PROPOSED DRAFT Proceedings of ASME International:
20th Computers and Information in Engineering (CIE) Conference
September 10-13, 2000, Baltimore, Maryland

Paper Number Here

Implementation of VRML/Java Web-based Animation and Communications for the Next Generation Inspection System (NGIS) Real-time Controller

William Shackleford
National Institute of Standards and Technology
100 Bureau Drive, Stop 823
Gaithersburg, MD 20899-8230
(301) 975-4286
william.shackleford@nist.gov

Keith Stouffer
National Institute of Standards and Technology
100 Bureau Drive, Stop 823
Gaithersburg, MD 20899-8230
(301) 975-3877
keith.stouffer@nist.gov

ABSTRACT

The Next Generation Inspection System (NGIS) project is a testbed that consists of a Cordax Coordinate Measuring Machine (CMM), advanced sensors, and the National Institute of Standards and Technology (NIST) Real-Time Control System (RCS) open architecture controller. The RCS controller permits real-time processing of sensor data for feedback control of the inspection probe. The open architecture controller permits external access to internal data, such as the current position of the probe. A remote access web site was developed to access this data to drive a Virtual Reality Modeling Language (VRML) model of the Cordax CMM. The remote access web site contains a client-controlled pan/tilt/zoom camera which sends video to the client as well as the VRML 3D model of the CMM that is controlled by the NGIS controller located at NIST. This remote access web site allows a client to monitor a remote inspection with a PC and an internet connection.

INTRODUCTION

The Next Generation Inspection System (NGIS) project is a testbed that consists of a Cordax Coordinate Measuring Machine (CMM), advanced sensors, and the National Institute of Standards and Technology (NIST) Real-Time Control System (RCS) open architecture controller. The goal of the NGIS project (involving a consortium of compainies organized by the National Center for Manufacturing Sciences) is to increase the speed and flexibilty of data acquisition using CMMs while still maintaining today's accuracy.

NOMENCLATURE

CMM - Coordinate Measuring Machine
EAI – External Authoring Interface
NGIS - Next Generation Inspection System
NML - Neutral Messaging Language
RCS – Real-Time Control System
VRML - Virtual Reality Modeling Language

NGIS COMMUNICATIONS

The communications requirements for NGIS are somewhat complex. The controller consists of a hierarchy of modules based on the NIST RCS Reference Model Architecture[1][10]. Each module in the hierarchy reads commands and posts status each cycle and may send commands and read the status of a number of subordinate modules. Within the NGIS controller these cycle times vary from 2 milliseconds for the module that reads the touch probe data to 100 milliseconds for the modules that process images. The size of the messages involved vary from less than 100 bytes for the servo control module to 300 kb for images. The modules are also distributed among 3 different operating systems: SunOs, VxWorks, and Windows NT [2],[3]. Although it would be possible to use many different communications packages for the different channels, using a single communications package for the majority of the channels makes the controller easier to understand and promotes better software maintainability.

DISCLAIMER: Any mention of commercial products within this paper is for information only; it does not imply recommendation or endorsement by NIST.

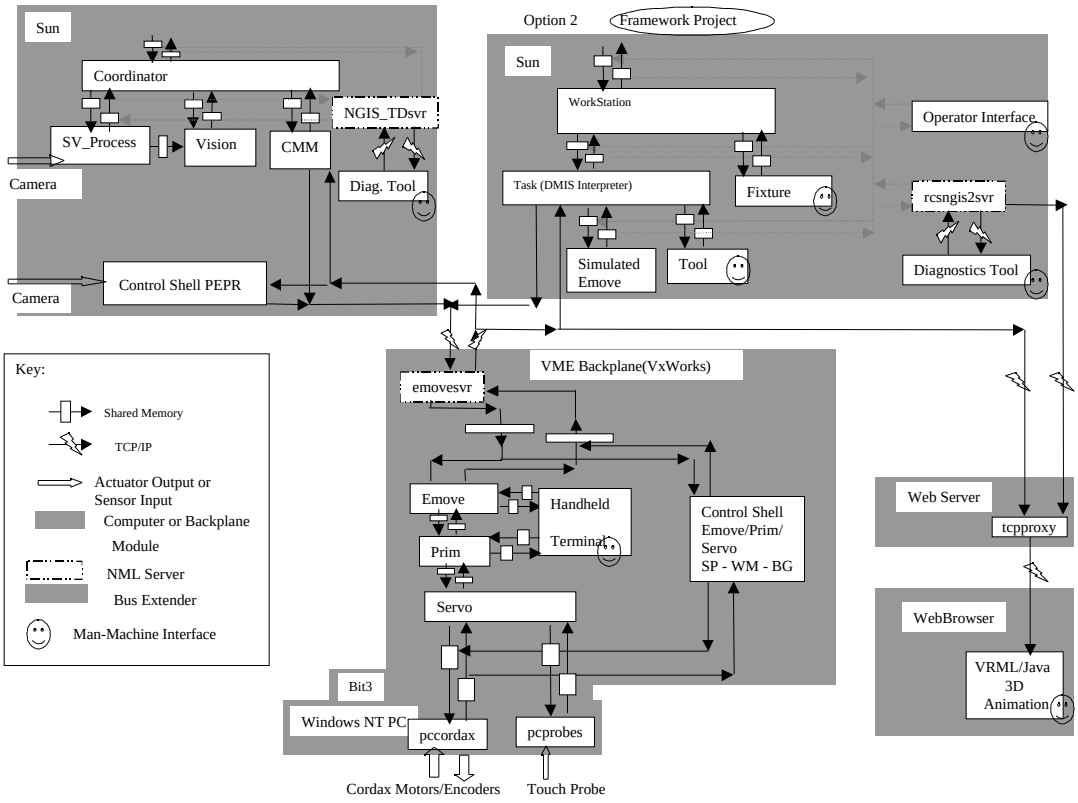

Figure 1: NGIS Communications Paths

Figure 1 shows a diagram of the communications paths in the NGIS controller. There are several different subsystems that may or may not be used in a particular inspection, demonstration or experiment depending on the version of the software being used and the focus of the particular inspection. A complete explanation for the purpose of these modules is beyond the scope of this paper. However there are several important observations to be made:

1. All of the shared memory and TCP/IP connections shown here use an Application Programming Interface (API) called the Neutral Messaging Language (NML).
2. NML provides a convenient means of defining interfaces between modules, which is especially important given the large number of compatible alternative subsystems created by different teams.
3. The majority of communications occurs via shared memory which allows the processes to communicate with minimal delay with real-time determinism.
4. Special processes called NML servers provide access to the shared memory buffers to remote processes via TCP and isolate the real-time modules from the delays associated with TCP.
5. Because NML is open-source it could be modified to incorporate additional physical communication protocols. For example, NML was modified to support the Bit3 bus extender between the PC and the VME backplane .
6. The 3D animation in the web browser can be configured either to use simulated position values from the Sun or to get the real positions directly from the VxWorks board.
7. Because of security restrictions in most web browsers it is necessary to run the tcpproxy process on the web server to forward position messages from the controller to the animation.

There were several levels in the controller where position data suitable for the 3D animation was available. However, the emove status level which is approximately in the middle of the controller is a good choice for several reasons. The same data is available in both simulation and when moving the real machine. When moving the real machine, the data is sensed from the encoders or scales on the machine rather than being a calculated value of where the machine is supposed to be. The controller updates this position every 30 milliseconds, which is more than enough for smooth animation. The data is also already being converted to the eXternal Data Representation (XDR) so the format is not processor specific and being sent via TCP, which makes it much easier to access in Java.

MODELING THE CORDAX CMM IN VRML

The Virtual Reality Modeling Language (VRML) is a recently ratified ISO standard (ISO 14772) [4] file format for the description of geometry and behavior of 3D computer graphics. VRML is the most widely used open file format for integrating 3D computer graphics with Web based content. We are using VRML as the primary interface technology for the representation of interactive virtual worlds. VRML provides a robust foundation upon which 3D application can be built that are portable and distributable via the World Wide Web. It defines most of the commonly used semantics found in today's 3D applications such as hierarchical transformations, light sources, viewpoints, geometry, animation, fog, material properties, and texture mapping. One of the primary goals in designing VRML was to ensure that it at least succeeded as an effective 3D file interchange format.

The Cordax CMM was originally modelled using a robotic simulation package from Deneb Robotics called TeleGRIP. This simulation package allows for design, evaluation, and off-line programming of robotic workcells, incorporating real world robotic and peripheral equipment, motion attributes, kinematics, dynamics and I/O logic.

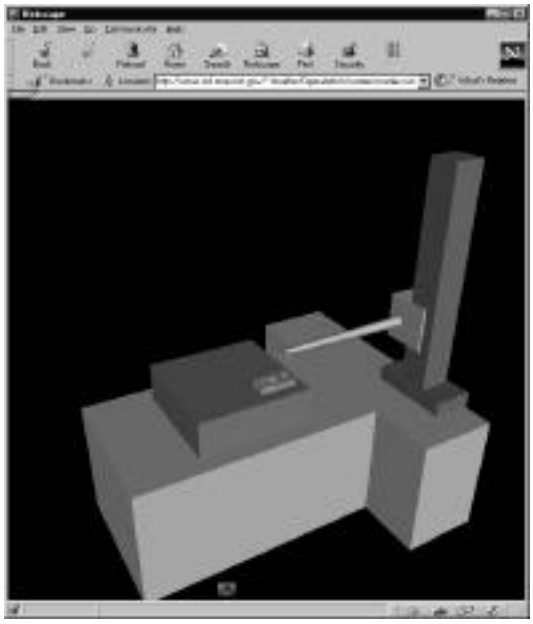

Figure 2 VRML Model of the Cordax CMM

In order to view the 3D model of the CMM within a web browser, the model must first be translated to VRML. A translator program was written by Qiming Wang and Sandy Ressler of the Information Technlogy Laboratory (ITL) at NIST which can be used to translate devices and workcells generated by the TRGIP to VRML[9]. The translation includes the geometry information as well as the kinematic information of the device. Figure 2 shows the translated VRML model viewed within a web browser using a VRML plugin such as Cosmoplayer or Blaxxun Contact.

CREATING JAVA INTERFACE FILES FROM EXISTING C++ CODE

To use the controller interfaces from within a web browser, they must be converted to Java. Creating Java interface files from existing C++ code could be fairly tedious without a tool to automate the process. Although the languages are very similar, there are some important differences that make conversion more difficult.

1. Java does not allow macros (#define's) so any macros used in the C++ classes to be converted have to be expanded out in the Java interface.
2. Java does not allow typedefs. There are two ways to handle the typedefs. For small one-line typedefs like "typedef unsigned long uint32;" it is easiest just to replace the name with the full type information in the typedef. For others such as "typedef struct { double x,y,z; } pose;" it is better to create a Java class with the same name as the typedef.
3. Java does not allow enums. However, it is usually very helpful to convert the constants defined in the enum to static final int's in the Java interface.
4. Java does not allow unsigned integer types (except for char). Usually one can just remove the "unsigned" tag and use a signed integer of the same size. However if the very large values of the unsigned integer are really being used it may be necessary to convert to a larger integer type.
5. Java uses "byte" instead of "char". (A Java char corresponds to an unsigned short since Java uses Unicode rather than ASCII characters.)
6. Java allows only one public class in each .java file. Since all the interface classes should be public instead of one or two C++ header files many Java files each with a different class are needed.
7. Java does not allow objects of classes to be embedded within other classes as C++ does. Instead one can insert a reference that needs to be initialized with the new operator.

Fortunately, the Neutral Messaging Language (NML) that was used for communications within the controller also provides a tool to automatically convert classes in C++ header files to Java files. The tool can be run in two modes. An interactive mode that can be accessed as a Java applet (http://isd.cme.nist.gov/proj/rcs_lib/CodeGen.html) that is easier to use for one-time only conversions. A non-interactive standalone version that can be used within a makefile for keeping Java interface files synchronized with the C++ header

files they depend on. The C++ classes used for communicating with the emove level is listed in Figure 2.

```
// Declare NML/CMS status message
class.
  class emoveStatus: public
RCS_STAT_MSG
{
  public:
  emoveStatus();
  void update(CMS *);
#ifdef JAVA_DIAG_APPLET
  EMOVETYPE        statusType ;
       /* status type  */
#else
  int       statusType ;
       /* status type  */
#endif

  double  emoveToolPoseTranX  ;
  double  emoveToolPoseTranY  ;
  double  emoveToolPoseTranZ  ;
  double  emoveToolPoseRotS  ;
  double  emoveToolPoseRotX  ;
  double  emoveToolPoseRotY  ;
  double  emoveToolPoseRotZ  ;
  int   emoveCS ;
  int        statusReportNumber ;
  int          emoveTermCondition ;
};
```

Figure 3

Figure 3 shows the automatically generated Java interfacclass:

```
public class emoveStatus extends
RCS_STAT_MSG
{
public int statusType = 0;
public double emoveToolPoseTranX = 0;
public double emoveToolPoseTranY = 0;
public double emoveToolPoseTranZ = 0;
public double emoveToolPoseRotS = 0;
public double emoveToolPoseRotX = 0;
public double emoveToolPoseRotY = 0;
public double emoveToolPoseRotZ = 0;
public int emoveCS = 0;
public int statusReportNumber = 0;
public int emoveTermCondition = 0;

// Constructor
public emoveStatus()
{
super(8050);
}

public void update(NMLFormatConverter
nml_fc)
{
super.update(nml_fc);
statusType =
nml_fc.update(statusType);
emoveToolPoseTranX =
nml_fc.update(emoveToolPoseTranX);
emoveToolPoseTranY =
nml_fc.update(emoveToolPoseTranY);
emoveToolPoseTranZ =
nml_fc.update(emoveToolPoseTranZ);
emoveToolPoseRotS =
nml_fc.update(emoveToolPoseRotS);
emoveToolPoseRotX =
nml_fc.update(emoveToolPoseRotX);
emoveToolPoseRotY =
nml_fc.update(emoveToolPoseRotY);
emoveToolPoseRotZ =
nml_fc.update(emoveToolPoseRotZ);
emoveCS = nml_fc.update(emoveCS);
statusReportNumber =
nml_fc.update(statusReportNumber);
emoveTermCondition =
nml_fc.update(emoveTermCondition);
}
}
```

Figure 4

The update function is needed to convert from XDR to the data format used by the particular Java Virtual Machine (JVM) the user is running. The code however is designed to work with any JVM.

This class is then used in Figure 5 with the regular NMLConnection class to read the emoveStatus data:

```
emove = new NMLConnection(new
emoven(),"emoveSts","CordaxApplet",
"http://isd.cme.nist.gov/~stouffer/Ope
ratorInt/cordax/ngis.nml" );
. . .

msg = (emoveStatus) emove.peek();
```

Figure 5

(See [5] for the complete API.)

Some CORBA development systems provide automatic tools that generate Java and C++ code from IDL. This would not have been very helpful here, since this requires modifying or adding a wrapper to the C++ controller and hand-coding the IDL.

WEB-BASED JAVA COMMUNICATION WITH THE CONTROLLER VIA NML

In general the communications between an NML client and server is fairly simple. First the server starts, reads a configuration file that tells it whether to use TCP or UDP and which port, then binds the port and listens for requests. Then a client starts, reads the same configuration file that tells the port whether to use TCP or UDP and which host the server is running on, connects to the appropriate port and sends requests to the server. The client can send requests to write a message, read a message only one-time, or read a message and automatically be sent updates whenever the data changes or at a particular rate.

Figure 6 shows the relevant portion of the configuration file used by the server and animation client. The file format is explained in more detail in [6]:

```
# Buffers
# Name      Type  Host        size
       neut? RPC#
       buffer#        MP . . .
B emoveSts     SHMEM abyss    2048
       0       0x203fd6bf     17
       16 38858 TCP=4575 disp
# Processes
# Name                 Buffer
Type    Host   Ops     server?
       timeout        master?
cnum
# wspl(0)
P CordaxApplet emoveSts   REMOTE
abyss  R       1              1.0
        1        7
```

Figure 6

TCP PROXY

One feature that complicates things somewhat is that the web browsers usually prevent Java applets running inside them from connecting to any hosts other than the web server they were loaded from. Also more and more institutions are adding firewalls that prevent any host outside the wall from directly contacting hosts inside the wall. Tcpproxy is a fairly generic program that knows nothing about NML or the NGIS controller but simply forwards TCP data from one host to another in order to provide a very limited bypass of these security restrictions.

The Tcpproxy is started on the web server with command line arguments indicating the host with the NML server and the TCP port number. It binds that port on the web server and listens for connections. Each time a client connects it connects to the NML server on the controller. It forwards any requests made by that client to the server and any data sent by the server back to the client in such a way that neither the client nor the server is aware that tcpproxy is involved at all.

LOW BANDWIDTH CONNECTION

When a client visits the web page, a VRML model of the Cordax CMM is downloaded to their machine. Once the VRML model of the Cordax has been downloaded to the client's machine, only the joint positions need to be sent across the socket from the controller (server) to the web browser (client) to move the joints of the VRML CMM model to the current controller position. All of the graphics are handled on the client machine. This low bandwidth is usually handled easily, even on a slow network connection.

CONTROLLING THE CORDAX VRML MODEL WITH THE CONTROLLER

For communication between a VRML world and its external environment an interface between the two is needed. A proposed standard is called the External Authoring Interface (EAI) [7] ,which defines the set of functions on the VRML browser that the external environment can perform to affect the VRML world. In essence, the EAI provides a method for developing custom applications that interact with, and dynamically update a 3D scene. These outside applications "talk" to the VRML scene. The interface described here is designed to allow an external program (referred to here as an applet) to access nodes in a VRML scene using the existing VRML event model. The EAI Spec has been submitted to the VRML Review Board (VRB) [8] for ISO formalisation. Cosmoplayer 2.1 and Blaxxun Contact are two VRML browsers that incorporate the EAI.

The VRML model of the inspection cell contains a model of the Cordax CMM that is controlled by the real world NGIS controller. This is accomplished by the NML connection between the client's web browser and the real world controller located at NIST. The current position of the measurement probe, which is stored in the world model buffer in the controller, is collected by a Java applet on the web page. The applet then updates the VRML model of the Cordax via the EAI of the VRML browser. Figure 7 shows the remote access web site was developed that contains a client-controlled pan/tilt/zoom camera which sends video to the client as well as the continually updated VRML 3D model of the CMM. This remote access web site allows a client to monitor a remote inspection with a PC and an internet connection.

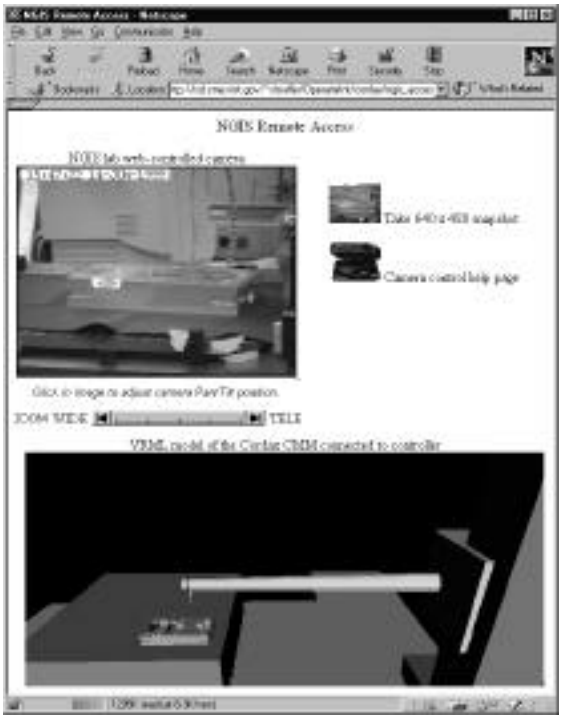

Figure 7: NGIS Remote Access Page

JAVA APPLET -> VRML PLUGIN COMMUNICATION CODE

Figure 8 contains a piece of the VrmlTrans.java code which defines a externally controllable VRML translational joint

```
public class VrmlTrans
{
    Node myNode;
    EventInSFVec3f translation;
    String nodeName;
    float current[];

    public VrmlTrans(String name,
Browser browser)
    {
        myNode =
browser.getNode(name);
        translation = (EventInSFVec3f)

myNode.getEventIn("set_translation");
    }

    public void updateValue(int axis,
float newValue)
    {
          float current[] =
((EventOutSFVec3f)
(myNode.getEventOut("translation_chang
ed"))).getValue();

         if (axis == 1)
            current[0] = newValue;

         if (axis == 2)
            current[1] = newValue;

         if (axis == 3)
            current[2] = newValue;

          // Update position to VRML
through the EAI

translation.setValue(current);
    }
}
```

Figure 8

Figure 9 contains a small piece of the CordaxApplet.java file which handles the communication between the real NGIS controller and the VRML model of the Cordax

```
public class CordaxApplet extends
Applet implements Runnable
{
    emoveStatus emoveStatusData;
    NMLConnection emove;
    boolean stayAlive;
    Thread myThread;
    String err_msg;

 // Added for EAI stuff
    Browser browser;
    Node[] joint;
    VrmlTrans[] myJoints;

     public void init()
    {  myJoints = new VrmlTrans[3];  }

     public void start()
    {
browser=(Browser)vrml.external.Browser
.getBrowser(this);
        myJoints[0] = new
VrmlTrans("cordax_j1", browser);
        myJoints[1] = new
VrmlTrans("cordax_j2", browser);
        myJoints[2] = new
VrmlTrans("cordax_j3", browser);
        …
     }
public void run()
    {
        try { emove = new
NMLConnection(new
                emoven(),"emoveSts",
        "CordaxApplet",
                "http://isd.cme.nist.go
        v/~stouffer/
                OperatorInt/cordax/ngis
                .nml" );
               } …

        while(stayAlive)
        try{  msg = (emoveStatus)
emove.peek(); }

                emoveStatusData = msg;

              // Update the VRML model
with the new values
        myJoints[0].updateValue(1,
          -
        (float)emoveStatusData.emoveToo
        lPoseTranY);

        myJoints[1].updateValue(2,
        (float)emoveStatusData.emoveToo
lPoseTranZ);

        myJoints[2].updateValue(3,
        (float)emoveStatusData.emoveToo
lPoseTranX]);
      }
```

Figure 9

CONCLUSIONS

The NGIS project is a testbed that consists of a Cordax CMM, advanced sensors, and the NIST RCS open architecture controller. The RCS controller permits real-time processing of sensor data for feedback control of the inspection probe. Because of the open architecture, data such as the current probe position, can be obtained from the controller. A remote access web site was developed to access this data to drive a VRML model of the Cordax CMM. The remote access web site contains a client-controlled pan/tilt/zoom camera which sends video to the client as well as the VRML 3D model of the CMM that is controlled by the NGIS controller located at the NIST. This remote access web site allows a client to monitor a remote inspection with a PC and an internet connection. The remote access web site is currently located at http://www.isd.cme.nist.gov/~stouffer/public_html/OperatorInt/cordax/ngis_access.html.

Although this seems to be a fairly easy means of allowing users some access to the machine remotely. We still do not know how useful this type of remote access will be. It might be used by a remote expert on a particular part or type of metrology to verify that an inspection is being done correctly. It could also be used by the developer of the CMM control software as a very cheap simulation tool. Finally this might be adapted for monitoring a large variety of machines that have significant 3D movement and can not be easily monitored directly.

REFERENCES

[1] Albus, J.S., Meystel, A.M.,"A Reference Model Architecture for Design and Implementation of Intelligent Control in Large and Complex Systems," International Journal of Intelligent Control and Systems, Vol. 1, No. 1 pp. 15-30

[2] Nashman, M., Yoshimi, B., Hong, T.H., Rippey, W.G., Herman, M., "A Unique Sensor Fusion System for Coordinate Measuring Machine Tasks", SPIE Internat'l Symp. on Intelligent Systems & Advc. Manufact. Session:Sensor Fusion & Decentralized Control in Autonomous Robotic Systems, Pitts.., PA, 10/97

[3] Scott, H.A., "The Inspection Workstation-based Testbed Application for the Intelligent Systems Architecture for Manufacturing**",** Proceedings of the International Conference on Intelligent Systems: A Semiotic Perspective, Gaithersburg, MD, October 20-23, 1996

[4] VRML VRML2.0 Specification ISO/IEC CD 14772, 1996

[5] Shackleford, W.P., "The NML Java Programmer's Guide", http://www.isd.cme.nist.gov/proj/rcs_lib/NMLjava.html.

[6] Shackleford, W.P., "Writing NML Configuration Files", http://www.isd.cme.nist.gov/proj/rcs_lib/NMLcfg.html.

[7] External Authoring Interface Working Group: http://www.vrml.org/WorkingGroups/vrml-eai/

[8] VRML Review Board

http://www.vrml.org/vrb/

[9] Quiming Wang, Sandy Ressler, “Translating: IGRIP Workcells into VRML2”,NISTIR 6076, September 1997

[10] Moore M., Gazi V., Passino K., Shackleford W., and Proctor F., "Complex Control System Design and Implementation Using the NIST-RCS Software Library," IEEE Control Systems Magazine, Vol. 19, No. 6, pp. 12-28, Dec. 1999.