Wrocław University of Technology

Chair of Cybernetics and Robotics

Introduction to robotic programming environments

Mariusz Janiak

p. 331 C-3, 71 320 26 44

Programme content

- Introduction to robotic programming environments
- Component/agent based approach for distributed control system
- Communication protocols
- OROCOS framework
- ROS framework
- Mathematical libraries

Course credit terms

- To pass a course student has to obtain a lecture (**F1**) and a lab (**F2**) credits
- To obtain lecture credits stundent has to pass one of the following
    - end-term test – pass mark 51%
    - "Grandson" tests – short 3-minute, 1-2 questions, 3-5 minutes long, scored 0-3 points (0 – absent, 1 – blank page, 2-3 – answer to the questions), the grade
        - below 60% has to pass end-term test,
        - 4.0 from 60%,
        - 4.5 from $73,33\%$,
        - 5.0 from $86,66\%$.
- To obtain lab credits stundent has to actively participate in lab classes (obligatory presence, pass all laboratory exercises, preparation for the laboratory classes, self study) – pass mark 51%,
- Final grade

$$P = 0.4 * F1 + 0.6 * F2$$

Literature

- R. Simmons, D. Kortencamp, D. Brugali, *Robotics Systems Architectures and Programming*, Handbook of Robotics II-ed., Springer 2013
- A. J. A. Wang, K. Qian, *Component-Oriented Programming*, John Wiley & Sons, Inc., 2005
- D. Brugali, P. Scandurra, *Component-based robotic engineering (Part I)*, in Robotics & Automation Magazine, IEEE , vol.16, no.4, pp.84–96, December 2009
- D. Brugali, A. Shakhimardanov, *Component-Based Robotic Engineering (Part II)*, in Robotics & Automation Magazine, IEEE , vol.17, no.1, pp.100-112, March 2010
- R. Patrick Goebel, *ROS By Example HYDRO - Volume 1*, 2014
- Jason M. O'Kane, *A Gentle Introduction to ROS*, CreateSpace Independent Publishing Platform, 2013

Resources

- BRICS (http://www.best-of-robotics.org)
- OROCOS (www.orocos.org)
- ROS (www.ros.org)
- GAZEBO (http://gazebosim.org)
- VREP (http://www.coppeliarobotics.com)
- ACADO Toolkit (www.acadotoolkit.org)
- CasADi (https://github.com/casadi/casadi/wiki)
- Sundials (https://computation.llnl.gov/casc/sundials)
- Eigen (http://eigen.tuxfamily.org)
- ZeroMQ (http://zeromq.org)
- Protocol Buffers (https://developers.google.com/protocol-buffers)

Introduction to robotic programming environments

Fundamental questions

- How difficult is developing a complex robotic system?
- How to avoid re-inventing the wheel?
- Is there any methodology that support that process?
- What kind of tools should I use?

Introduction to robotic programming environments

Software for complex robotics systems usually is

- embedded,
- concurrent,
- real-time,
- distributed,
- data intensive,

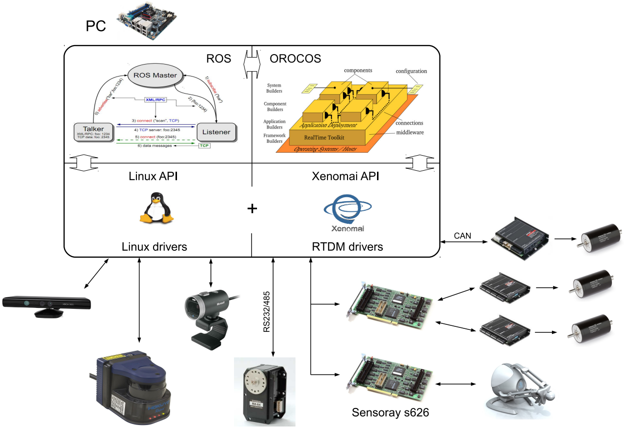
and must guarantee properties such as

- safety,
- reliability,
- fault tolerance.[1]

[1]D. Brugali, P. Scandurra, *Component-based robotic engineering (Part I)*, in Robotics & Automation Magazine, IEEE , vol.16, no.4, pp.84–96, December 2009

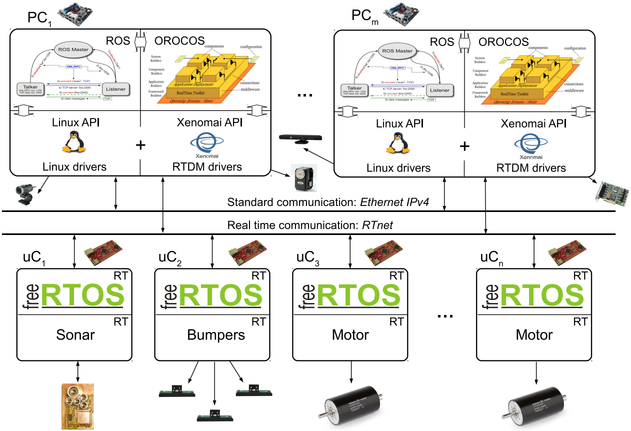
Introduction to robotic programming environments

Centralized vs Distributed



Introduction to robotic programming environments

Centralized vs Distributed



Introduction to robotic programming environments

Distributed System

- Multiple independent computers that appear as one
- Lamport's Definition
  "You know you have one when the crash of a computer you have never heard of stops you from getting any work done."
- *A number of interconnected autonomous computers that provide services to meet the information processing needs of modern enterprises.*

Introduction to robotic programming environments

Distributed System (cont.)

- Multiple Autonomous Computers
  - each consisting of CPU's, local memory, stable storage, I/O paths connecting to the environment
  - geographically distributed
- Interconnections
  - some I/O paths interconnect computers that talk to each other
- Shared State
  - shared memory is not required
  - systems cooperate to maintain shared state
  - maintaining global invariants requires correct and coordinated operation of multiple computers

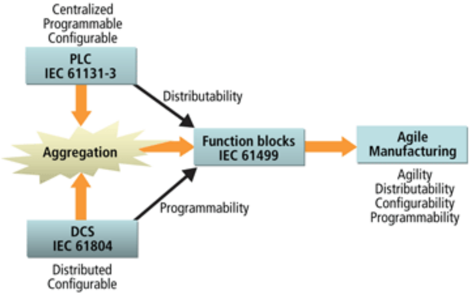
Introduction to robotic programming environments

Communication issues

- Fast, reliable physical communication interface
- Real-time constraints (RTOS required) – software vs hardware
- Communication protocol many-to-many – typically message oriented publish/subscribe pattern.
- Quality of Service (QoS)
- Data representation – serialization
- Portability and availability (different architectures, general computing vs embedded, OS vs bare-metal)

Introduction to robotic programming environments

Standard IEC 61499 – origin

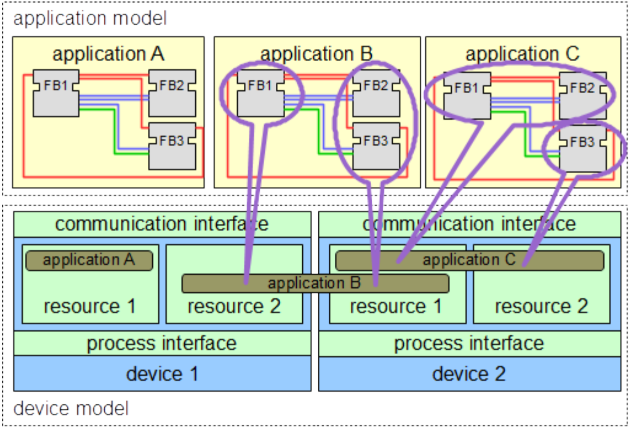


2

---

2 www.isa.org

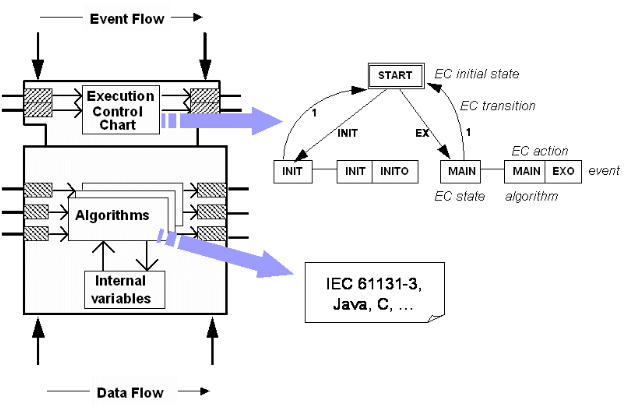
Standard IEC 61499 – architecture



3

3 wikipedia.org

Standard IEC 61499 – component



4

Introduction to robotic programming environments

BRICS: Best Practices in Robotics. *The prime objective of BRICS is to structure and formalize the robot development process itself and to provide tools, models, and functional libraries, which help accelerating this process significantly.*[5]

- European project aimed at find out the "best practices" in the developing of the robotic systems
- Investigate the weakness of robotic projects
- Investigates the integration between hardware and software
- Design an IDE for robotic projects – BRIDE
- Define showcases for the evaluation of project robustness with respect to BRICS principles

[5] http://www.best-of-robotics.org/

Introduction to robotic programming environments

A component

- is a binary unit of deployment,
- implements well defined interfaces,
- provides access to predefined set of functionality,
- may be customized by set of parameters without access to source code.

Component-Based Development

- Software development from pre-produced parts
- Modularity
- Re-usability
- Easly maintaining and customizing to produce new functions and features

Introduction to robotic programming environments

Robotics middleware[6]

- Simplifying the development process
- Support communications and interoperability
- Providing efficient utilization of available resources
- Providing heterogeneity abstractions
- Supporting integration with other systems
- Offering often-needed robot services
- Providing automatic recourse discovery and configuration
- Supporting embedded components and low-resource-devices

[6] N. Mohamed, J. Al-Jaroodi, I. Jawhar, "Middleware for Robotics: A Survey," in Robotics, Automation and Mechatronics, 2008 IEEE Conference on , vol., no., pp.736-742, 21-24 Sept. 2008

Introduction to robotic programming environments

Robotics middleware

- OROCOS
- ROCK
- ORCA
- YARP
- OpenRTM
- OpenRave
- ROS
- Player
- ...

Introduction to robotic programming environments

- *Open Robot Control Software*
- Open source license
- Supported languages: C++, LUA and native scripting
- *Component Oriented Programming* model
- Framework components
    - *Orocos Toolchain*
    - *Kinematics & Dynamics Library*
    - *Bayesian Filtering Library*
- Real-time control and communication
- Integrated with Xenomai and ROS

www.orocos.org

Introduction to robotic programming environments

- *Robot Operating System*
- Open source license
- Supported languages: C++, Python
- Agent based programming model
- Message passing
  - *Topics* – publish/subscribe model
  - *Service* – remote operation
- Name and Parameter Services
- Application building blocks: coordinate system transform services, visualization tools, debugging tools, robust navigation stack, arm path planning, object recognition, ...

www.ros.org

Introduction to robotic programming environments

Mathematical libraries

- Algebra – LAPACK, Eigen, uBlas, GSL, MPL, MKL, ACML, . . .
- ODE – Sundials, Odeint, GSL, Acado Toolkit, . . .
- Optimization – IPOPT, Acado Toolkit, CasADi, . . .
- Automatic differentiation – ADOL-C, CasADi, Acado Toolkit, ad, . . .

Introduction to robotic programming environments

Robotics libraries (a brief overview)

- motion planning – MoveIt, MPK, OMPL, OOPSMP, MSL, ...
- vision – OpenCV, PCL, LIBELAS, LIBVISO, VTK, ...
- navigation – ROS Navigation Stack, MRPT, ...
- SLAM – MRPT, RobotVision, gmapping, ...
- simulation – Bullet Physics, ODE, AMD Havok, NVIDIA Phisics, ...
- control – ros_control, OpenRTDynamics, Robotics Toolbox, ROCK ...
- ...

The End

Thank you for your kind attention.