# Development of R-Cubed Manipulation Language

## The design of an RCML 2.0 system

Dairoku Sekiguchi, Wei-Chung Teng, Yasuyuki Yanagida, Naoki Kawakami, and Susumu Tachi
School of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, JAPAN
*{dairoku, waldo, yanagida, kawakami, tachi}@star.t.u-tokyo.ac.jp*

**Abstract**

The concept of R-Cubed (Real-time Remote Robotics: R$^3$) aims to provide a way to telexist anywhere in the world by controlling remote robots over the network. RCML (R-Cubed Manipulation Language) is considered to be a language for describing the interface for controlling remote robots in an R-Cubed concept. RCML 1.0 is an extension of VRML97 and uses a PROTO node, which is an extension node of VRML97. Through the experimental implementation of an RCML 1.0 system, two design problems were revealed. One is a limitation on implementation, and the other is a separation of user interface and control information definition. To overcome these problems, we designed a new version of the RCML system. This paper proposes a new design of the RCML system called RCML 2.0. In RCML 2.0, we introduced a language RXID 2.0 for defining Graphical User Interface (GUI), which is used for controlling the remote robot into the system. Both RCML 2.0 and RXID 2.0 are XML- based languages. By using XML, expandability and flexibility in implementation are introduced to the RCML system. RXID 2.0 has mechanism for a one-way link to an RCML data structure, and this mechanism provides for the complete separation of the control of the robot and the user interface. We also show the reference implementation of the RCML 2.0 system.

**Key words**: R-Cubed, RCML, RCTP, RXID, XML

## 1. Introduction

R-Cubed (Real-time Remote Robotics: R$^3$)[1] is a concept that enables a user to telexist anywhere in the world with a sensation of actually being there. This is accomplished by controlling remote robots over the network. Users of an R-Cubed system feel and act as if they really existed in a remote environment, regardless of the physical limitations of time and space [2].

RCML (R-Cubed Manipulation Language) is considered to be a bottom-up approach of the R-Cubed concept. The design of an RCML system utilizes existing infrastructures and devices such as the Internet and PC and, users of the system will be able to use it easily and intuitively. In a manner similar to the way a VRML browser provides a standard method for accessing the virtual world, we intend to provide a standard method for accessing the remote real environment with an RCML system.

## 2. Related Work

Recently, network robotics is active research area. Many implementation methods have been examined. The simplest implementation is the combination of CGI and HTML [3]. A CGI and HTML based system generates a new web page whenever a user requests a command to a robot. Hence this implementation does not allow a user to control a robot continuously and is not suitable for a system such as RCML that requires continuous control of a remote robot.

An implementation that uses web browser and Java applet [4] is widely used method [5][6][7]. By using Java applet, a user can control a remote robot without installing any special software and, continuous control of a remote robot is achieved at the same time. However it is difficult to build a system such as a high end RCML system that requires real-time processing, because Java has limitation on its performance.

To become more general and sophisticated method for controlling a remote robot, an approach that uses an ORB (Object Request Broker), such as CORBA [8] and DCOM [9], has been also examined. Hirukawa et al. [10] use CORBA to implement their teleoperation system. ORiN [11] that is developed by JARA (Japan Robot Association) [12] uses DCOM. These ORB are mechanism for handling distributed objects and do not define interfaces between each object. Hence it is necessary to define a standard method (API) that can adapt to various robots, but it is very difficult to define such general interface in advance of actual system implementation. Until now several implementations that use an ORB have been proposed, but a standard method for controlling a remote robot is not established yet.

## 3. Previous Implementation

As previously stated, our goal is to provide a standard method to access the real world. Our first step toward this goal was to design the first RCML system called RCML 1.0 in 1997 [13]. RCML 1.0 consists of RCML 1.0 and RCTP/1.0. RCML 1.0 is a description language for controlling a remote robot, and RCTP/1.0 is an HTTP/1.1-based protocol for transferring control data. The design of RCML 1.0 is based on VRML97 [14]. By adding a method to describe the real world to VRML97, we aimed to merge access to the real world and access to the virtual world seamlessly. To maintain upper-compatibility with VRML97, RCML 1.0 uses a PROTO node, which is an extension node of VRML97. By retaining upper-compatibility with VRML97, we can make good use of the existing VRML browser to develop a client program (RCML browser). Furthermore, users who only have a VRML browser will still be able to access an RCML 1.0 file and browse the virtual worlds.

We also developed an experimental implementation to examine and verify the design of the RCML 1.0 system and demonstrated the remote control of the omnidirectional mobile robot [15][16]. Compared to approaches using CGI and HTTP, our system has a short response time that enables us to control the remote robot in a continuous operation and not in a one-by-one command-based operation. In addition, by combining a VRML view, the seamless integration of the two access methods to the virtual and real worlds and intuitive operation were achieved.

However, some design problems within the RCML 1.0 system were revealed at the same time.

The first problem is limitation on implementation. Because the design of RCML 1.0 is an extension of VRML97, it is most efficient to implement the client side program by extending the existing VRML browser. Hence, the development of a client program will always be restricted by limitations of the VRML browser. For instance, there is no choice other than Java to extend the VRML browser, and Java is not a very suitable development environment for a system such as RCML that requires real-time processing.

The second problem is the necessity for the separation of the user interface and the control information definitions. In RCML 1.0, user interface definition such as choice of input GUI and the control information definition such as definition of value are mixed and described in one file. Since control information is specific to each robot, once it is written, it will not be modified so frequently. However, user interface is sometimes modified more frequently than control information. Because various configurations of user interface can be considered, two or more user interfaces may be prepared for one robot. In fact, our RCML 1.0 experimental system has several user interfaces, and each user interface has a different type of input interface, such as a scroll bar and a button. When several user interfaces are prepared for one robot in the RCML 1.0 design, the same control information will exist simultaneously in each file. Such a situation is inefficient and difficult for file management. Hence, it is important to separate user interface definition and control information.

## 4. The Design of the RCML 2.0 System

We tried to make the design of the new RCML 2.0 system as simple as possible. To simplify the system, a target robot is described as a set of variables that are necessary for controlling a robot, and the control of the target robot is considered to be equivalent to accessing variables. The following figure shows a simple example by two degrees of freedom with a pan/tilt camera.

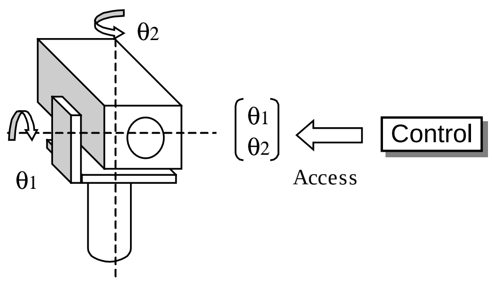

Fig. 1 An example by pan/tilt camera

In the example above, there are two variables that correspond to each pan/tilt axis, and the camera is controlled by accessing these two variables. At this point, we show a very simple example that has only two variables. When we have to handle more variables, it is better to manage variables in a tree structure than in a flat structure. Thus, the RCML 2.0 system manages variables in a tree structure, and this tree structure is called an RCML data structure.

In the RCML 2.0 system, RCML 2.0 is a language for describing an RCML data structure. As described in the previous section, RCML 1.0 inherits not only the advantage of VRML 97 but also the disadvantage of VRML 97. Hence, we decided to base the design of the RCML 2.0 on Extensible Markup Language (XML) [17]. By using XML, the following advantages are introduced into RCML:

- Expandability
- Clear syntax
- Flexibility in implementation

In the RCML 2.0 system, RCTP/2.0 defines a method for accessing the RCML data structure via a network. Upon defining the specification of RCTP/2.0, we considered the following things:

- Ease of implementation
- Expandable design
- Providing the mechanism for real-time control by minimizing the overhead of a data stream

We used the syntax and the sequence of the well-known HTTP/1.1, instead of creating a new protocol entirely, thus making it more understandable. Moreover, because RCTP/2.0 is based on HTTP/1.1, the user can learn it easily, and expandable design is also satisfied because it uses the expansion mechanism of HTTP/1.1. In order to minimize the overhead, we designed a special format for the data stream. RCTP/2.0 also has a mechanism for real-time control by synchronizing time between a server and a client.

We introduced a language for defining GUI, which is used for controlling the remote robot into the system. This language is called RXID (RCML Extensible Interface Definition) 2.0. RXID 2.0 supports well-known common GUI elements such as window, scroll bar, button, and text input and can define property for each element, such as position, size, and caption. Hence, a user can easily design various kinds of user interfaces for controlling the remote robot. RXID 2.0 is also an XML-based language and has mechanism for a one-way link to an RCML data structure, which is illustrated in the following section.

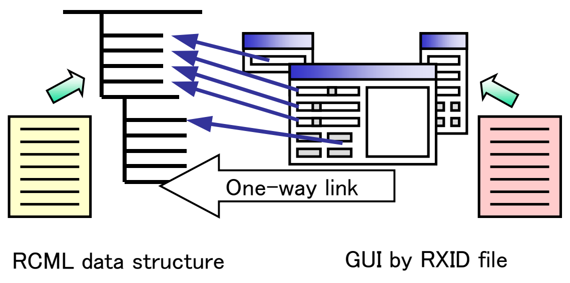

Fig. 2 One-way link in RXID 2.0

This one-way link defines the relationship between a GUI element described in an RXID file and a variable in an RCML data structure defined by an RCML file. By linking these two elements, the input from the GUI side is transferred to an RCML data structure, and the change of variables in the RCML data structure is transferred to the GUI side. Thus, a user can control remote robots by GUI and know the status of the remote robot. Because RXID 2.0's one-way link starts from the RXID file side, it is not necessary to modify the RCML file when describing the RXID file. This provides for the complete separation of the control of the robot and user interface. Hence, multiple user interfaces for one RCML file (Fig. 3) can be defined. Or, one integrated user interface for multiple RCML files (Fig. 4) can be defined.

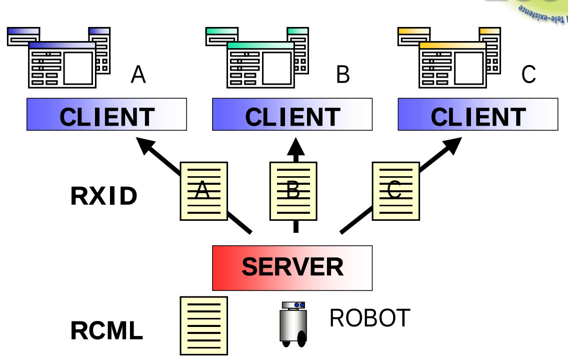

Fig. 3 Multiple interfaces for one robot

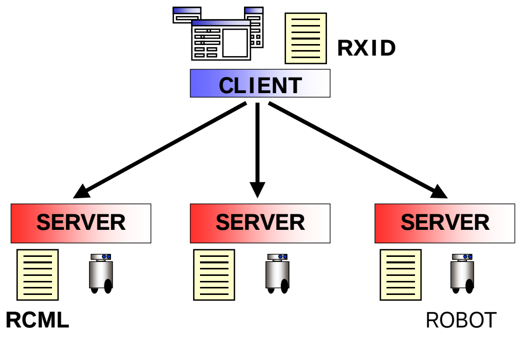

Fig. 4 One user interface for multiple robots

## 5. Outline of the RCML 2.0 system

The next diagram shows an outline of the RCML 2.0 system.

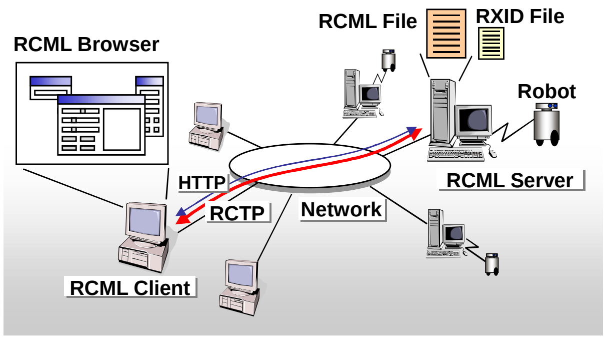

Fig. 5 An outline of the RCML 2.0 system

The RCML 2.0 system consists of an RCML server and an RCML client. A robot is connected to an RCML server. Each RCML server has an RCML 2.0 file, which contains the information of the robot connected to the server and an RXID 2.0 file, which defines the user interface for controlling the robot. An RCML client program specially designed for controlling a remote robot by a human operator is called an RCML browser. An RCML browser downloads the RCML 2.0 file and the RXID 2.0 file by using a standard protocol such as HTTP. An RCML browser then displays a GUI panel based on the RXID 2.0 file and connects to the server using RCTP/2.0 based on the information described in the RCML 2.0 file. Once an RCTP/2.0 connection is established, a user can freely control the remote robot with the RCML browser.

## 6. RCML 2.0

The specification of RCML 2.0 is very simple. RCML 2.0 has only six nodes as follows:

Table 1. Elements of RCML 2.0

| Elements | Explanation |
|---|---|
| <rcml> | The root element of RCML. This element is used to describe the information about an RCML site. |
| <group> | This element declares a group of data. |
| <access> | This element declares a method to access the <data> node. |
| <data> | This element declares the <data> node in an RCML data structure. |
| <link> | This element declares a link for its parent element. |
| <meta> | This element declares a metadata for its parent element. |

In the above list, four elements from <rcml> to <data> elements are used to describe the RCML data structure. The <link> and <meta> elements are elements for describing additional information (metadata) for a data node.

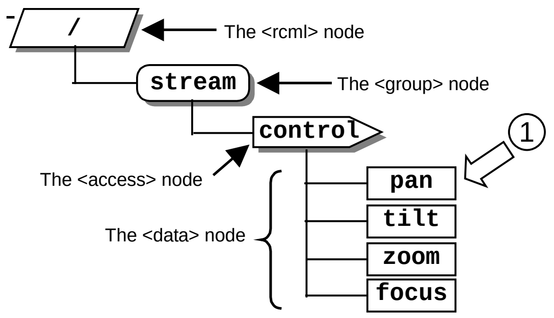

Fig. 6 RCML data structure by sample RCML (Listed in Appendix A)

In an RCML data structure, to indicate a specific node path expression that can be commonly seen at file system is used. For instance, the path to the <data> node located at (1) in Fig. 6 is described as follows:

```
/stream/control/pan
```

In an RCML data structure, the name of a node must satisfy the following rules:

- The same rule that is defined as "Name" in an XML syntax applies to a node name.
- Nodes in the same level must have different names.
- The order of nodes does not have a specific meaning, unlike an XML document.

## 7. RCTP/2.0

RCML 2.0 only defines interface for controlling remote robots. Hence, to make an actual system, some sort of communication method is required. RCTP/2.0 is used as a communication method in the system. RCTP/2.0 defines the protocol for reading and writing data that are described by RCML 2.0. RCTP/2.0 has the following functions:

- Access for data - read and write
- Controls of access privilege

### 7.1 Access methods in RCTP/2.0

RCTP/2.0 has some data access methods. In RCML 2.0, these access methods can be specified for each <data> node. Each access method is briefly described in the next section.

#### 7.1.1 Normal access

When no access type is specified in an RCML file, the normal access method is used. The normal access method uses connection-oriented connection. An access occurs to each <data> node. This is the simplest access method.

#### 7. 1.2 Event-type access

When an event-type access method is specified in an RCML file, this access method is used. The same as a normal access method, an event-type access method uses connection-oriented connection. The difference from the normal method is simultaneous access for set of data and the occurrence of a “data change event” from a server. It places the importance of the assurance of changing variables between a server and a client. So, an event-type access is suitable to set the parameter for the robots or to send a sequence of commands.

#### 7.1.3 Stream-type access

When a stream-type access methods are specified in an RCML file, this access method is used. Different from other methods, a stream-type access uses a connection-less data stream. By sending data as a stream, a stream-type access can change data continuously. To send the newest data without delay, a lost packet is not sent again in a stream-type access. A stream-type access attaches more importance to real-time access of data than event-type access. So, when a bandwidth of network is very broad and time delay is short, it is very useful.

### 7.2 Connections of RCTP/2.0

RCTP/2.0 defines two types of connections: control and data stream.

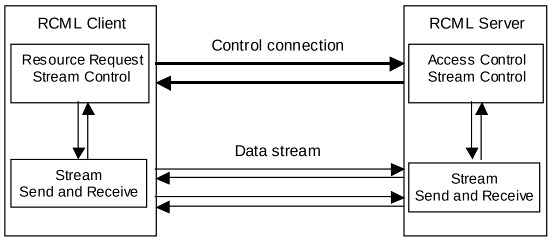

Fig. 7 Connections of RCTP/2.0

Control connection mainly obtains access control and controls data stream. Control connection uses a connection-oriented method. A client establishes a control connection to a server. A session is a period starting when the client establishes a control connection and ending with disconnection. Normal access and event-type access use control connection.

On the other hand, data stream continuously transfers the control data that is needed to control remote robots. Data stream is used to transfer control data for the robot that requires real-time control. Thus, it uses a connection-less method that does not handle the re-transmission of packets. A stream-type access uses this data stream.

### 7.2.1 Control connection

As in HTTP/1.1, control connection is a protocol based on a request and response pair. The structure of the message is also the same, where a start-line includes a method and an RCTP version in a request, and it includes a status code and a Reason-Phrase in a response. A message header follows the status line and the message body comes in last. RCTP/2.0 defines the following 10 methods:

Table 2. Methods of RCTP/2.0

| **Method name** | **Explanation** | C→S | C←S |
|---|---|---|---|
| CONNECT | Starts an RCTP session | ○ | × |
| ACQUIRE | Acquires an access permission | ○ | × |
| RELEASE | Releases an access permission that was obtained | ○ | × |
| READ | Obtains the value of <data> node | ○ | × |
| WRITE | Sets the value of <data> node | ○ | ○ |
| SETUP | Sets the parameters for access method | ○ | × |
| GO | Instructs the beginning of access | ○ | × |
| PAUSE | Instructs the pause of access | ○ | × |
| STOP | Instructs the end of access | ○ | ○ |
| BYE | Ends the session | ○ | ○ |

RCTP/2.0 allows a server to issue a request on a client in the method WRITE, STOP, and BYE, which is quite different from HTTP/1.1. In the above table, C→S represents the request from a client to a server while C←S represents the request from a server to a client.

### 7.2.2 Data stream

Data stream uses a connection-less method that uses packets to communicate. HTTP/1.1 does not have data stream connection. Data stream is used for stream-type access. To ensure real-time communication, it does not re-transmit data when packets are lost. A data stream packet can include several "payloads," which are the minimum units of data transmission. By making several payloads that are generated at the same time into one packet, it is possible to decrease the number of packets in a data stream. A payload also has a field that shows the type of information it contains. Thus, it is possible to overlap several types of information in one data stream. When transmitting real data in a data stream, a binary format is used as in READ and WRITE methods in control connection.

In addition to data-stream payload for real data transmission, RCTP/2.0 also defines flow-control payload. The protocol for flow control is very simple: a request for operation and the acknowledgement of the request and the negative acknowledgement. Operation provides heartbeat operation for synchronizing local time and reading and setting operation of flow-control parameters. As flow-control parameters, RCTP/2.0 defines the transmission interval of payloads and the timeout value of receiving payloads.

## 7.3 Two aspects of RCTP/2.0

The control of a data stream and the management of the right to control the robot must take place at the same time. Because, when controlling remote robots, to give permission for sending and receiving a data stream for a client is equivalent to giving the right to control the robot to the client. Thus, it is inefficient and complicated to implement when they are managed by different protocols. Therefore, RCTP/2.0 has two aspects: management of server resources and transmission and control of a data stream.

## 8. RXID 2.0

RXID 2.0 defines the following elements:

Table 3. Elements of RXID 2.0

| **Name** | **Explanation** |
|---|---|
| <rxid> | The root element of RXID. |
| <window> | This element creates a window. A window can be used as a placeholder for all other RXID widgets. |
| <session > | This element declares an RCTP session. |
| <access> | This element declares an access method to data nodes in an RCML data structure. |
| widget elements | This kind of element creates user interface elements (RXID widgets). |

The <window> element is always the child element of the <rxid> root element. One <window> element corresponds to one window displayed by the RCML browser. Attributes of the <window> element represent the property of a window such as position, size, title, and background image. Each <window> element must have at least one <session> element to specify the URL of a target RCML file. The <access> element can be used to declare the access method to the specific node in an RCML file. The <window> element also has widget elements as child elements. Widget elements are used to place various user interface elements (RXID widgets) inside the window. The current version of an RCML browser supports the following widget elements:

Table 4. Currently supported widget elements

| Name | Explanation | R | W |
|---|---|---|---|
| <box> | This element creates a box. | × | × |
| <label> | This element creates a label. A label is used to display static text. | × | × |
| <text> | This element creates a text. A text is used to show values that can be updated in real time. | ○ | × |
| <button> | This element creates a button. | × | ○ |
| <checkbox> | This element creates a checkbox. | ○ | ○ |
| <radioGroup> | This element creates a group of radio buttons. | ○ | ○ |
| <scroll> | This element creates a scroll bar. | ○ | ○ |
| <slider> | This element creates a slider. | ○ | ○ |
| <edit> | This element creates an edit box. | ○ | ○ |
| <popUpMenu> | This element creates a pop-up menu. | ○ | ○ |
| <netmeeting> | This element creates a live video viewer component (NetMeeting). | × | × |
| <html> | This element creates an html viewer component. | × | × |
| <actionButton> | This element creates an action button. | × | × |

In the list above, 'R' indicates that the widget can read data from an RCML data structure. For instance, the <slider> element reads a current position of the slider knob from an RCML data structure and updates the position of the slider knob. On the other hand, 'W' indicates that the widget can write data to an RCML data structure. The element, which supports 'write' action, such as a button, checkbox, and scroll, can write the change of value inputted from a user to an RCML data structure. There are also elements that support neither read nor write action. Boxes and labels, for example, represent static widgets and are not related to an RCML data structure.

The widget that can do read or write action has a "dataPath" attribute to declare a one-way link to an RCML data structure. Here is brief example of a scroll bar:

```
<scroll dataPath="/stream/control/pan" … />
```

The scroll bar above is linked to the node specified by "/stream/control/pan" in the RCML data structure (Fig. 6 (1)).

## 9. Reference System

We also implemented an actual system based on the design of RCML 2.0. The main purpose of this system is to show the reference implementation of the RCML 2.0 system. Hence, we tried to fullfill the specifications of an RCML 2.0 system as much as possible, and we also tried to keep the system simple and easy to understand and to extend.

The target platform of our system is Windows (Windows 98, NT 4.0, 2000) and Unix (FreeBSD, LINUX, etc.). Currently, the RCML client supports Windows platforms only. The main development language is C++, and "XML for C++ (Version 2.3.1)" [18] is used as an XML processor.

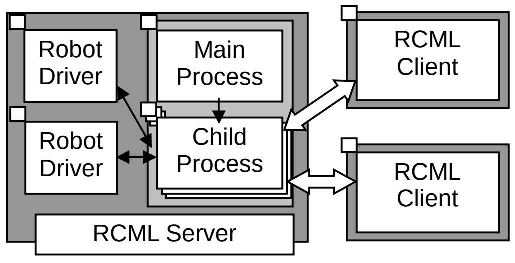

Fig. 8 Processes in RCML 2.0 system

The RCML server consists of the main process, the child processes, which handle each session to an RCML client, and the robot driver processes, which handle each robot. The RCML client is one independent application and connects to the desired RCML server by typing URL as would be done in an ordinal web browser.

The following image is the screen-shot of the RCML client:

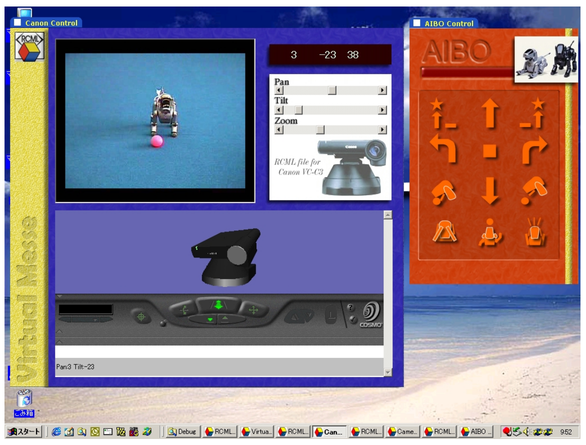

Fig. 9 The screen-shot of the RCML client

## 10. Conclusion

In this paper, we showed a new design for an RCML system (RCML 2.0) [19]. By using XML in the system design, the new design provides expandability and flexibility to the RCML system. In RCML 2.0, a language RXID 2.0, which is used for defining user interface, is introduced. By introducing RXID 2.0 into the system, complete separation of the control of the robot and user interface is achieved. We also developed the reference implementation of RCML 2.0 system. Our reference implementation fulfills almost all the specifications defined by the specifications of the RCML 2.0 system.

## References

1. MITI of Japan, R-Cubed WG ed.: "R-Cubed", *Nikkan Kogyo Shinbun*, (1996).
2. S. Tachi: "Real-time Remote Robotics – Toward Networked Telexistence", *IEEE Computer Graphics and Applications*, pp. 6-9, (1998).
3. R. Simmons: "Xavier: An Autonomous Mobile Robot on The Web", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 43-47, (1998).
4. http://java.sun.com
5. M. R. Stein: "Painting on the World Wide Web: The PumaPaint Project", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 37-42, (1998).
6. Roland Siegwart, et al.: "Guiding Mobile Robots through the Web", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 1-6, (1998).
7. P. Saucy, F. Mondada: "KhepOnTheWeb: One Year of Access to a Mobile Robot on the Internet", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 23-29, (1998).
8. http://www.omg.org
9. http://www.microsoft.com
10. H. Hirukawa and I. Hara: "The Web Top Robotics", *Preprints IROS'98 Workshop 'Robots on the Web'*, pp. 49-54, (1998).
11. Mizukawa,M., Matsuka,H., Koyama,T., Matsumoto,A.: "De-facto standard API for Open and Networked Industrial Robots", *Proc. 30th Int. Symp. on Robotics*, pp.455-462, Oct. 1999
12. http://www.jade.dti.ne.jp/~jara/
13. Y. Yanagida, N. Kawakami, S. Tachi: "Development of R-Cubed Manipulation Language - Access Real Worlds Over the Network", *Proc. of the 7th International Conference on Artificial Reality and Tele-existence*, pp159-167, 1997
14. http://www.web3d.org
15. W. C. Teng, A. Nukuzuma, N. Kawakami, Y. Yanagida, S. Tachi: "Development of R-Cubed Manipulation Language -The specification of RCML and RCTP-", *Proc. of the 8th International Conference on Artificial Reality and Tele-existence*, pp152-162, 1998
16. W. C. Teng, D. Sekiguchi, A. Nukuzuma, N. Kawakami, Y. Yanagida, S. Tachi: "Development of R-Cubed Manipulation Language –Implementation and Evaluation of RCML System-", *Proc. of the 9th International Conference on Artificial Reality and Tele-existence*, pp79-83, 1999
17. http://www.w3c.org
18. http://alphaworks.ibm.com/
19. http://www.rcml.org

## Appendix A: RCML 2.0 Sample

```
<?xml version="1.0" encoding="Shift_JIS"?>
<!DOCTYPE rcml SYSTEM "rcml.dtd">

<!-- RCML Version 2.0 sample -->

<rcml
    version="2.0"
    site="rctp://rrr.rcml.org"
    timeSource="GPS"
    timePrecision="1E-4"
    title="RCML sample"
    author="D.Sekiguchi"
    info="RCML test site."
    contact="mailto:dairoku@rcml.org"
>

  <group name="stream" permission="rw">
    <access name="control"
         type="stream"readInterval="16e-3"
         writeInterval="16e-3"
         readTimeout="10"
         writeTimeout="10">
       <data name="pan" type="int"/>
       <data name="tilt" type="int"/>
       <data name="zoom" type="int"/>
       <data name="focus" type="int"/>
    </access>
  </group>

  <link kind="UserInterface"
    href="sample.rxid">

</rcml>
```