CONFERENCE DIGEST

ロボティクス・メカトロニクス講演会2010

2010 JSME Conference on Robotics and Mechatronics

ROBOMEC2010 in ASAHIKAWA

ロボティクス・メカトロニクス・フロンティア・ビッグバン

Robotics・Mechatornics・Frontier・Big-Bang

June 13 Sun. - 16 Wed. , 2010

Asahikawa TAISETSU Arena

主催　社団法人 日本機械学会 ロボティクス・メカトロニクス部門

The Japan Society of Mechanical Engineers, Robotics and Mechatronics Division

rtcshell: Command-line tools for OpenRTM-aist

Geoffrey BIGGS and Noriaki ANDO and Tetsuo KOTOKU

Intelligent Systems Research Institute
National Institute of Advanced Industrial Science and Technology (AIST)

Component-based software is a major recent design trend in robotics. It brings many benefits to system design, implementation and maintenance. The management of such systems often depends on graphical tools. These tools are powerful and provide a rapid way to layout component networks. However, they also typically require considerable resources to run. This is not ideal in robotics, where low-resource environments are common. We have created a set of command-line tools for use with the OpenRTM-aist component-based robot middleware. The tools follow the UNIX philosophy of simplicity and aggregation. These tools allow whole component-based systems to be created, managed and monitored from a command-line. They are ideal for use in environments where a graphical interface is not available. By combining tools together using shell scripts, more complex functionality can be easily created.

Key Words: Robot programming systems, component-based architectures, RT-Middleware

1. Introduction

A key part of using component-based robot software, such as architectures such as OpenRTM-aist [1] and ROS [2], is interacting with the component network that makes up the software system. Interaction may be via a specialised tool designed to monitor a specific network of components, or a generic tool for the component architecture upon which the software system is built may be used. In some cases, no such tool may be available, with all interaction via text files.

This paper presents a set of tools for managing OpenRTM-aist-based systems, known as RT-Systems. These tools differ from the usual approach in that they apply a file-system abstraction to the components for interaction, they are command-line tools and they follow the UNIX philosophy of doing one thing well, using aggregation to add power. It shows that such a collection of simple tools gives great flexibility and ease of use to developers creating and managing component-based robotics software, where the interaction method may be constrained by unique environmental factors.

2. OpenRTM-aist

OpenRTM-aist [1] is a component-based architecture for intelligent systems. The central concept in OpenRTM-aist is the RT-Component. An RT-Component's internal state can be monitored and controlled. In order for a component to begin executing, it must be placed in the *Active* state. Execution can be terminated by returning it to the *Inactive* state. If an error occurs in the component, it moves to the *Error* state. A component network is formed by making connections between ports of components.

OpenRTM-aist uses CORBA [3] as its communications layer. Components must register on a known name server. Typically, components are organised on the name server using naming contexts below the root context in order to provide hierarchical categorisation.

3. Interacting with RT-Systems

OpenRTM-aist uses a tool, known as RTSystemEditor, for interacting with and managing the components of RT-Systems. This tool can be used both at system-construction time to create a component network, and at run-time to monitor the health of and to alter the component network.

Unfortunately, the use of a graphical tool has certain requirements that cannot always be met:

- A graphical tool uses valuable resources in a low-resource environment, such as a robot's internal computer.
- Running the graphical tool from a remote computer may not be possible due to a lack of a network connection..
- Graphical tools cannot easily be scripted to perform repetitive tasks.

Inspired by the difficulty in using RTSystemEditor on our outdoor robots, we have experimented with tools for OpenRTM-aist that do not rely on a graphical interface. These tools treat RT-Components as part of a pseudo-file system structured around the naming services to which they register.

4. Shell utilities for OpenRTM-aist

The set of tools created are collectively called rtcshell. The tools follow the UNIX philosophy of doing one thing and doing it well.

4.4.1 File system layout

The pseudo-file system provides information about and addressing of all known OpenRTM-aist objects. Its general structure is illustrated in Figure 1. All paths branch off from a single root node, `/`. Below this root are all the known root naming contexts, each of which represents a naming service, and which contain other naming contexts (treated as directories) and "files."

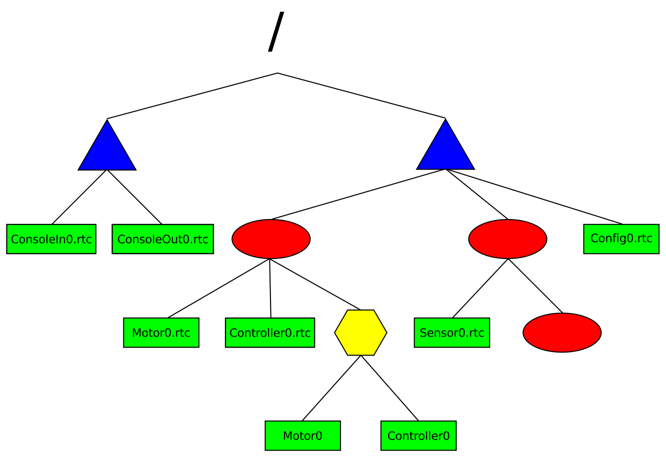

Figure 1: The general structure of the pseudo-file system used by the rtcshell utilities, dubbed the "RTC Tree". The root node is at the top of the file system tree. The blue triangles are naming service nodes. Red ovals are naming contexts below a root context. Green boxes are RT Components. The yellow hexagon is a manager. Below it are aliases to the two RT Components it is managing.

The files of the pseudo-file system are RT-Components. A file "contains" information about the component it represents, such as its state and its available ports.

Manager objects, used by OpenRTM-aist to manage running component instances, are also present in the tree. They are treated as directories containing *aliases* to the components they manage.

4.4.2 Navigating the pseudo-file system

rtcshell includes some commands that allow the user to change their focal point in the pseudo-file system. `rtcwd`, which behaves like the standard `cd` command, is used to change the current working directory. The current working directory can be displayed using `rtpwd`.

The `rtfind` command searches the pseudo-file system for nodes matching given search criteria, such as being of a specific type or matching a file name.

The `rtls` command lists a directory's contents. Like its standard namesake, this has both short and long forms. The long form will also display some brief useful information about components, such as their current state. It is useful for monitoring the state of running components.

4.4.3 Viewing component information

The component "files" in the pseudo-file system contain information that can be viewed with the appropriate tool. These are:

- `rtcat` prints out information contained within a component, such as the component's state and its ports.
- `rtconf` is used to display and alter component parameters, including the active parameter set.
- `rtprint` displays the data being sent over an output port.

4.4.4 Manipulating components

The following commands are used to change the state of components:

- `rtact` activates a component, causing it to being executing.
- `rtdeact` deactivates a component, halting its execution.
- `rtreset` resets a component, moving it back to the *Inactive* state after an error has interrupted execution.

As mentioned earlier, the `rtls` or `rtcat` commands can be used to monitor component state.

Connections between ports are managed using three commands:

- `rtcon` connects two ports together. Ports of a component are specified at the end of the component's path, separated by a colon.
- `rtdis` removes a connection between two ports, or all connections from a port or component.
- `rtinject` injects data into a port. This is useful to send data to a component without using another component.

`rtcat` can be used to check the connections of a port.

4.4.5 Managers

Managers are used to deploy and manage components. The `rtmgr` tool is used for working with managers. With it, developers can load and unload modules, and create and destroy component instances. `rtcat` is also capable of working with manager nodes in the pseudo-file system, printing information about the loaded modules, managed components, and so on.

5. rtsshell

rtcshell only works with individual components. An additional set of tools was created for manipulating entire RT Systems, "rtsshell". They work with RTSProfile-format XML files, which describe RT Systems.

- `rtcryo` examines all components on all known naming services and creates an RTSProfile for those with connections.
- `rtteardown` uses an RTSProfile file to remove all connections in an RT System.
- `rtresurrect` is the opposite of rtcryo and rtteardown. It reads an RTSProfile-formatted file and uses it to reconstruct a complete RT System.
- `rtstart` is the tool responsible for activating the components of the RT System.
- `rtstop` stops the RT System by deactivating the involved components.

6. Discussion

The command line tools maintain the UNIX philosophy of doing one thing and doing it well. Each tool is specialised for an individual task. However, while each tool is simplistic by itself, the collection as a whole is both flexible and powerful. This is particularly the case when they are combined with a scripting system such as a shell scripting language. As with all UNIX tools, aggregation gives these simple tools their power.

The tools are designed with this in mind. For example, combining the standard UNIX `watch` command and `rtls -l` gives a continuously-updating display of component state, allowing their health to be monitored in real time. A list of all output ports of a component can be obtained by combining `rtcat` and `grep`. The shell script **for** command can be combined with `rtfind` and `rtact` to activate all components matching a given name specification.

rtcshell's strength is in quickly creating small systems for experimentation, for managing both large and small RT Systems, and for automation of common tasks. No other system currently matches all of rtcshell's functionality.

7. Conclusions

This paper has described a set of simple, single-purpose command-line tools for interacting with and managing RT Components and component networks based on the OpenRTM-aist robot middleware. The tools treat a set of known components as though part of a file system. The user can navigate around and inspect known components in a console. This form of interaction is particularly well suited to the low-resource environments that are commonly found in robotics. The tools can also be scripted using standard shell scripting facilities. We believe that such a set of command-line tools adds additional usability to robot-oriented component-based software architectures.

References

[1] N. Ando, T. Suehiro, K. Kitagaki, T. Kotoku, and W.-K. Yoon, "RT-middleware: distributed component middleware for RT (robot technology)," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, August 2005, pp. 3933–3938.

[2] (2009) ROS.org. [Online]. Available: http://www.ros.org

[3] M. Henning and S. Vinoski, *Advanced CORBA Programming with C++*. Addison-Wesley Professional, 1999.