


**Software development in robotics:  
frameworks and tools**

*Daniele Calisi and Daniele Nardi*

DIPARTIMENTO DI INFORMATICA  
E SISTEMISTICA ANTONIO RUBERTI



SAPIENZA  
UNIVERSITÀ DI ROMA

### Complexity of robotic software development (1)

- Complex heterogeneous distributed systems
  - Complex algorithms but limited computation and memory
  - Requires data sharing mechanism (middleware)
  - Concurrent execution of processes/threads
- The robot is “embodied” in the real world
  - The world is uncertain: sensor readings, action outcomes, unexpected events, etc.
  - Physics cannot be delayed: strict time constraints
- Challenging problems
  - Trial and error
  - Method/algorithm tuning
  - Fast prototyping

Software development in robotics

2

### Complexity of robotic software development (2)

- Different hardware devices
  - Different protocols and interfaces
- Inherently complex setting
  - e.g. visualizing sensor data requires to:
    - Connect to the sensor
    - Understand and implement the protocol
    - Build a GUI
- A lot of system engineering
  - Challenging programming
  - Debugging and testing
- Heterogeneous Team of developers

Software development in robotics

3

### Software frameworks

- A **software framework** is an abstraction in which common code providing generic functionalities that can be selectively overridden or specialized by user code
- Frameworks are similar to *software libraries* in that they are reusable abstractions of code wrapped in a well-defined **API**.
- Unlike libraries, however, the overall program's *flow of control* is not dictated by the caller, but by the framework. This *inversion of control* is the distinguishing feature of software frameworks [Wikipedia]

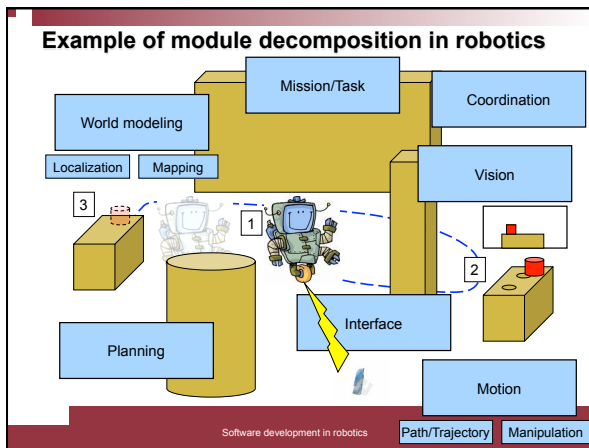
Software development in robotics

4

### Modularity

- Divide-et-impere approach
  - Common engineering method
  - A complex problem can be often subdivided in simpler sub-problems
- Module-level tests and debugging
  - Local search space for bugs
- Key features
  - Abstraction and common interfaces
  - Code reuse
  - Encapsulation
  - Decoupling

Software development in robotics 5



### Software frameworks for robotics

- Aims
  - Promote standard design techniques
  - Code reusability (components)
  - Ready-to-use design techniques
  - Rapid prototyping
  - Concurrent engineering
- Examples
  - OROCOS (EURON project), CLARAty (NASA), OpenRTM-aist (Japanese project), YARP, MS Robotic Studio, ...
  - Orca, Player/Stage, MARIE, MOAST, ...
  - ROS (Willow Garage)
  - OpenRDK (DIS RoCoCo ...)
- Main elements of a framework
  - Concurrency model
  - Information sharing model
  - Libraries and tools
  - Interoperability

Software development in robotics 7

### Concurrency model

- Call-backs
- Processes
- Threads ←

**Process**

Module

Module

```

// a callback is called by a scheduler
void callback() {
  // do your work quickly
  // and return the control
  // to the scheduler
}
                    
```

**Process**

Module

Frameworks that use threads:  
OpenRDK,  
(OROCOS),  
(OpenRTM)

Software development in robotics 8

### Information sharing model

- **Data ports** ←
- **Blackboard** ←
- **Services**
  - Use of third-party middleware
    - e.g., CORBA, ICE, OMG DDS
    - Ready-to-use
    - Different goals
  - Developing ad-hoc middleware
    - It's a complex task
    - Allows for application-oriented tuning

Software development in robotics 9

### Libraries and tools

- Libraries reduce programming time
  - Frameworks include libraries for geometrical computations, control, filesystem utilities, operations on maps, images, etc. ←
  - Often external libraries are used (e.g., OpenCV, libgsl, libxml, ImageMagick, etc.) ←
- Tools speed-up development and debugging phases
  - Graphical tools for debugging and inspection ←
  - Simulators or connection to simulators ←
  - Logging and replaying ←
  - Profiling ←

Software development in robotics 10

### Interoperability

- No "best" framework, it depends on
  - Different applications (service robotics, multi-robot teams, etc.)
  - Different focus (low-level real-time control, high-level artificial intelligence, machine learning, image processing, etc.)
  - Different people (students, researchers, control experts, etc.)
  - Different set of robot compatibility (mobile robots, robotic arms, frameworks for a single model of robot)
- Interoperability
  - *The ability of heterogeneous systems to suitably exchange information, using common protocols and abstractions*

Software development in robotics 11

### An example of frameworks comparison

<ul style="list-style-type: none"> <li>■ OpenRDK           <ul style="list-style-type: none"> <li>□ Real-time not supported</li> <li>□ Compatibility with a large set of robots</li> <li>□ Multi-threaded process</li> <li>□ Module development guided/constrained</li> <li>□ Pluggable inter-process communication (rdk, dds, http, file, etc.)</li> <li>□ Core libraries for robotic applications</li> <li>□ Small community</li> <li>□ GPLv3 license</li> <li>□ ...</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>■ ROS           <ul style="list-style-type: none"> <li>□ Real-time not supported</li> <li>□ Compatibility with a large set of robots</li> <li>□ One module = one process</li> <li>□ Free module development</li> <li>□ Proprietary protocol for inter-process communication</li> <li>□ Large set of libraries for robotic applications</li> <li>□ Large community</li> <li>□ BSD license</li> <li>□ ...</li> </ul> </li> </ul>
---	---

Software development in robotics 12

### A short history of OpenRDK

- SPQR-RDK (first commit to the CVS repository)
  - April, 2nd 2003
- SPQR-RDK 2
  - September, 30th 2005
- OpenRDK (SourceForge)
  - February, 25th 2008

Software development in robotics 13

### Questions

## Questions?

We are on SourceForge  
<http://openrdk.sourceforge.net>

Software development in robotics 14