Background

Construction of systems requires “glue” for integration
That includes
- Communication framework
- Components in “standard libraries”
- A control/synchronization model
- Tools to support development

Communication

- Language neutrality
- Handling of structured data
- Efficient (latency / bandwidth)
- Multiple control models
  - Push, Pull, Synchronize, Subscription
Std. Component Libraries

- Interfaces to standard actuators / sensors
  - Odometry, Camera, IR sensors, Sonar, LIDARs, ...
- Reference implementations
  - Mapping, Feature Extraction, Planning, Navigation, ...

Tools for Development

- Integration with UML for other design methodology
- Code templates / reference models
- Integration with debuggers for efficient development
- Facilities to accommodate logging
- Methods for simulation
- Methods for visualization of state

Outline

1. Introduction
2. MSRS Overview
3. A few details
4. Competitors
5. Summary

A note on robot economy

<table>
<thead>
<tr>
<th>Component Technologies</th>
<th>Turn over in B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMEs</td>
<td>2.8 B$</td>
</tr>
<tr>
<td>Robotics Manufacturers</td>
<td>3.5 B$</td>
</tr>
<tr>
<td>System Integrators</td>
<td>1.5 B$</td>
</tr>
<tr>
<td>Large Enterprises</td>
<td>10 B$</td>
</tr>
<tr>
<td>Universities and R&amp;D institutions</td>
<td>5 B$</td>
</tr>
</tbody>
</table>

Source: EUROP / Eurostat / World Robotics
Microsoft Robotic Studio

- An attempt for Microsoft to enter the systems integrator market
- A framework for process communication and synchronization
  - CCR - Concurrency and Coordination Runtime
  - A client model for application developments
    - DSS - Decentralized Software Services
      - "Physics" based simulation
        - Based on the AGEIA package
      - A framework for visual programming (PnP)

CCR

- Concurrency and Coordination Run-Time
- Based on .NET 2.0
- Asynchronous message passing framework
- Separating communication and computing
- Light weight process management
- Single DLL for interfacing to a number of languages
  - C#, C++, Python, VB
- Inspired by a web based services
- Used in applications such as MySpace

DSS

- Decentralized Software Services
- Defines standard I/O methods for services
  1. definition of a standard service identifier
  2. setup of a “service” contract for interaction
  3. object to capture service “state”
  4. methods to handle service requests
  5. error recovery methods
- Includes yellow page service
- Standard I/O models (Push, Pull, Query, Subscription)
Web based status

A trivial example

A little realism
**Communication**

- Client Service
- Target Service
- Subscriber
- Publisher

**Visual Programming**

![Visual Programming Diagram]

**MSRS Simulation**

- AGEIA / PhysX has a physics-based simulation engine
- Exploits graphics hardware
- Simplified robot models
- Adequate for basic prototyping

**Standard Interfaces**

- Has interfaces for standard robots
  - Pioneer, iRobot Create, Lego NXT, Parallax, . . .
- Interfaces to standard sensors
  - USB camera, SICK LMS-221, Canon IR, Garmin GPS, . . .
- Number of supported units has grown significantly the last year
Getting to know MSRS

- Rich set of tutorials on the web
- CCR basics, DSS basics, to understand the basis
- C# knowledge is recommended
- Use Visual Studio 2005 Express Edition (Free)
- Sample Tutorials on simple actions and full systems
- KUKA Education Framework
  - Joint Control, Cartesian Control, Mobile Systems

Competitors

- OROCOS Open Robot Control System
  - Manipulator Oriented Open Source Control
- ORCA Open Robot Control Architecture
  - Mobile Platforms software with rich library
- Saphira Mobile Robotics / Pioneer System
- Player-Stage Open Source Device Library
  - Most widely used OSS for robotics
- ERSP Evolution Robotics Software Package
  - The objectives are the same as for MSRS
  - Comprehensive package - cost is a challenge

Short Comparison

<table>
<thead>
<tr>
<th>System</th>
<th>Architecture</th>
<th>Language</th>
<th>Devices</th>
<th>Doc</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSRS</td>
<td>HD</td>
<td>C#; VH, C++, Python</td>
<td>Some</td>
<td>++</td>
</tr>
<tr>
<td>OROCOS</td>
<td>React</td>
<td>C++</td>
<td>Manip</td>
<td>++</td>
</tr>
<tr>
<td>ORCA</td>
<td>HD</td>
<td>C++ / Java</td>
<td>Mobile</td>
<td>++</td>
</tr>
<tr>
<td>Saphira</td>
<td>HD/SHM</td>
<td>C, C++</td>
<td>Pioneer</td>
<td>+</td>
</tr>
<tr>
<td>Player</td>
<td>C++, Java...</td>
<td>MANY</td>
<td>MANY</td>
<td>+++</td>
</tr>
<tr>
<td>ERSP</td>
<td>HD</td>
<td>C++, Python</td>
<td>ER1...</td>
<td>++</td>
</tr>
</tbody>
</table>
Outline

- Introduction
- MSRS Overview
- A few details
- Competitors
- Summary

Summary

- Brief introduction to MSRS
- Systems integration is key to success
- MSRS is one option and the future will be interesting
- C# / .NET knowledge a significant advantage
- There is a learning curve but now there are good tutorials and the documentation is OK.
- For 8803 there are standard drivers for all devices

Questions?

Questions?