

AN OBJECT-ORIENTED PC-BASED SYSTEM FOR TSPI COLLECTION AND DISTRIBUTION

Mike Paulick
TYBRIN Corporation
Fort Walton Beach, Florida
under contract to
Freeman Computer Sciences Center
Eglin AFB, Florida

Tim Thomas
TYBRIN Corporation
Fort Walton Beach, Florida
under contract to
Freeman Computer Sciences Center
Eglin AFB, Florida

ABSTRACT

The Range Instrumentation and Control System (RICS) is a PC-based client/server application designed to collect time-space position information (TSPI) from remote radar test sites and distribute it in real-time across a wide area network (WAN). The system architecture is composed of two main parts—the Data Interface Adapter (or DIA, which runs under VxWorks and is implemented using C/C++) and the RICS console PC (which runs under Windows 2000 and is implemented in Java). CORBA is used to provide communication between the RICS console and DIA. This paper describes the design of the system, focusing primarily on the DIA software.

KEY WORDS

Real-Time, Java, C++, CORBA, VxWorks, UDP Multicast

INTRODUCTION

In 1995, the 46th Test Wing at Eglin AFB in Northwest Florida initiated a reconstitution program for the radar test sites on Santa Rosa Island to repair the extensive damage caused by Hurricane Opal. Due to the extremely high cost of rebuilding and repairing facilities, the Test Wing personnel decided to consolidate and modernize the most actively used sites. Their goals were to provide more generalized support and to expand capabilities from Electronic Combat to C4I support and munitions testing.

The Range Instrumentation and Control System (RICS) is the twenty-first century replacement of the aging Universal Data System (UDS) that has been in operation at Eglin for over 25 years. RICS will serve as the nucleus for collecting, formatting, recording, verifying, and routing Time-Space Position Information (TSPI) in real-time. It will capture radar TSPI data at rates from 10 to 60 hertz and provide it to its clients via a fiber optic network. When the reconstitution program is completed at Eglin, there will be three focus test sites equipped with RICS.

The RIC computer is the command and control component of the system providing the operators a graphical interface to the radar(s). It displays information from one or more DIAs and allows remote configuration. It is located in the control building at the focus site. The RIC computer is a standard high-performance desktop PC running MS-Windows 2000. The RIC software is implemented in Java.

The relationship of the DIA to the RIC computer is a classic client/server distributed computer configuration where the DIA is the server and the RIC computer is the client. Command and control communication is accomplished via CORBA with the DIA using ORBExpress as its Object Request Broker (ORB) and the RIC using the VisiBroker ORB.

A second relationship exists between the DIA and RIC computers when the DIA is disseminating data. The DIA provides data to its clients via the connectionless multicast-UDP. Thus, the DIA acts as a transmitter and clients of its data (the RIC included) act as passive listeners. The clients accomplish this by subscribing to the multicast address and port that the DIA is using. The DIA also uses multicast-UDP to transmit a “heartbeat” message (on a different multicast address and port) to allow a RIC computer to discover that the DIA is up and running and to provide information needed to facilitate binding for CORBA communication.

A third computer exists in the RICS system—the R2R converter. This computer provides a means for backward compatibility between the legacy Range Slaving System (RSS) and RICS. The RICS-to-RSS computer is intended as a temporary bridge until the remaining legacy sites at Eglin AFB are upgraded to RICS capability. The RICS-to-RSS software is implemented in C/C++ and runs under VxWorks.

DIA DATA FLOW

Data acquisition and transmission is accomplished by the DIA through a collection of I/O cards and threads. The GPS receiver card is used to initiate data collection and to time-tag the data. The HiRes card is used to record discrete mode changes at 100 nanosecond resolution. The W1W2W3 card collects azimuth, elevation, range, and pedestal angle track gate positions for up to four targets. The Acromag-IP carrier card is used to collect analog input and digital I/O data. The network card is used for data transmission and radar control.

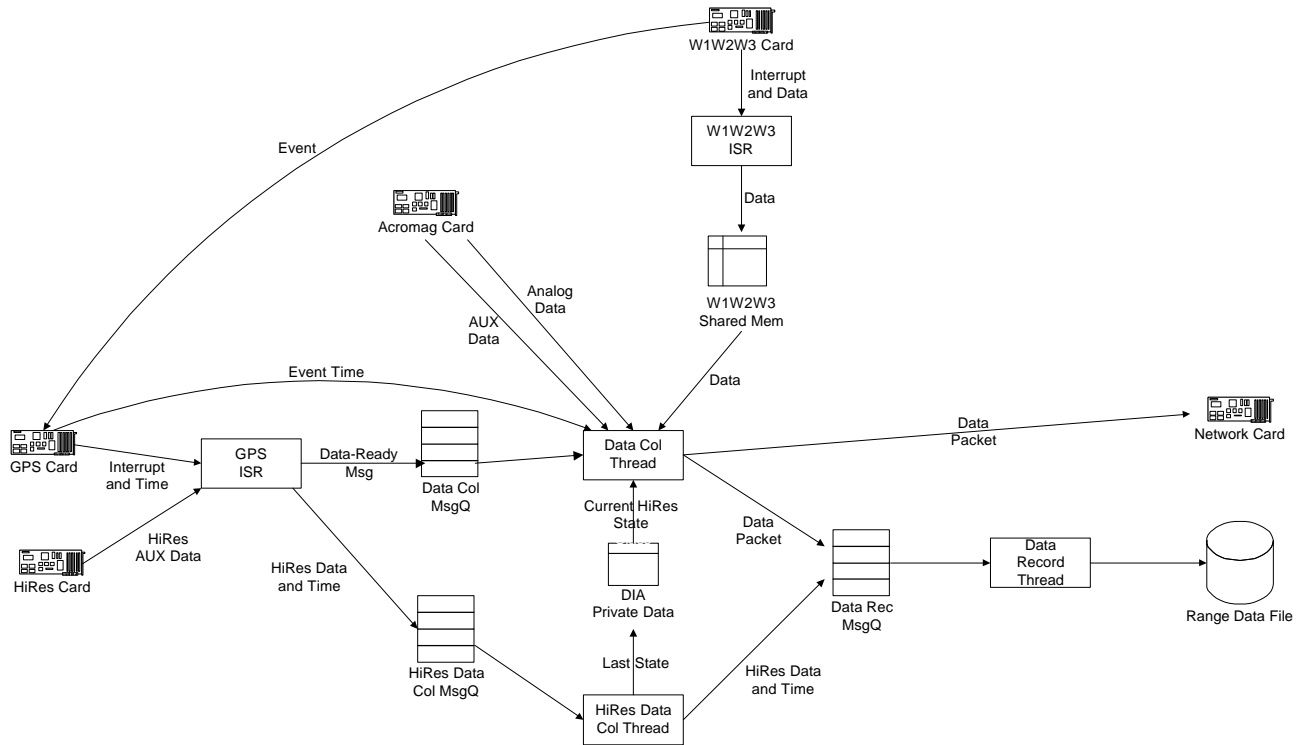


Figure 2. DIA Data Flow

Figure 2 shows the primary data flow through the DIA. Data collection is initiated via an interrupt from the GPS card. The GPS card can either generate interrupts itself at a periodic rate or interrupts can be triggered by an event from the W1W2W3 card. The detection of a GPS interrupt causes the GPS Interrupt Service Routine (ISR) to place a message on the data collection message queue to inform the data collection thread that data is ready. The W1W2W3 card generates an interrupt that causes its ISR to collect the current data and place it into shared memory for the data collection thread.

The GPS ISR also collects any available HiRes events and places them on the HiRes data collection message queue for processing. Upon receipt of the HiRes event data, the HiRes data collection thread places the HiRes data on the data record message queue and saves the last HiRes state for the data collection process.

The data collection thread sleeps until it receives a message in its message queue. Once the data-ready message is received, it begins the data collection process. First it obtains the event time from the GPS card. It then collects the current W1W2W3 data from shared memory, records the current HiRes data state, and reads the current analog and digital input values from the Acromag card. The data items are placed into a RICS format data packet that is multicast on the network.

The information collected by the Data Collection Thread and the HiRes Data Collection Thread is recorded by a separate thread during processing. This is accomplished by placing the data in the Data Record Message Queue for the Data Record Thread to write to mass storage.

DIA SOFTWARE DESIGN

The implementation of the DIA is based on an Object Oriented Design (OOD). The Unified Modeling Language (UML) was used to describe the software components (classes) and their interactions. Because of the large number of classes and the complexity of the system, the object model is broken into “packages” containing related and/or coupled classes. Figure 3 shows the dependencies between packages.

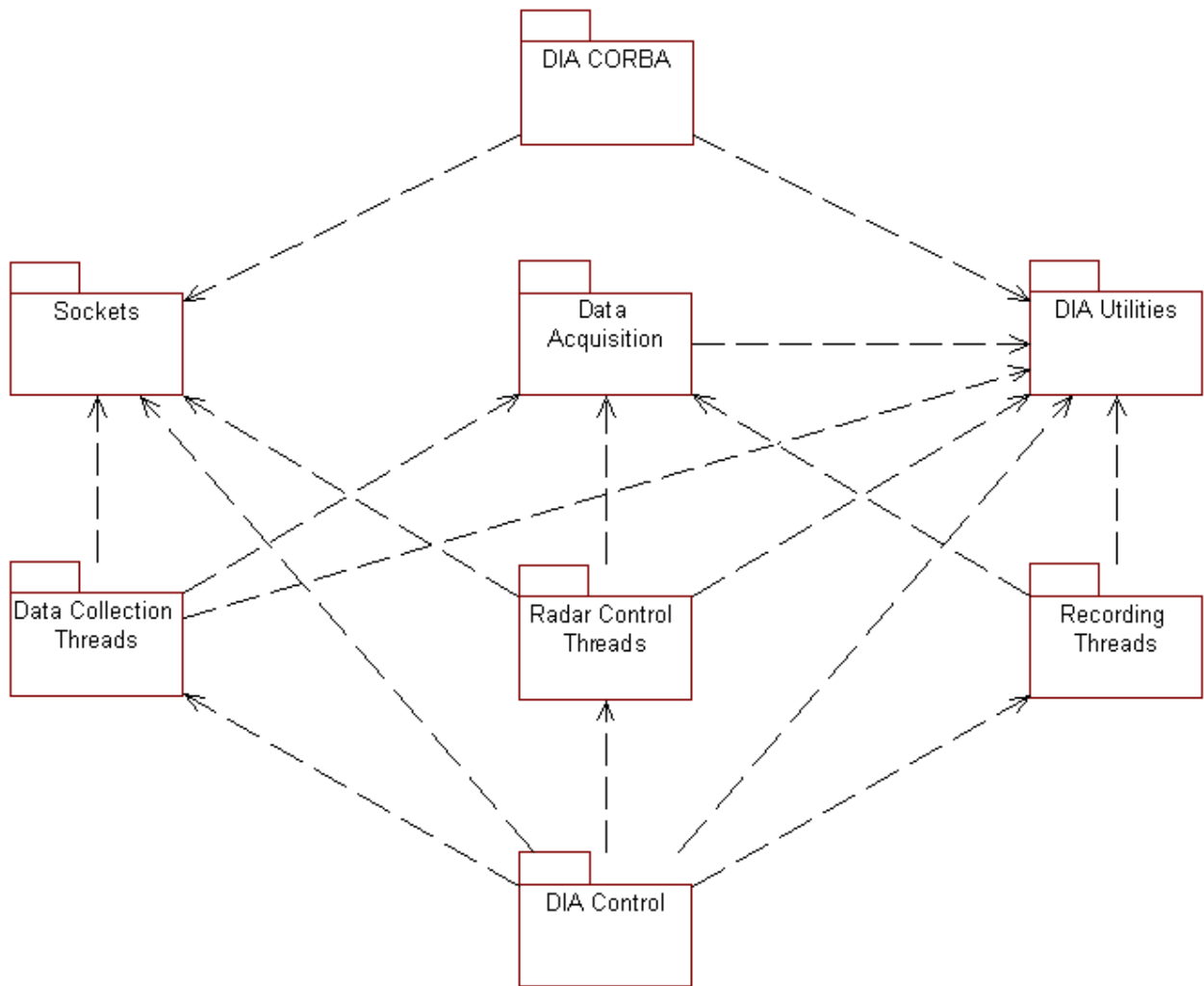


Figure 3. DIA Package Diagram

Each package represents a category of classes of the DIA. The DIA CORBA package contains the classes necessary to allow the DIA to be controlled remotely from a RIC computer. The Sockets package contains classes for reading and writing UDP datagrams. The DIA Utilities package is a collection of classes that are used throughout the DIA software. The Radar Control Threads package contains classes for radar slaving and slew tests. The Recording Threads package contains classes for logging, recording, and replaying RICS data. The DIA Control package contains classes for

managing threads and operating the DIA's console. The Data Acquisition package and the Data Collection Threads package will be described in detail below.

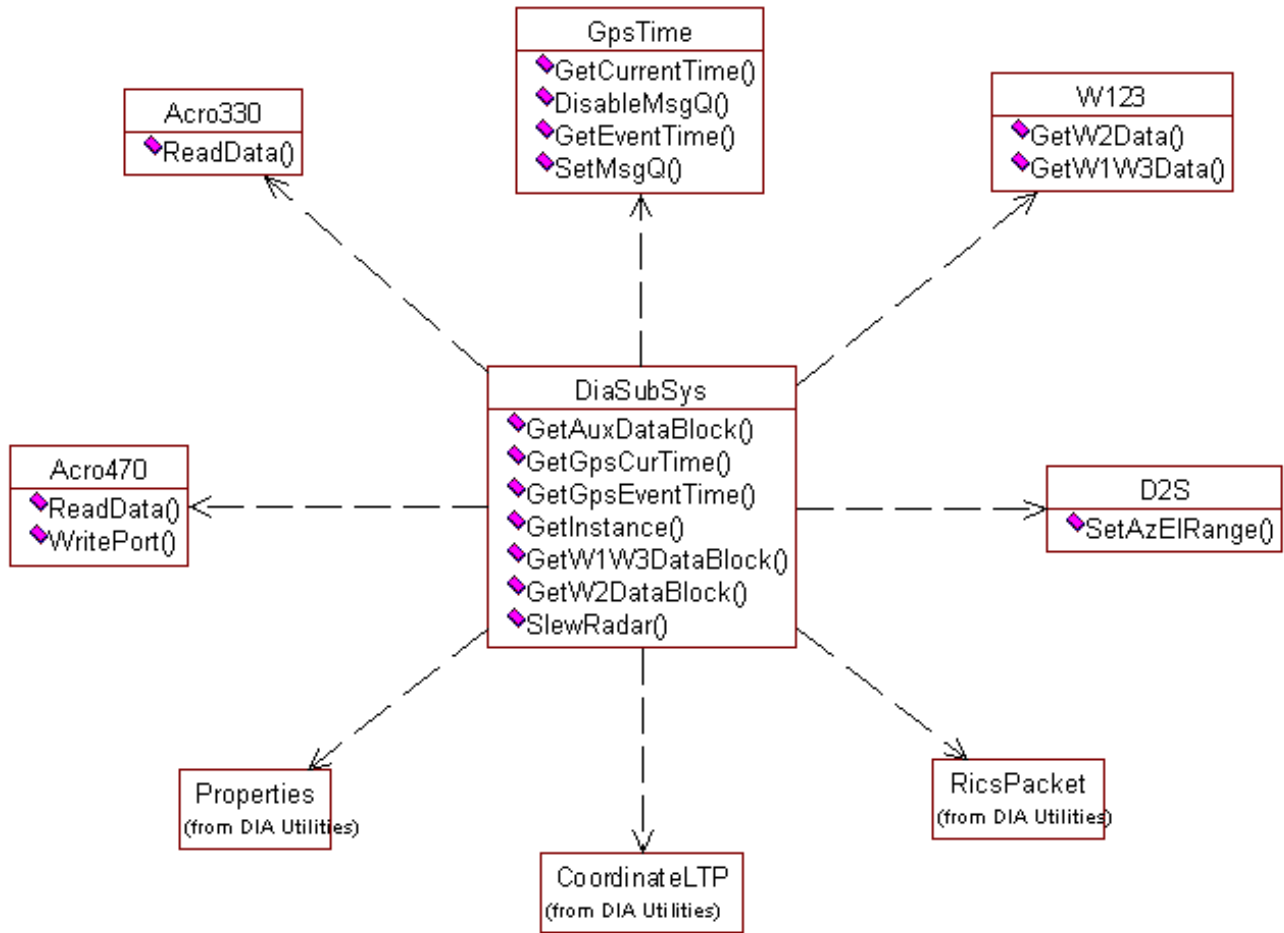


Figure 4. Data Acquisition Object Model

The object model in Figure 4 shows the Data Acquisition package in detail. It contains C++ wrappers for the drivers that access the data collection hardware. The DiaSubSys class implements the Façade design pattern. This pattern encapsulates the hardware wrappers to provide a simplified single point of access for the DIA software.

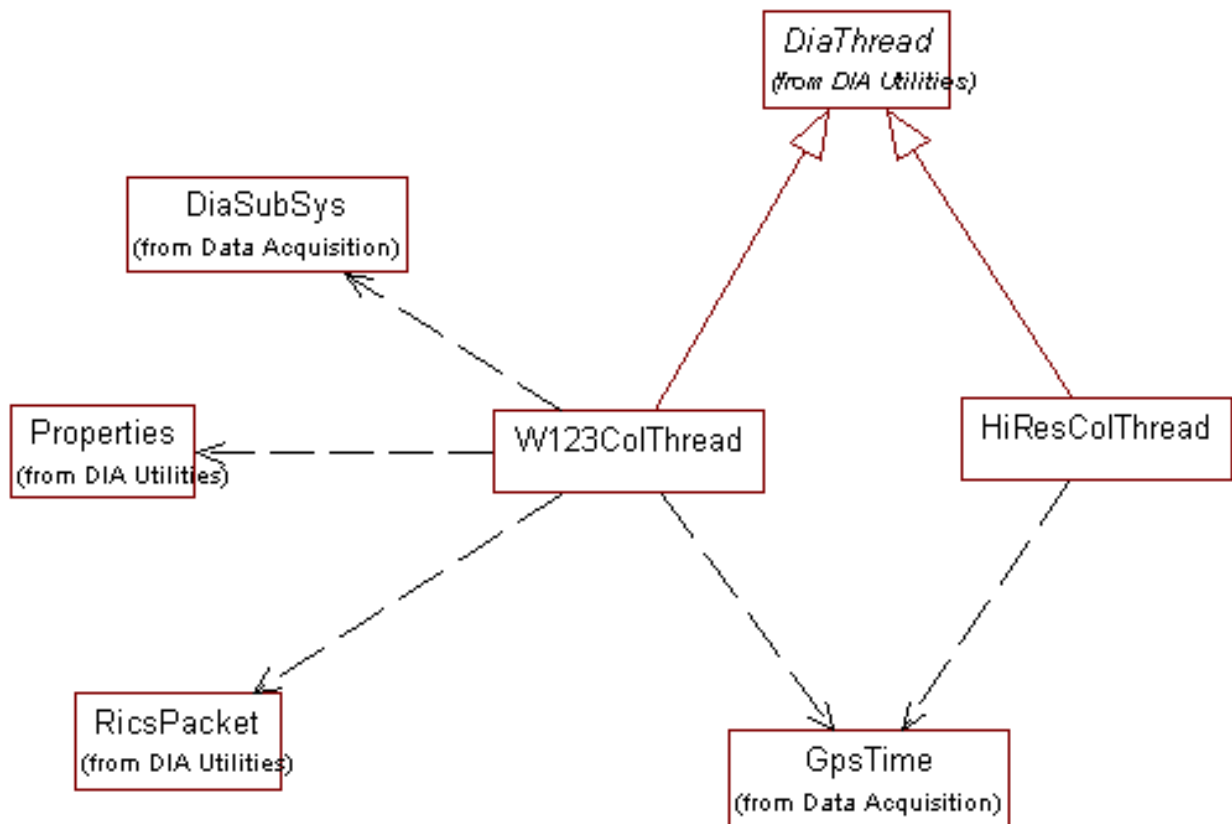


Figure 5. Data Collection Threads

The Data Collection Threads package (Figure 5) models the data collection thread class hierarchy. The *DiaThread* class is a base class containing general methods and attributes to implement threads under VxWorks. The *W123ColThread* class and *HiResColThread* class are derived from the *DiaThread* class. The *W123ColThread* class gathers radar data indirectly through the *DiaSubSys* class (part of the Data Acquisition package). The *HiResColThread* class receives its data via its message queue.

CONCLUSION

RICS is a complicated system spanning multiple computers and a diverse set of inputs. The object-oriented approach worked well for modeling the software design. The use of UML packages made the design manageable by dividing the system into distinct class categories. Its distributed nature lent itself to the use of a component architecture such as CORBA for client/server communication. At this time, the DIA software and the RIC software are complete and ready for testing. The system test, slated for the summer of 2001, will involve installing a RICS at an FPS-16 radar site at Eglin AFB, Florida.

ACKNOWLEDGMENT

The 96th Communications Group, Test and Analysis Division, Eglin AFB, Florida funded this effort.

REFERENCES

1. Recortec Inc., "Passive Backplanes and Single Board Computers," <http://www.recortec.com>.
2. Wind River Systems, "Tornado II." <http://www.wrs.com/products/html/tornado.html>.
3. Object Management Group, "Component Object Request Brokerage Architecture (CORBA)," <http://www.corba.org>.
4. Stevens, W.R., TCP/IP Illustrated, Volume 1, Addison-Wesley, 1994.
5. Datum Inc., "bc635PCI Time & Frequency Processor," <http://www.datum.com>.
6. Acromag, "PC-based IP Carrier Cards," <http://www.acromag.com/ioboards>.
7. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F.; Lorenzen, W.; Object-Oriented Modeling and Design, Prentice-Hall, Inc., 1991.
8. Fowler, M., Scott, K., UML Distilled, Addison-Wesley, 1997.
9. Gamma, E.; Helm, R.; Johnson, R.; Valissides, J.; Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.