Memorandum

To: J. Webber

From: J. Effland

Date: 2003-05-23

Subject: Unified Software Design for ALMA Front End Test Systems

Introduction

An essential element for the ALMA front-end project is the software required for testing the following systems:

SIS mixers,
cartridges, and the
overall receivers.

Early plans assumed each cartridge manufacturer would develop their own software for testing their assigned mixers and cartridges, and pass (either electronically or on paper) measurement data to a central repository. Scheduling demands and fiscal realities may now require more unified approaches to test system construction, including software development. That means that either entire programs should be shared among different groups, or at a minimum, software modules could be shared by reusing code already written and designed by others. This memo discusses some concepts and issues for unifying the software development effort. Also included is an appendix showing how mixer I-V data are stored and retrieved from a database.

Common Programming Language

A common programming language is essential for code reuse. As discussed in later sections, there should be significant opportunities for code sharing between the cartridge and receiver test systems.

There are two essentially incompatible programming languages that are commonly used for instrumentation control:

LabVIEW, and
Visual Basic.

LabVIEW is a graphical development tool that allows rapid automation of instrumentation systems. A huge number of virtual instruments, essentially drivers for various instruments, have been developed over the years to support LabVIEW applications. Program flow and interfacing is created graphically in LabVIEW by connecting lines to various boxes representing virtual instruments. LabVIEW has been used successfully by a number of groups at the NRAO to develop small-to-medium scale test programs. Another reason for LabVIEW's success is that the user doesn't need to learn the mechanics of object-oriented software development to use LabVIEW.

Visual Basic is a combination graphical and textual-based system also used for controlling instrumentation systems. Microsoft has recently rewritten Visual Basic from scratch (now called Visual Basic.NET) to allow all the primary Microsoft compilers; Visual Basic, Visual C++, and Visual C#, to call routines written in each language. Database integration in Visual Basic is very strong, with an entire Application Programming Interface written to support database interfacing, called ADO.NET (ActiveX Data Objects).

Visual Basic.Net finally supports true object-orientation, and in particular fully implements the tremendously important concept of class inheritance. In addition, robust error handling has been included in the form of "Try" and "Catch" exception handling to replace the horribly antiquated "On Error Goto" feature of the earlier versions.

It remains unclear which of these two programming languages is optimum for software development, but it is the author's and Philip Dindo's (of HIA) opinion that Visual Basic is better suited for the following reasons:

greater design flexibility,
better suited for large-scale applications,
better database interfaces, and
more thorough documentation.

Virtual instruments available in LabVIEW are incredibly convenient, assuming that they meet the requirements of the task. Virtual instruments and the graphical connection and interfacing concept are the primary reason that instrumentation systems can be automated rapidly using LabVIEW. However, virtual instruments can be cumbersome to change if additional, but unsupported, features are needed. In addition, the test systems required will most likely evolve into large software development projects, and LabVIEW's use for such large systems is questionable. In fact, it has been reported that large, complex logic loops can be difficult to program in LabVIEW.

Database interfacing in Visual Basic is extremely robust and employs the same routines used in C++ and C#. LabVIEW's database interfacing is available through the optional "Enterprise Connectivity Toolset", but most likely at reduced flexibility compared with Visual Basic's routines.

Documentation in LabVIEW consists of a group of drawings depicting the interfaces and signal flow between each virtual instrument. Classes designed in Visual Basic are amenable to formal documentation methods such as Class, Sequence, and Activity Diagrams that are part of the Unified Modeling Language (UML). UML is now used for most documentation in large software designs.

Software Reuse Possibilities by Tasks

The following sections describe the feasibility of reusing software between the major front-end tasks, which include mixer tests, cartridge tests, and receiver tests.

1.1 Mixer Test Software

Each ALMA cartridge builder employs their own mixer development group that uses varying types of new and legacy hardware and software, so common test interfaces for mixer testing may be difficult to realize. However, test data should be similar for all mixer tests to facilitate review and archiving of the data. Perhaps each mixer manufacturer should be allowed to continue using their own custom software, but the could be required to store mixer data in a database that's either common among all the groups, or multiple databases designed from a common schema.

1.2 Cartridge Test Software

The hardware interface to control cartridges will most likely be identical for all the bands, so all builders should be able to use a common the software interface, at least for cartridge control. This interface should be programmable, that is, it should consist of a number of routines callable from other software modules. Designed this way, the cartridge builders can then, if desired, write their own display routines that use the standard set of interface routines.

Both the hardware and software for instrument control for the cartridge test system should also be shared between the cartridge builders where possible. For example, the CDL and HIA are sharing designs and classes for chopper wheel construction and control.

1.3 Receiver Test Software

Receiver test software, used at the integration centers, will have many similarities to cartridge test software, and hence could greatly benefit by reusing code from the cartridge manufacturers.

Common Format for Deliverable Data

To simplify the management of measured results, each cartridge builder and mixer tester should use the same format for both storage and display of performance data.

Perhaps requiring each group to populate a database with identical schema would force each builder to adhere to a common format and also would provide data in a readily available means.

Maintenance and Reliability

The software will be used for a long period of time as ALMA is constructed and operated. Proper software documentation and design is essential for others to be able to maintain the applications after the original designers move on to other tasks. For example, one design problem that occurs with the CDL's mixer measurement software is that reinstallation after a computer failure is exceedingly difficult.

Recommendations

The following recommendations are provided to promote discussion among the software designers assigned to the ALMA Front End task.

1. Agile programming methods should be used for all software development. This means writing requirements down, but realizing that some requirements are undefined and others will change. Agile programming also means delivering additional software functionality on a monthly basis.
2. Developers of cartridge test software should design reusable components into their software and share as much code as possible with other cartridge groups.
3. Integration center software should reuse much of the cartridge test software.
4. Visual Basic is the software of choice for development of cartridge and receiver test software, although LabVIEW could be used for mixer test routines.

Acknowledgments

The author wishes to thank Philip Dindo of the Herzburg Institute for Astrophysics, Dorsey “Skip” Thacker, and Jim Pisano both of the CDL for their useful comments.

Appendix: An Implementation Example

Figure 1 shows a sample architecture for a generic measurement system. A variant of this scheme has been used at the CDL for about 4 years to store and retrieve mixer I-V data.

The mixer I-V data acquisition routines, written in stand-alone Visual Basic, allow the user to view and enter test parameters. The data acquisition program stores the results by accessing the database through a set of common classes implemented as an "ActiveX" dynamic link library[1]. Data retrieval and analysis occurs in Excel using macros written in Excel's version of Visual Basic (called Visual Basic for Applications). The Excel routines allow selection of a particular record or set of records from the database as shown in Figure 2. After selection, the routines return the desired data to a spreadsheet in Excel, and other routines produce graphs of the results as shown in Figure 3. Using Excel in this manner as the presentation program has the advantage that the user is then able to easily modify the data or the graph.

Future implementations could provide the capability to upload and download the data to the ALMA operational database.

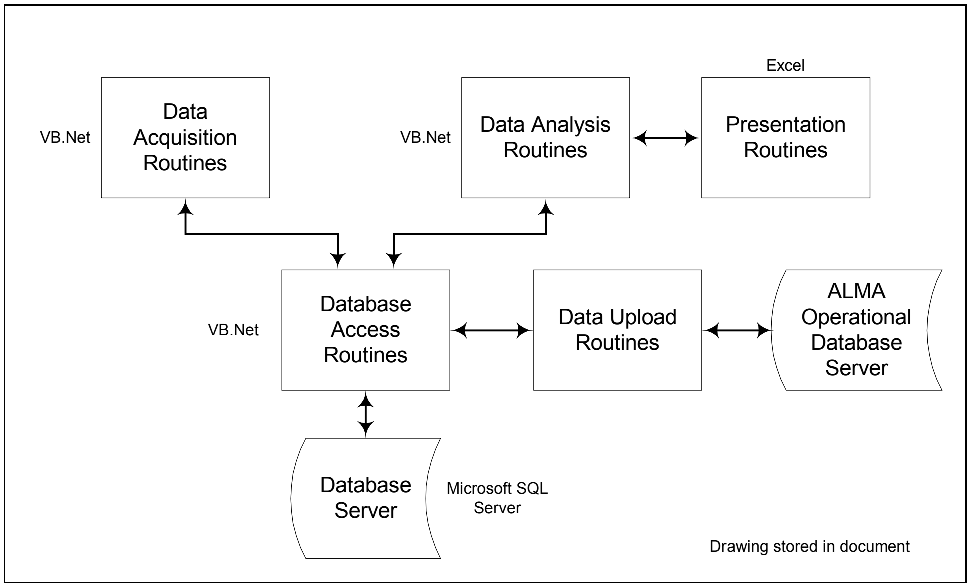
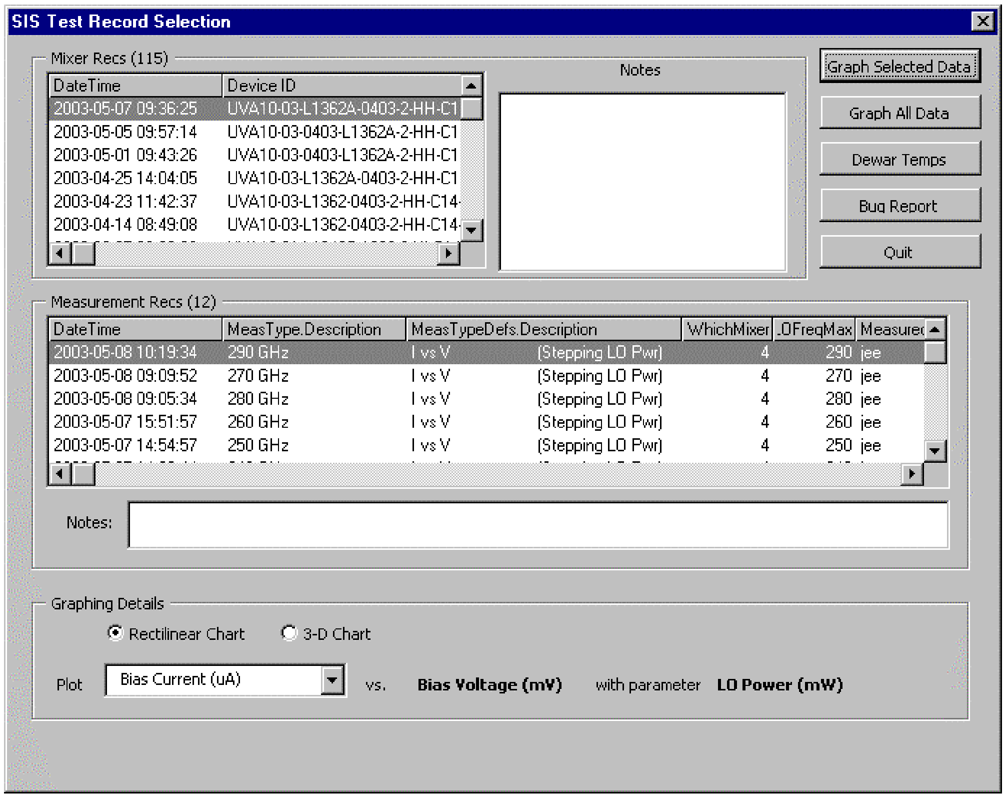


Figure 1: High Level Proposed Data Flow Diagram

[1] Note that many of the classes are being rewritten in Visual Basic.Net, which provides a "COM" wrapper to allow continued ActiveX support. Future versions of Excel are reported to interface directly with the managed code from Visual Basic.Net and hence will eliminate the need for a "COM" wrapper.



SIS Test Record Selection

Mixer Recs (115)

DateTime	Device ID
2003-05-07 09:36:25	UVA10-03-L1362A-0403-2-HH-C1
2003-05-05 09:57:14	UVA10-03-0403-L1362A-2-HH-C1
2003-05-01 09:43:26	UVA10-03-0403-L1362A-2-HH-C1
2003-04-25 14:04:05	UVA10-03-L1362A-0403-2-HH-C1
2003-04-23 11:42:37	UVA10-03-L1362-0403-2-HH-C14-
2003-04-14 08:49:08	UVA10-03-L1362-0403-2-HH-C14-

Notes

Measurement Recs (12)

DateTime	MeasType.Description	MeasTypeDefs.Description	WhichMixer	.OFreqMax	Measurer
2003-05-08 10:19:34	290 GHz	I vs V (Stepping LO Pwr)	4	290	jee
2003-05-08 09:09:52	270 GHz	I vs V (Stepping LO Pwr)	4	270	jee
2003-05-08 09:05:34	280 GHz	I vs V (Stepping LO Pwr)	4	280	jee
2003-05-07 15:51:57	260 GHz	I vs V (Stepping LO Pwr)	4	260	jee
2003-05-07 14:54:57	250 GHz	I vs V (Stepping LO Pwr)	4	250	jee

Notes:

Graphing Details

[✓ Rectilinear Chart] 3-D Chart

Plot Bias Current (uA) vs. **Bias Voltage (mV)** with parameter **LO Power (mW)**

Figure 2: Excel-Based Dialog for retrieval of Mixer I-V data.

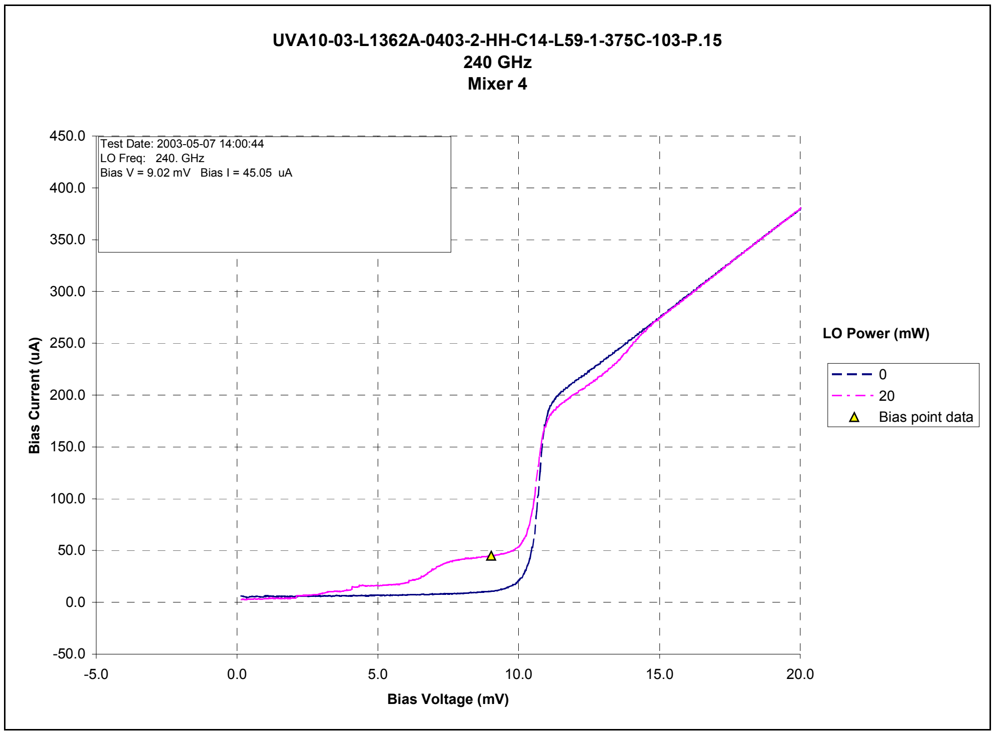


Figure 3: Mixer I-V data Graphed in Excel