Towards Automatic Migration of ROS Components from Software to Hardware

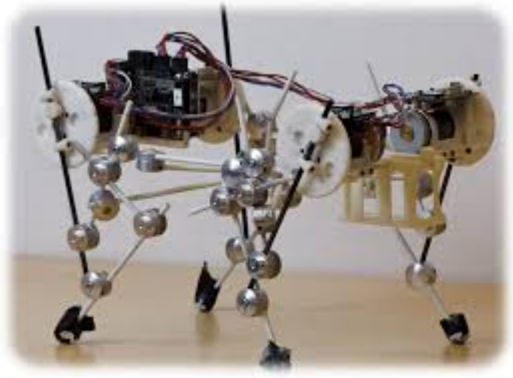
Presented at DSLRob 2013

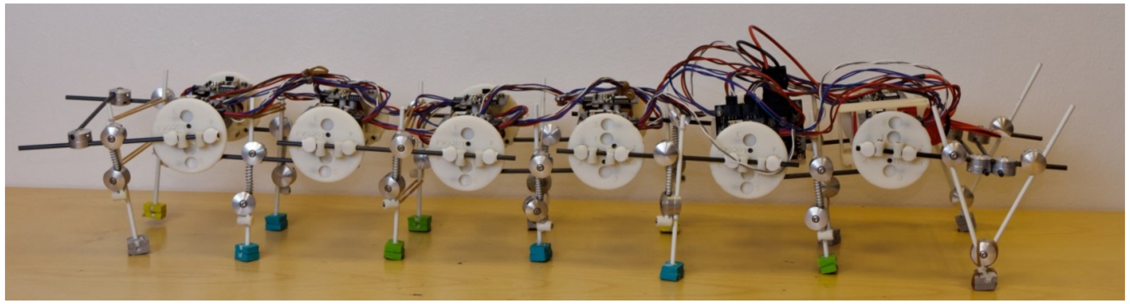
Anders Lange, Ulrik Schultz and Anders Soerensen
The Maersk McKinney Moeller Institute,
University of Southern Denmark, Denmark

Introduction

- Embedded Systems group
- What we do
  - Experimental & Modular robotics
  - Field Robotics
  - Cyber Physical Systems
    - Especially for welfare technology

Modular Robotics





Cyber Physical Systems and Welfare technology

Welfare Technology projects

- RoBlood
- RoboTrainer
- RoboTrainer-Light
- Elastic/Exercise-band sensor
- And much much more..

Why FPGA's

- We primarily use FPGA and Hybrid FPGA<>MCU SoCs (like the Xilinx Zynq) because they:
  - Reduces our need for multiple hardware platforms
    - i.e. we use a single generic FPGA node.
  - Enables flexible and rapid development of custom interfaces and controllers for virtually any application.
  - Performance and timing is easy to handle, as synchronous logic is clock cycle predictable.
- The problems
  - FPGA development is often much more complicated and tedious than software development partly due to a lack of good, open-source, vendor independent component libraries.
  - Even though vendor tools are becoming MUCH better… there is still the issue of many quality IP/components only being available for a steep price…

The Unity Framework

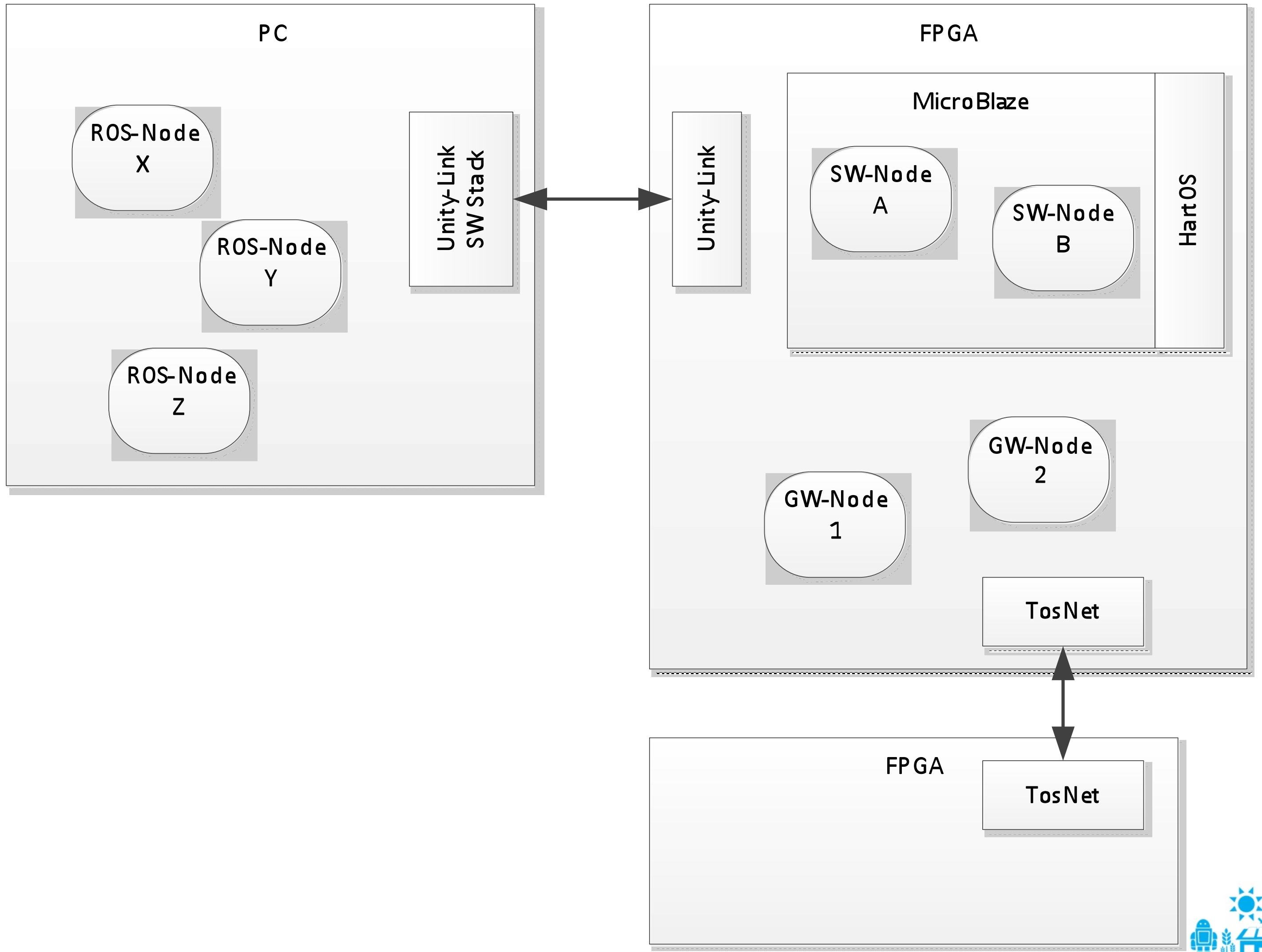
- **The Unity-Library**
  - A set of support libraries and configurable components
  - Sensor and Actuator interfaces
  - Communication interfaces
  - Real-Time Network(s) and a Real-Time Operating System
  - Signal processing component (e.g. image processing)

- **The Unity-Component architecture**
  - Enable construction of a complete system based on a simple specification (textual DSL) and using prebuilt/library subcomponent.
  - Mimic ROS' communication and processing paradigm in embedded Software and Gateware (FPGA logic)
    - Publish/Subscribe (Topic based) & Services (RPC)
    - Nodes (Software as well Gateware based, i.e. SW-eNode and GW-eNode)
  - Migration/easy-reimplementation of e.g. timing critical ROS nodes from a PC to an Embedded Hard Real-Time capable MCU in an FPGA or even directly into logic.

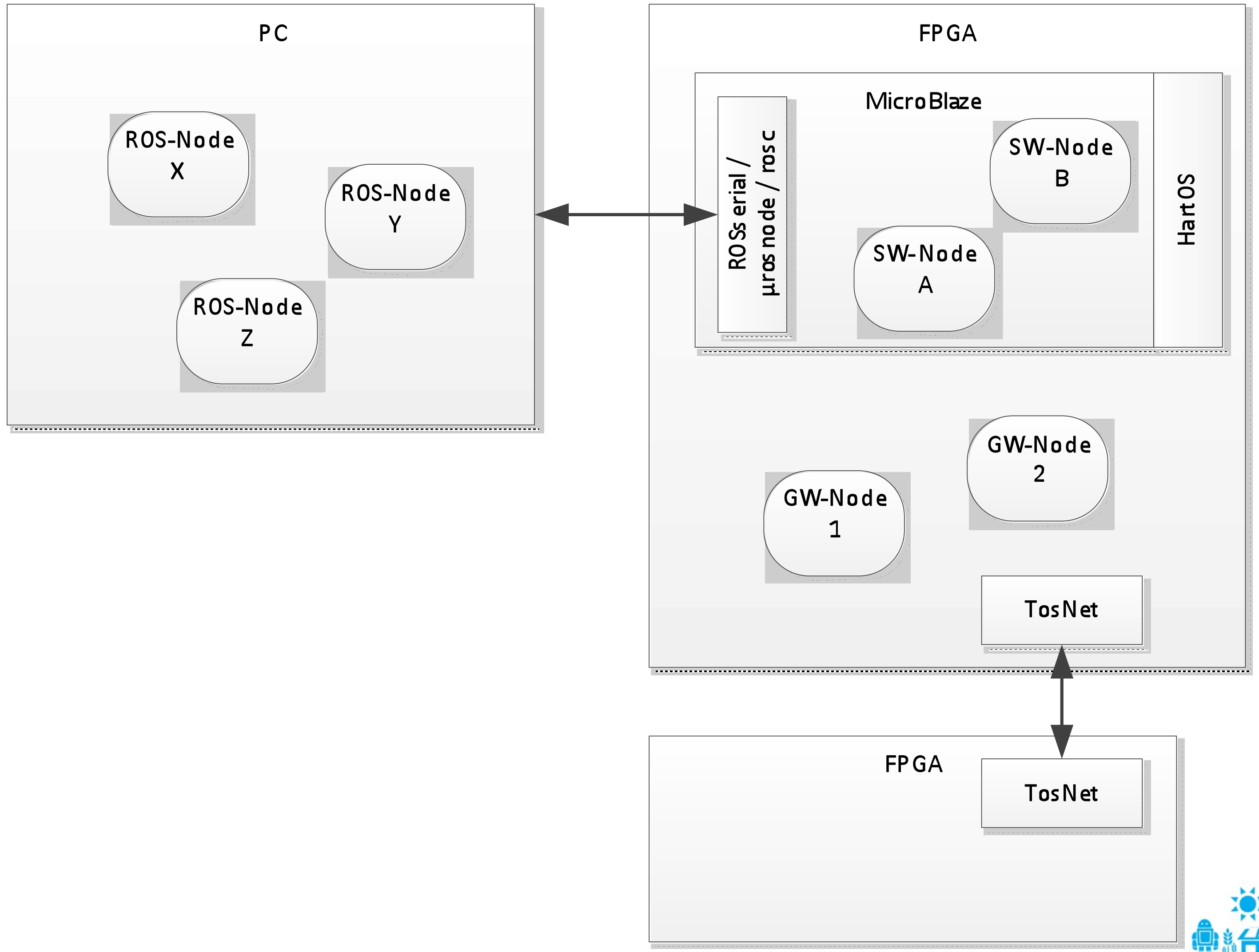
The Unity-Component architecture

- **Unity Component Architecture**
  - SW-eNodes will execute on one or more FPGA embedded MCU's
    - HartOS (Hardware RTOS) -> higher performance and improved real-time.
    - API that mimic ROS + a subset of POSIX for simplied kernel interaction.
  - GW based clock-cycle accurate Topic and Service-call infrastructure
  - TosNet for Real-Time networking between FPGA boards
  - Unity-Link as PC interface
    - Flexible interface selection: UART/USB communication, Ethernet, etc.
    - Scalable: can fit in very small FPGAs
    - Hard real-time (clock cycle accurate) GW-Protocol
    - Easy to interface with other high-level frameworks/tools
  - Could also easily use:
    - ROSSerial, uROSnode or ROSC executing on
      e.g. a softcore MCU, might not be as scaleable though.

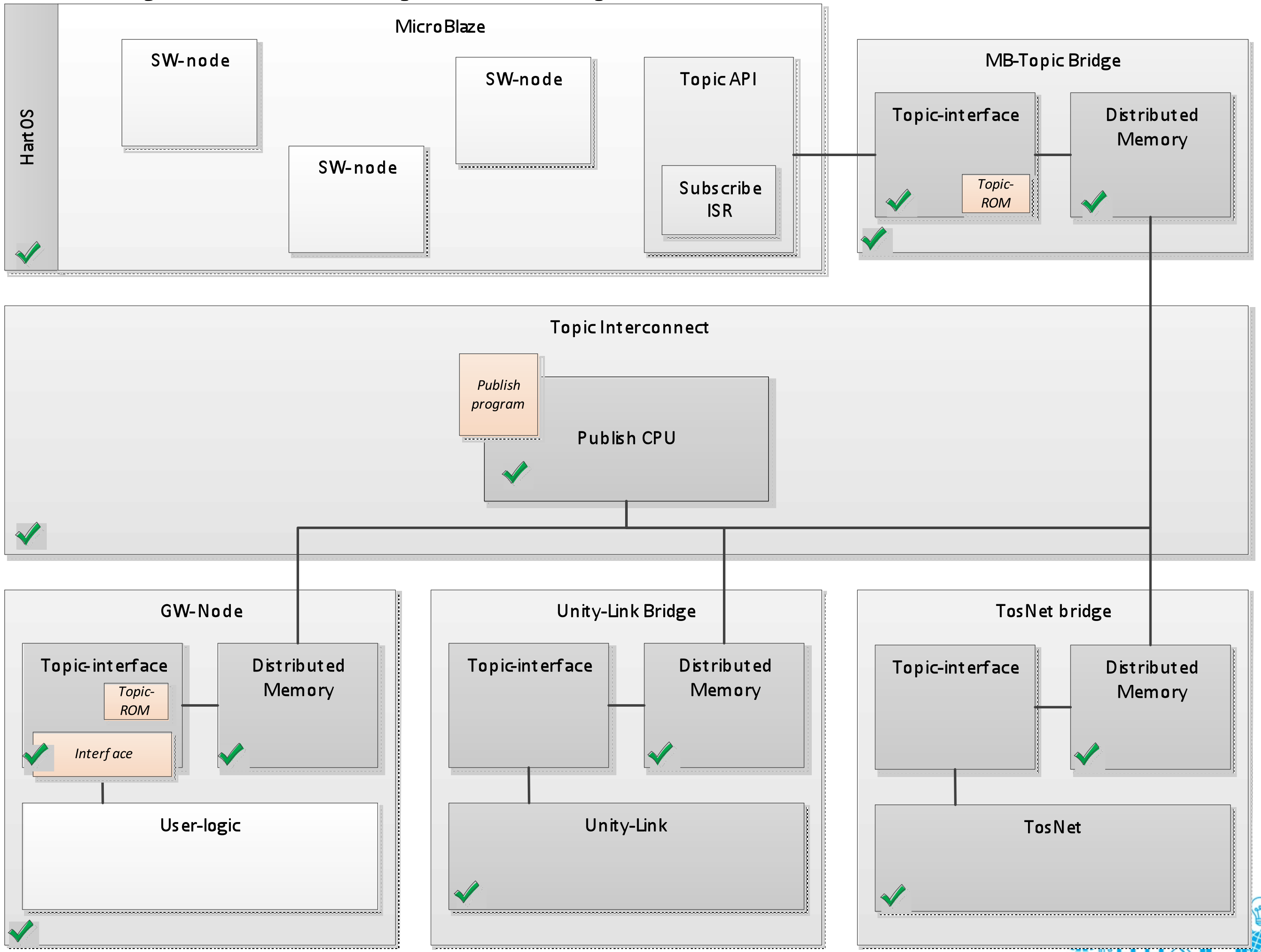
The Unity-Component architecture



The Unity-Component architecture



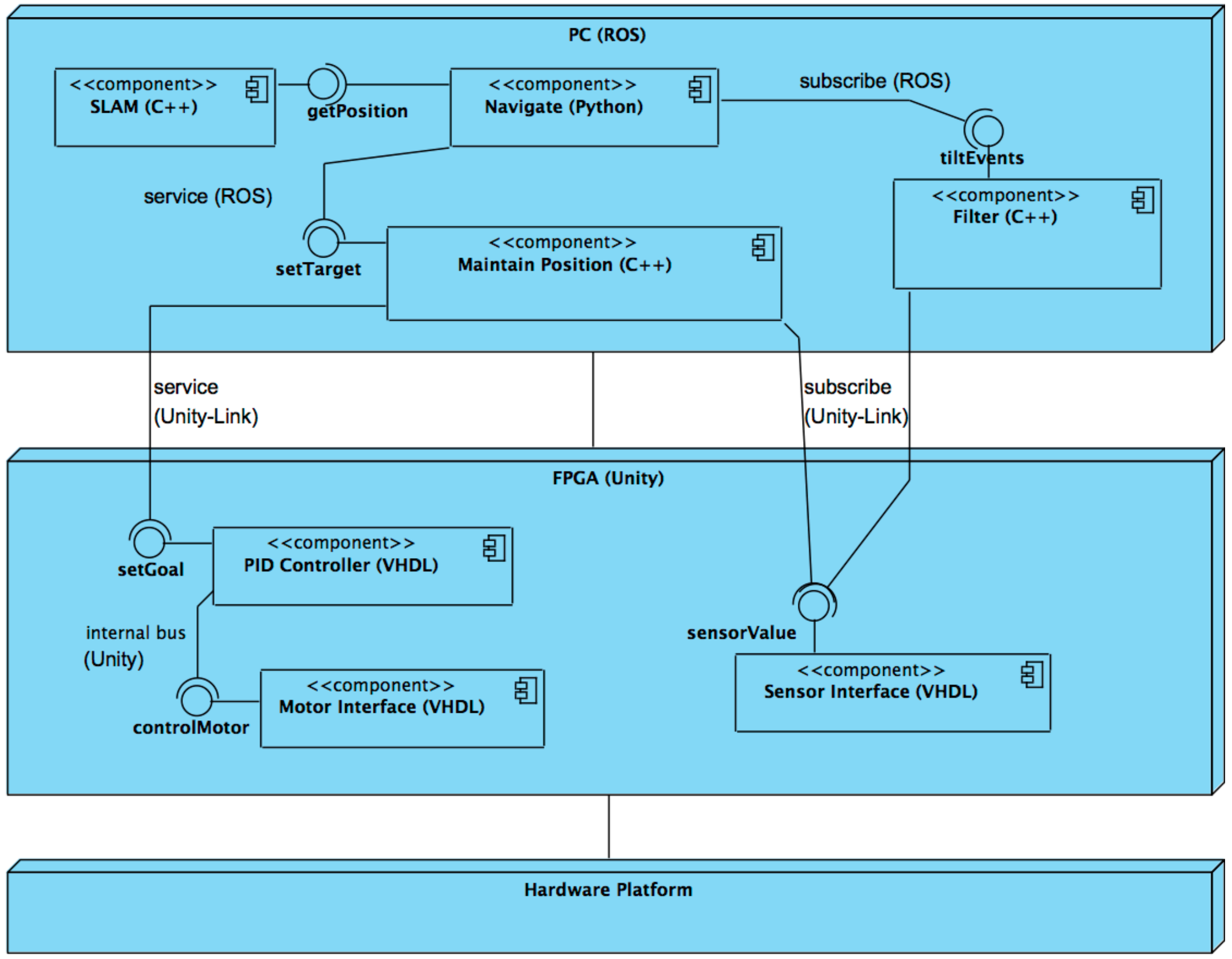
Unity-Comp. Topic Architecture



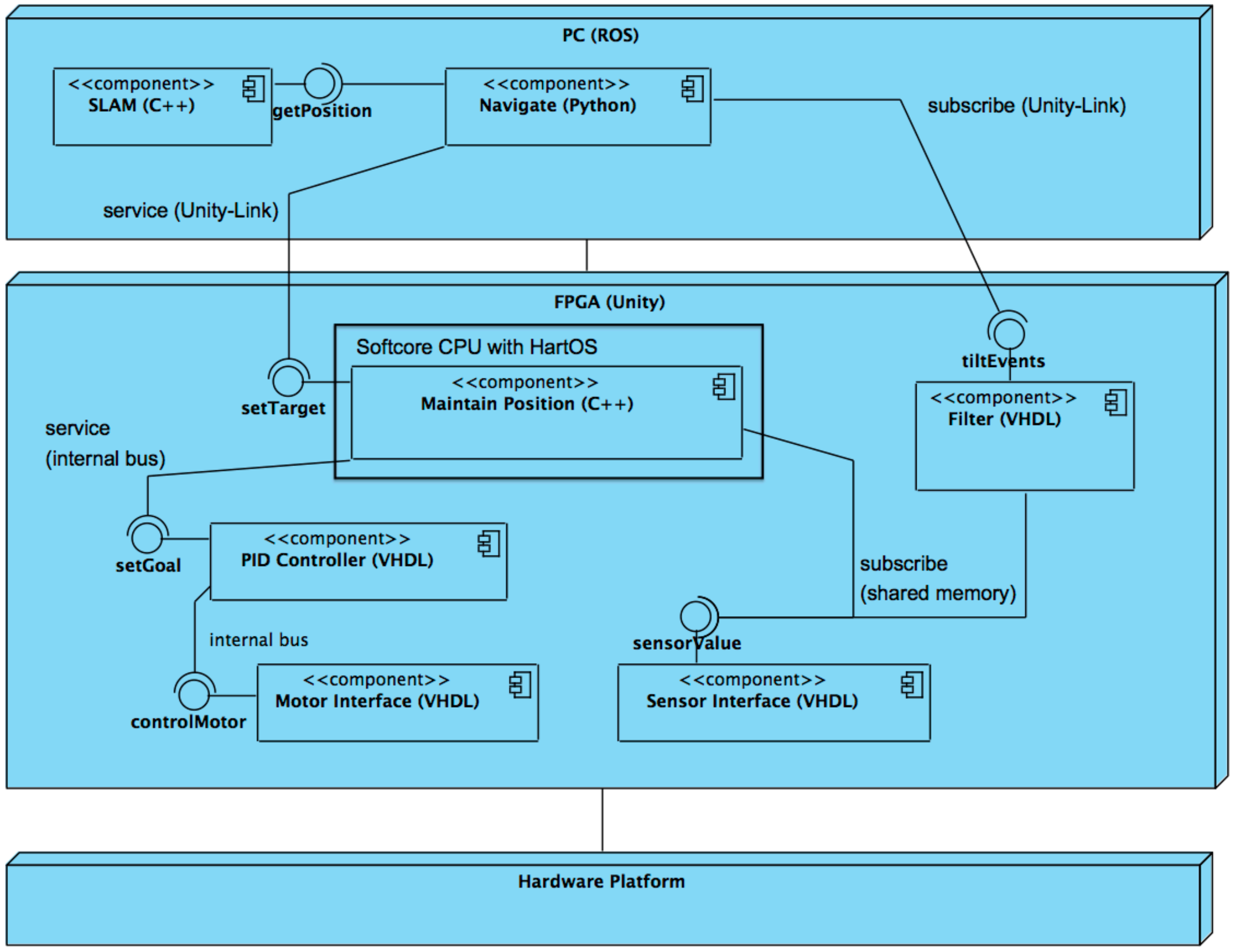
Unity-Comp. Service Architecture

- FIFO interfaces for each GW Service Provider/Consumer
  - MicroBlaze bridge
  - TosNet bridge
- Unity-Link naturally supports IRQ based publishing from FIFO's
- Interconnect
  - Wishbone master interfaces on Tx FIFOs
  - Wishbone slave interfaces on Rx FIFOs
  - Multi-master/slave wishbone-bus with round-robin arbitrations for simple applications.
  - Multi-master/slave crossbar-switch for applications requiring low latency.
  - Multiple bus' for highest performance with sparsely connected nodes
- Not implemented yet…

Unity Component Architecture



Unity Component Architecture


PC (ROS)
<<component>>
SLAM (C++)
getPosition
<<component>>
Navigate (Python)
subscribe (Unity-Link)
service (Unity-Link)
FPGA (Unity)
Softcore CPU with HartOS
setTarget
<<component>>
Maintain Position (C++)
tiltEvents
<<component>>
Filter (VHDL)
service
(internal bus)
setGoal
<<component>>
PID Controller (VHDL)
subscribe
(shared memory)
internal bus
sensorValue
controlMotor
<<component>>
Motor Interface (VHDL)
<<component>>
Sensor Interface (VHDL)
Hardware Platform