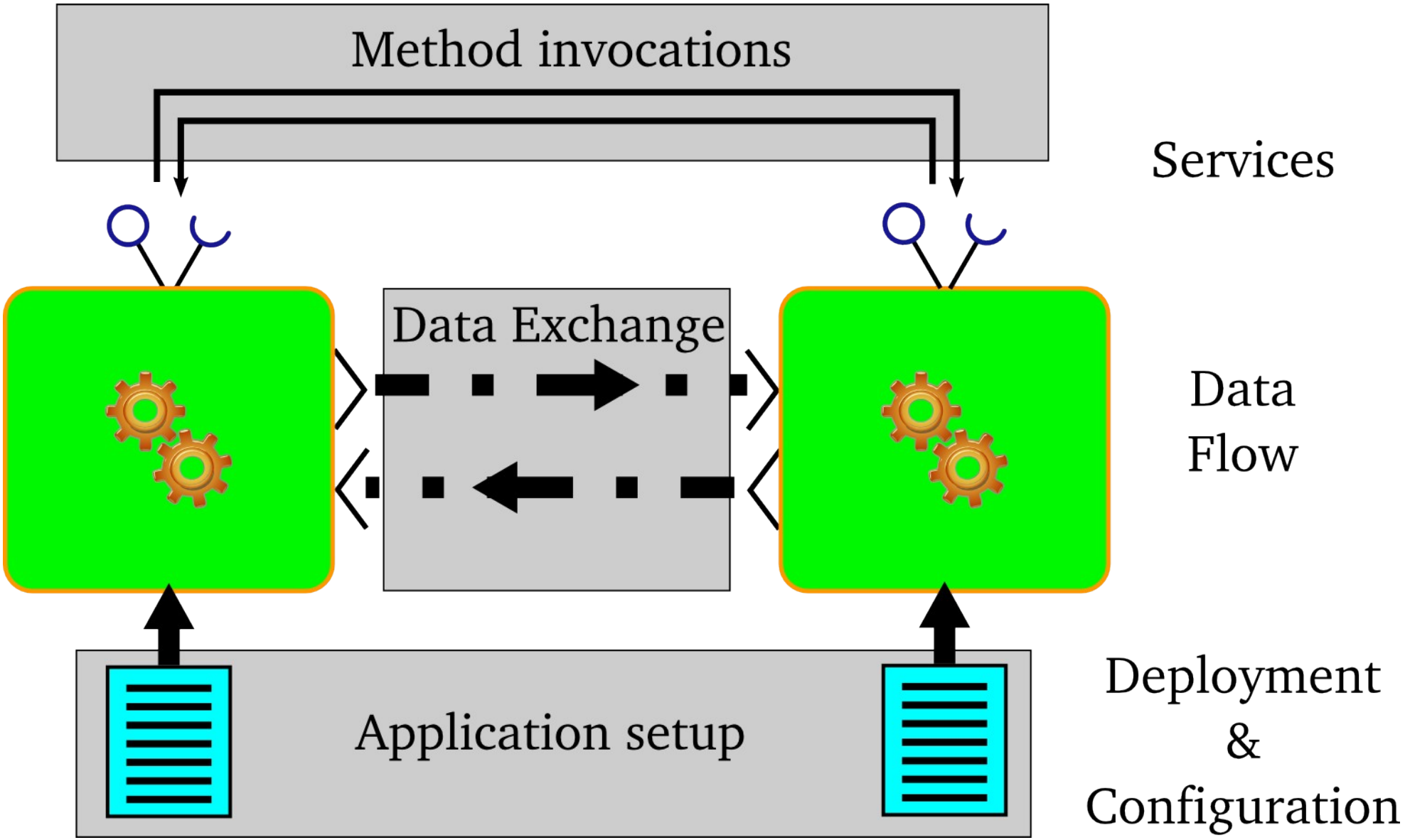
- Allows non real-time configuration and cleanup
- Starting is only allowed once configured correctly
- Extended with basic error states



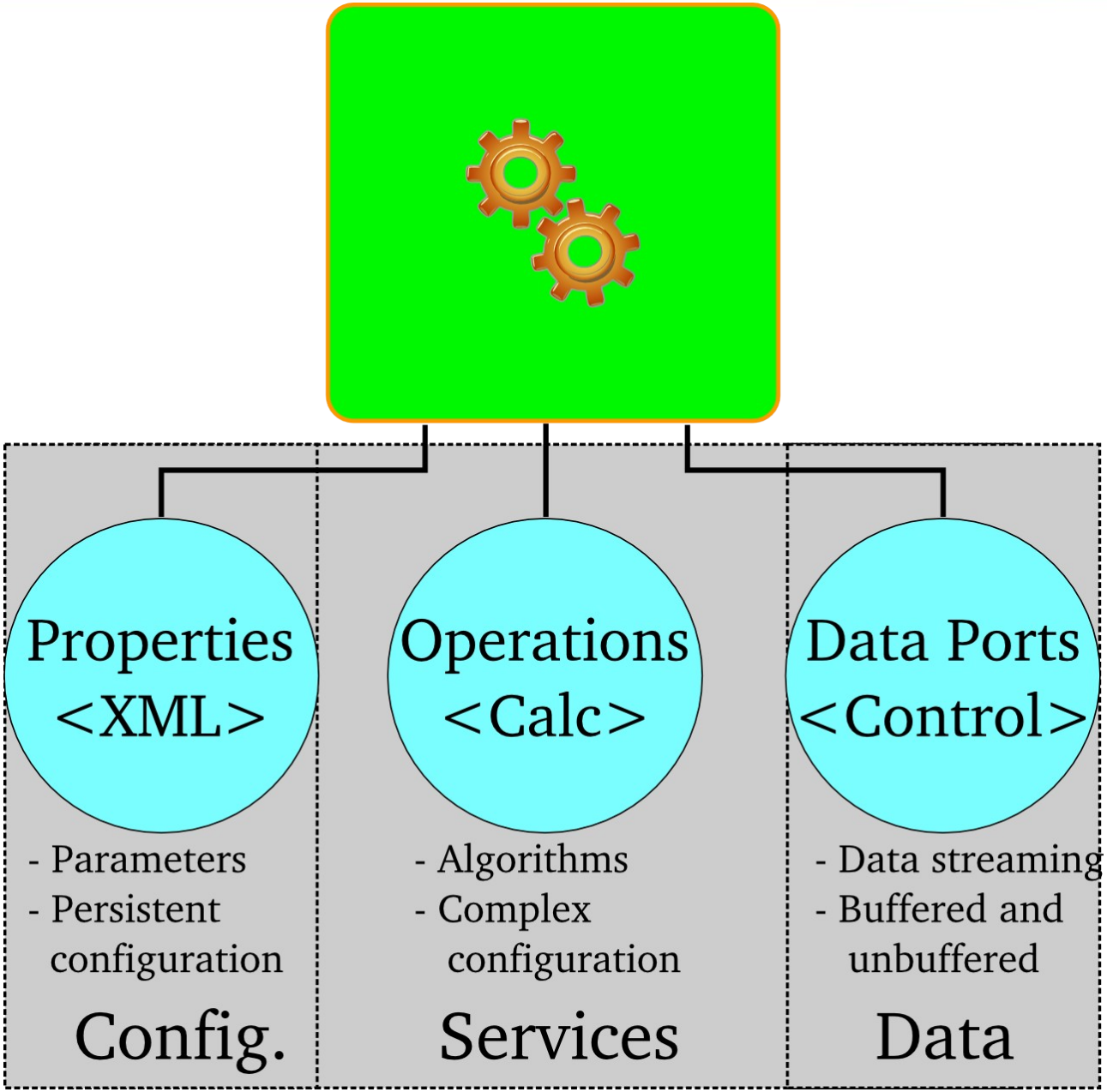
A Component’s Basic Communication

- In which ways can components communicate?
  - Configuration of parameters
  - Exchange (streaming) data
  - Cooperate to achieve a task

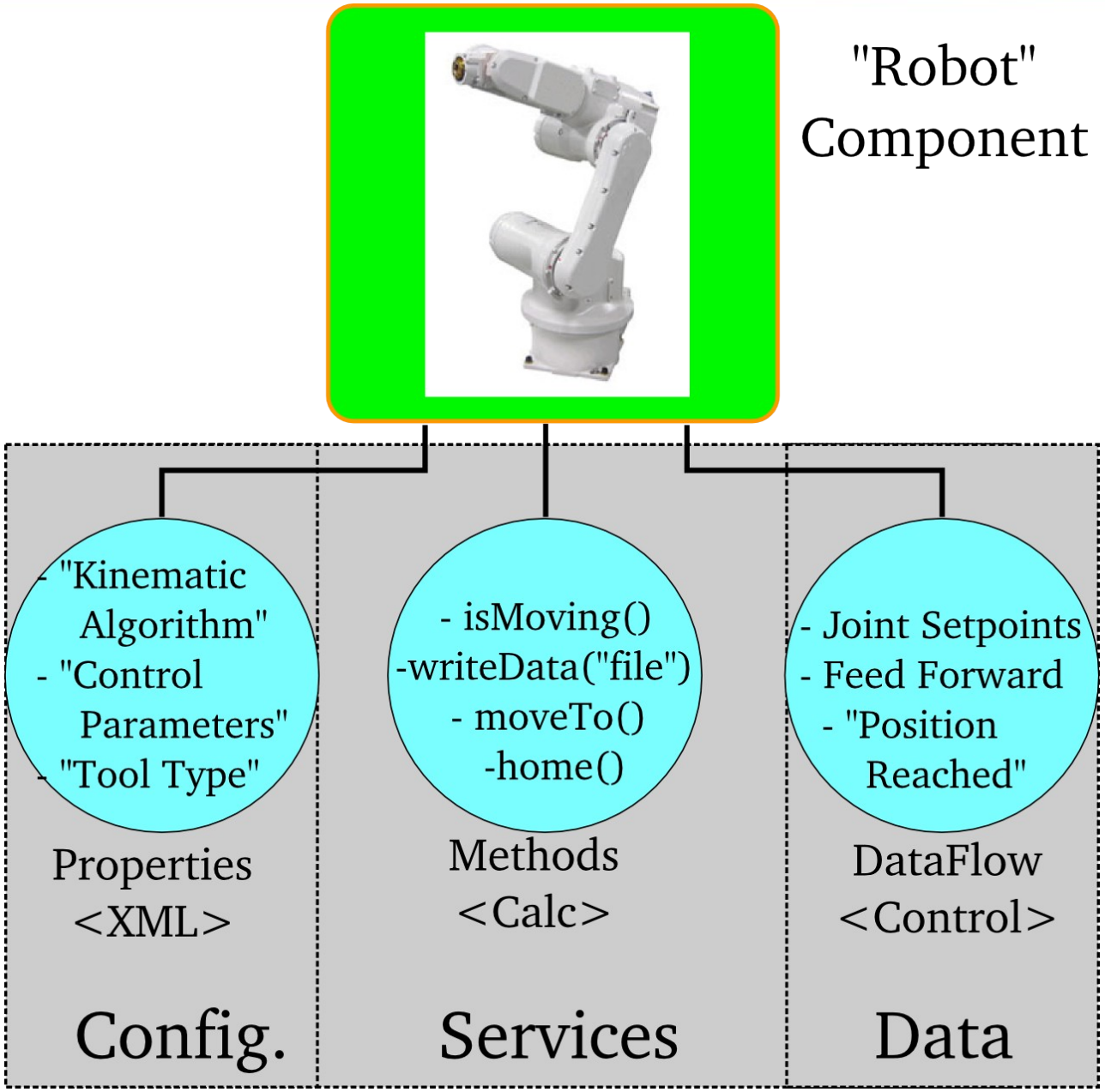
A Component's Basic Communication



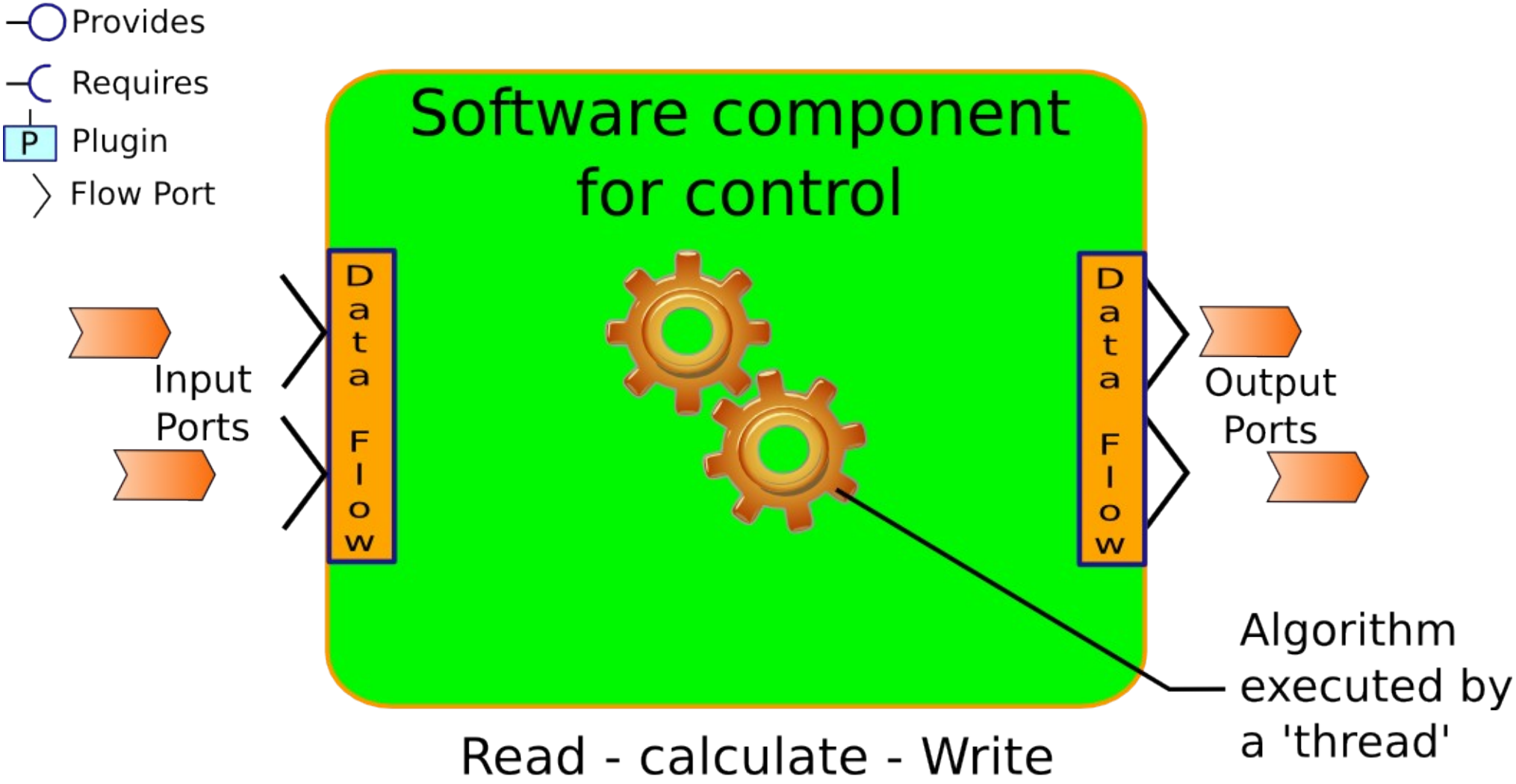
A Component's Interface: Robot Example



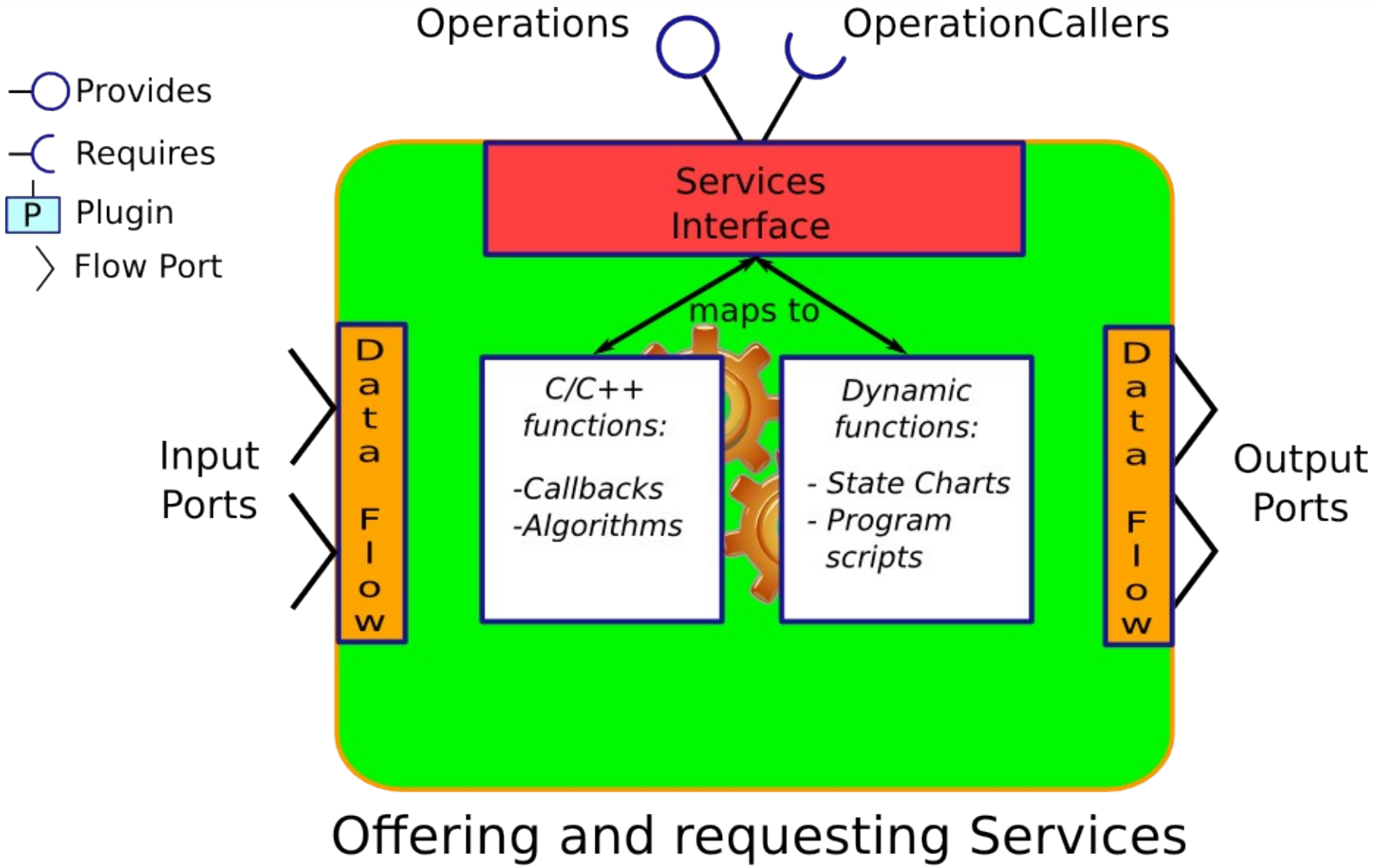
A Component's Interface: Robot Example



Component break-down

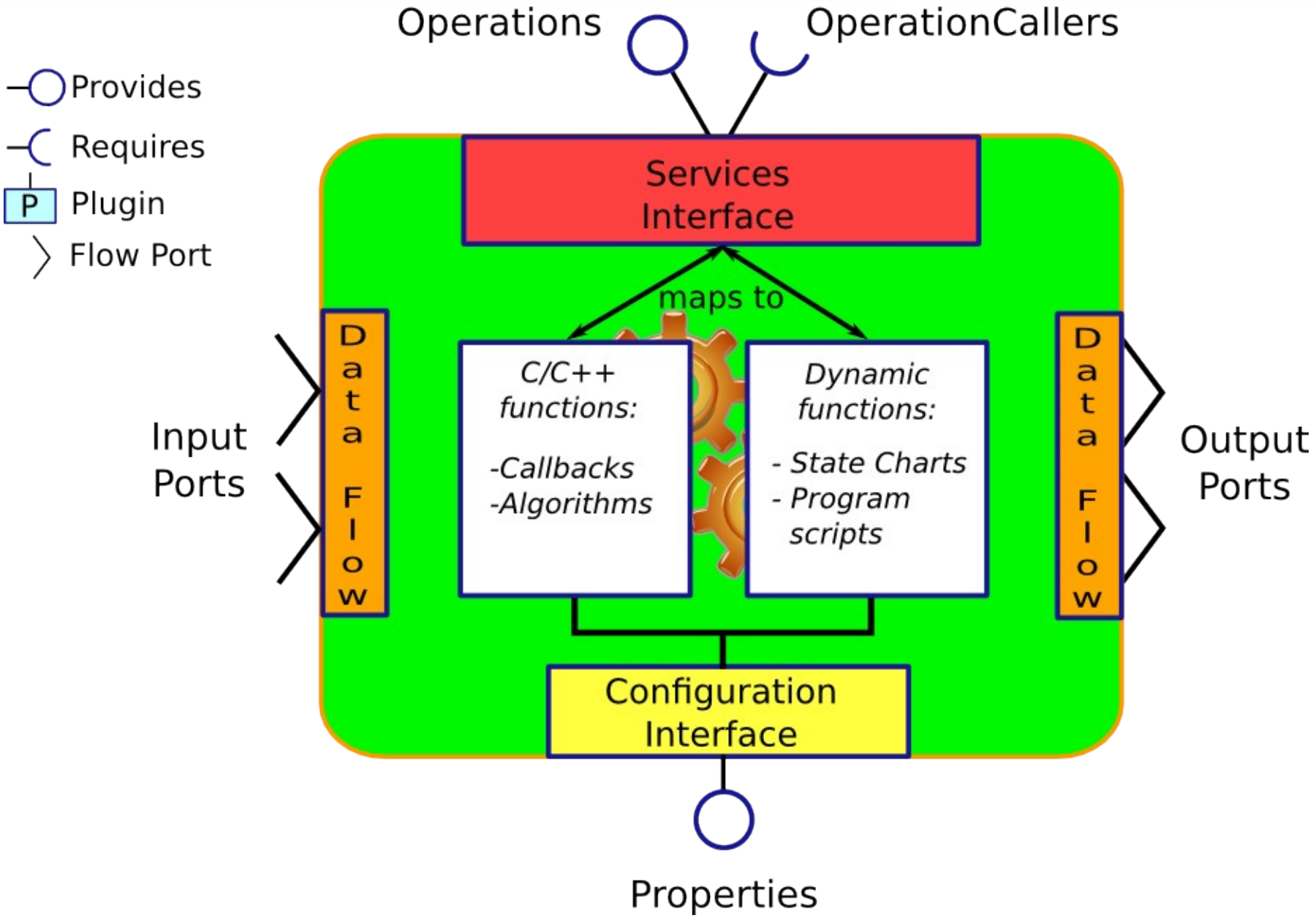


Component break-down

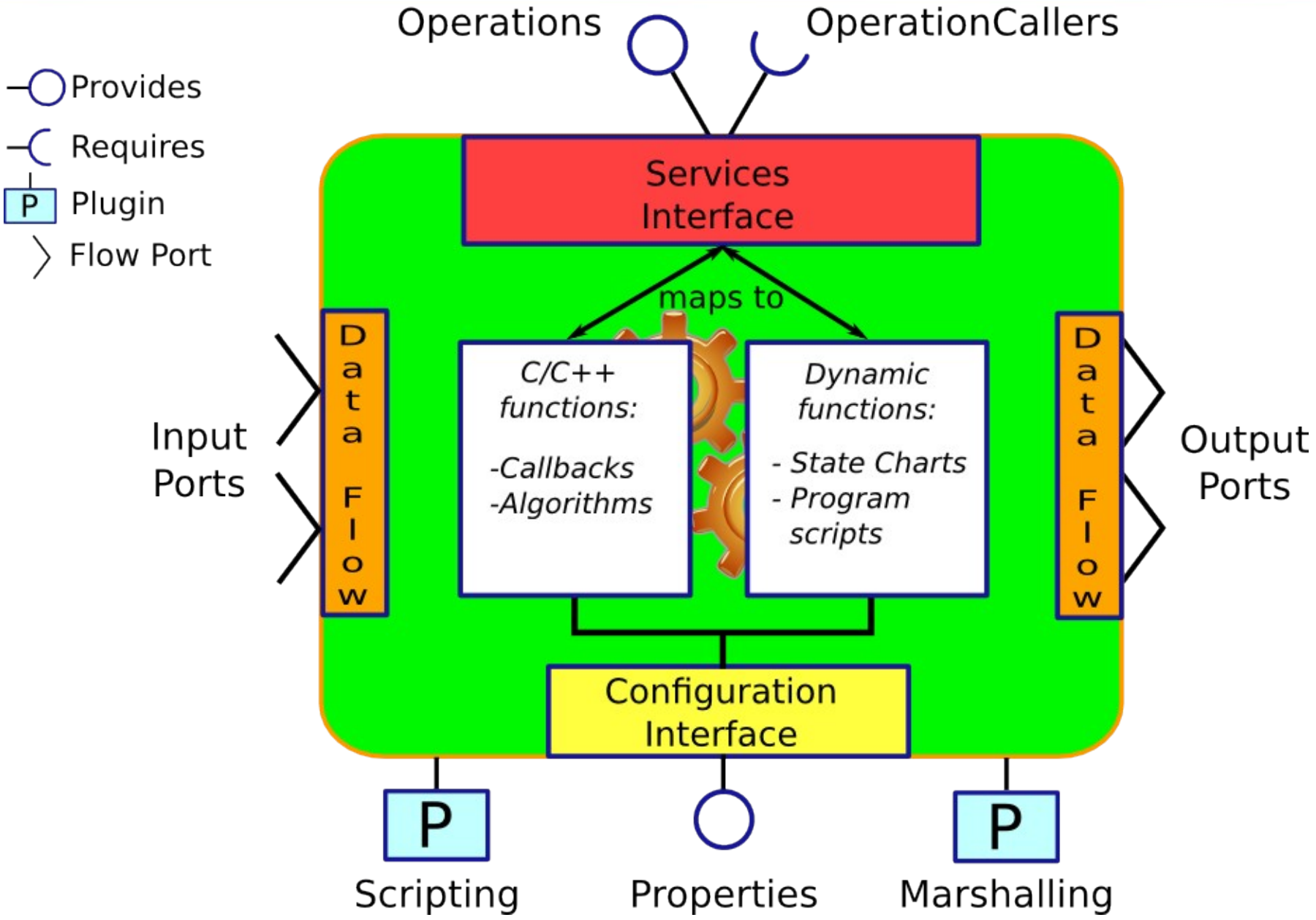


Offering and requesting Services

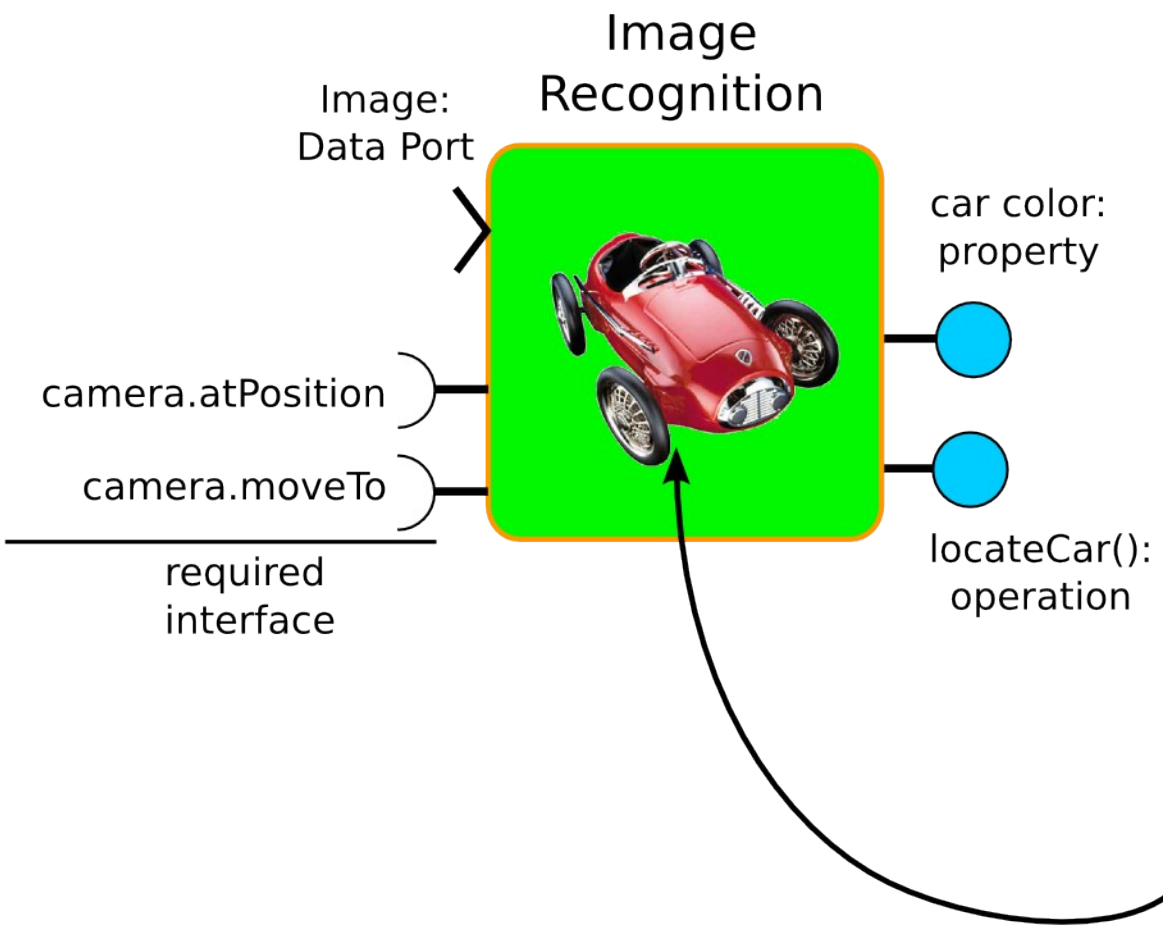
Component break-down



Component break-down



Example of image recognition



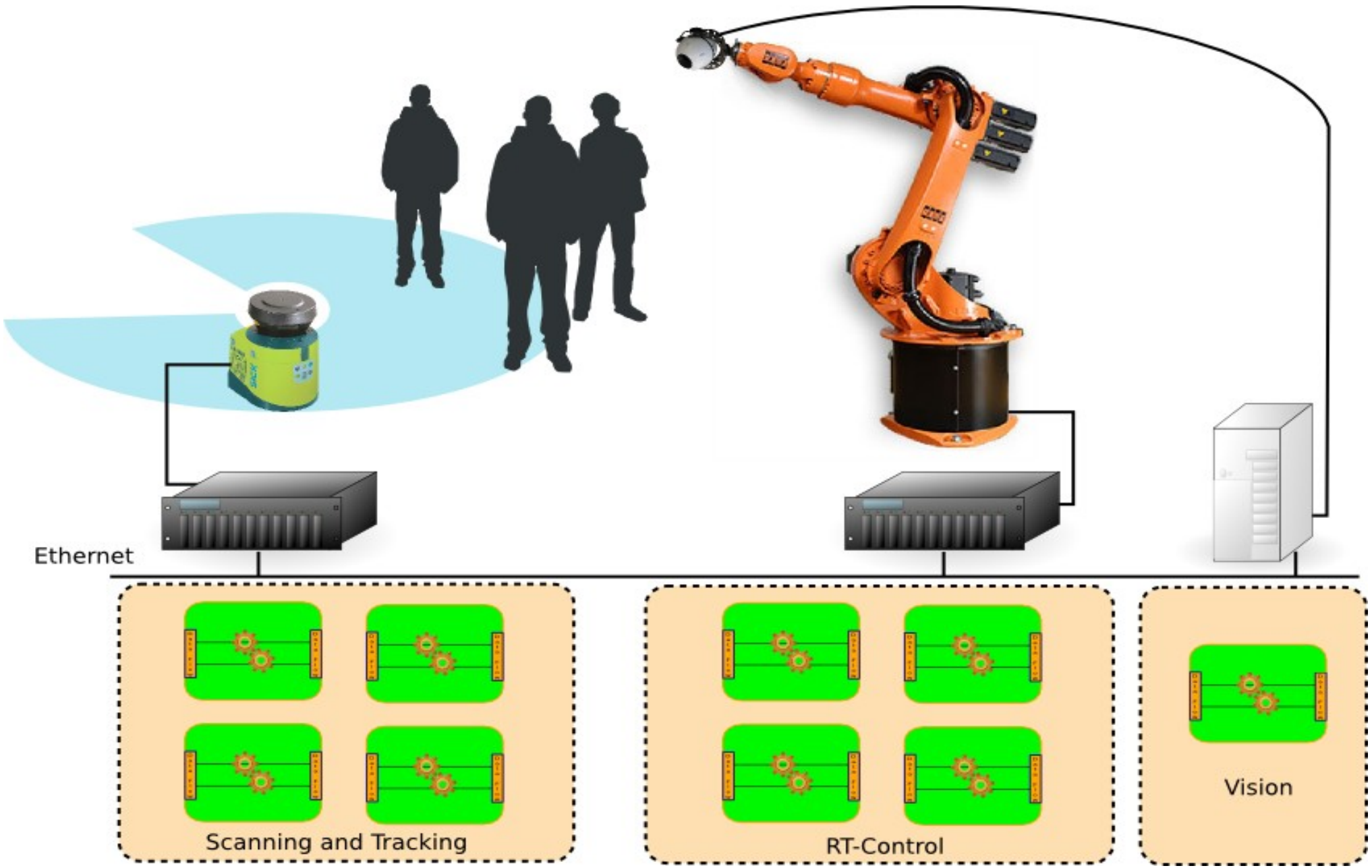
file: statemachine.osd

```
StateMachine ExampleSM
{
   initial state wait_for_image {
        // on imageReady event, make
        // transition to other state:
        transition Image( new_image )
             select image_captured;
   }

   state image_captured {
     // program executed when this state
     // is entered:
     entry {
        set p = this.locateCar();
        camera.moveTo( p );
     }
     // when entry is done, go back:
     if camera.atPosition( p )
          select wait_for_image;
   }

   final state end {}
}
RootMachine ExampleSM sm;
```

Example Application



Code break-down

```cpp
#include <rtt/TaskContext.hpp>
#include <ocl/ComponentDeployment.hpp>

/**
 * Note: we're defining a component as a class in a .cpp file
 * not in a header file !
 */
class MyComponent
    : public RTT::TaskContext
{
public:
    MyComponent(string name)
        : RTT::TaskContext(name)
    {
        addOperation("set_param",&MyComponent::set_param,this)
            .doc("Set parameter X")
            .arg("value", "The argument of this method.");
    }

    bool configureHook() {
        // further setup which could not be done in the
        // constructor...
    }

    void set_param(double x) {
        // ...
    }
};

/**
 * Make MyComponent dynamically loadable.
 */
ORO_CREATE_COMPONENT( MyComponent )
```

MyComponent.cpp

RTT::TaskContext is our base

Interface building

User code is placed in hooks...

... or in custom functions

Makes this library a loadable Orocos component

That’s all folks !

Pay us a visit at http://www.orocos.org