OPRoS-based RILS (Robot in the Simulation System)

SDIR workshop at ICRA2013
2013.05.06
Mi-sook Kim, Hong Seong Park, Jun Young Jang
Hong Seok Kim, Sang Hoon Ji

* This work was partially supported by MEST, DGIST, and MKE
Contents

- OPRoS
- OPRoS Component
- OPRoS Framework
- OPRoS IDE
- OPRoS RILS
What is Open Platform for Robotic Services (OPRoS)

- **Open Platform for Robotic Services**: Open Source Robot Platform
  - Period: 2007~present
  - License: LGPL and OPRoS: www.oprospace.org
  - Languages: C++, Java (IDE)
  - Supported Compilers: GCC, VS2008, VS2010
  - Supporting OS: Linux (Kernel 2.6.38 and higher), Windows (XP, Vista, 7)
    - Experiment: Android, iOS, Embedded Linux/Windows (ARM9), QNX, RTX

- **Purpose**
  - Fast and Easy Robot Software Development
    - Recycling OPRoS Components increase source code reusability
    - Connecting OPRoS Components replace socket programming implementation
  - Small Size of Framework: Minute Size - 300KB, Supporting IDE - 1MB
OPRoS Component

- 11 call-back functions, and 6 states

Service Port
- Blocking Service
- Non Blocking Service (planned)

Data Port
- Periodic data
- Asynchronous data

Event Port
- Event ID based data
OPRoS Component

- Elements of Component: Component profile, component executable file (.dll, or .so), device API (.dll, or .so)

- Development: Parallel development and code reusability
- Maintenance: Minimum changes

Example of OPRoS Component reusability
OPRoS Framework

- Component Execution Engine, TPL engine, and Managers ensures Operation of OPRoS Components
- Server extends functionalities
  - Global Repository, Proxy Component execution roles
OPRoS Component Development Flow

- **OPRoS Component Editor**
  - Create, Modify, and package OPRoS Component

- **OPRoS Component Tester**
  - State, Unite, Interface test for Component (Optional)

- **OPRoS Component Composer**
  - Create OPRoS Application

- **OPRoS Task Editor**
  - Create OPRoS Task Script (Optional)

- **OPRoS Simulator**
  - Test OPRoS Application with Simulator

- **OPRoS Robot**
OPRoS IDE: OPRoS IDE Packages

• User Tailored IDE Type1 & Type2
  – Type1 IDE is for general users
    • Component Composer, Task Editor, Debugger, Simulator
  – Type2 IDE is for C++ developers
    • Component Editor & Composer, Task Editor, Debugger, Simulator
  – One click & one menu movement between tools

• Component Editor
  – Created component template reduces programming loads

• Component Composer
  – Creating robot action model with connecting components
    • behavior model, reactive model, mixed model, user creative model, etc

• Task Editor Functionality Enhancement
  – Creating outline of program with graphic tool, then writing script program
OPRoS RILS: Robot-in-the-loop-simulation

- RILS is a simulation method to operate a real hardware device and a virtual device at the same time
- Real application
  - Participate as a virtual device
  - Virtual condition control hardware device
  - Imitate the real devices
OPRoS RILS: Robot-in-the-loop-simulation

• Use virtual laser scanner sensor because of lack of real sensor
• Virtual fence is required for robot control
OPRoS RILS: Robot-in-the-loop-simulation

- Imitate the robot for monitoring or security
• Same OPRoS Component is used in the simulator and the robot: WheelControllerComp
• Different properties in the component profile and different Device APIs are used for different devices
Conclusion

• OPRoS RILS uses virtual devices and hardware devices at the same time
• Virtual device is replaced with the hardware device
• Virtual device controls hardware device because of virtual condition
• Hardware device controls virtual device for monitoring purposes
• For homogeneous devices, switching device APIs are enough to use the same application, and the same component
Questions & Discussions
OPRoS IDE: Component Editor

- OPRoS Component templates
  - Component Profile: xml format
  - Component source code templates
- Drag-and-drop based graphical programming
- GNU C++ and VC++ compilers
  - Makefile
  - VS project files
- Component Packaging with manifest file

```xml
<properties>
  <property name="dllName" type="string">Simulator_LaserScanner_API.dll</property>
  <property name="deviceName" type="string">Simulator_LaserScanner</property>
  <property name="size" type="int">1</property>
  <property name="maximumStep" type="int">769</property>
  <property name="startStep" type="int">128</property>
  <property name="endStep" type="int">640</property>
  <property name="delStep" type="int">8</property>
</properties>
```
OPRoS IDE: Component Composer

- Application Profile
  - List of subcomponents
  - Connection information between components
- Deployment of application into multi boards
- Graphical Execution of application
OPRoS IDE: Task Editor

- Model files
  - Binding component services calls and scripts model
  - FSM based script for robot programming
- Graphical blocks for generating script
OPRoS IDE: Simulator

- Runs on Windows / Linux (MORSE)
- For application testing, instead of hardware manufacturer’s driver, use simulator device api modules