An Enterprise Software Architecture for the Green Bank Telescope (GBT)

Nicole M. Radziwill, M. Mello, E. Sessoms and A. Shelton
National Radio Astronomy Observatory, P.O. Box 2, Green Bank, WV 24944 USA

ABSTRACT

The enterprise architecture presents a view of how software utilities and applications are related to one another under unifying rules and principles of development. By constructing an enterprise architecture, an organization will be able to manage the components of its systems within a solid conceptual framework. This largely prevents duplication of effort, focuses the organization on its core technical competencies, and ultimately makes software more maintainable.
In the beginning of 2003, several prominent challenges faced software development at the GBT. The telescope was not easily configurable, and observing often presented a challenge, particularly to new users. High priority projects required new experimental developments on short time scales. Migration paths were required for applications which had proven difficult to maintain. In order to solve these challenges, an enterprise architecture was created which includes five layers: 1) the telescope control system, and the raw data produced during an observation, 2) Low-level Application Programming Interfaces (APIs) in C++, for managing interactions with the telescope control system and its data, 3) High-Level APIs in Python, which can be used by astronomers or software developers to create custom applications, 4) Application Components in Python, which can be either standalone applications or plug-in modules to applications, and 5) Application Management Systems in Python, which package application components for use by a particular user group (astronomers, engineers or operators) in terms of resource configurations. This presentation describes how these layers combine to make the GBT easier to use, while concurrently making the software easier to develop and maintain.

Keywords: architecture, application programming interface, API, scheduling block, execution block, GUI

1. INTRODUCTION

Several interconnected software systems are required to operate the Green Bank Telescope (GBT), which is a highly complex instrument incorporating several receivers and many diverse backends. [1] At the beginning of 2002, five core software applications were involved during the course of an astronomical observation. In order of their appearance to the astronomer during a typical observing session, these were:

- GBT Observe (**GO**), a GUI application written in glish
- Control Library for Engineers and Operators (**CLEO**), a Tcl/tk GUI interface
- **Ygor**, a distributed object-oriented telescope control system in C++
- Interim Analysis and Reduction of Data System (**IARDS**), an aips++ product for "quick looks" at data, and
- **DISH**, the aips++ single-dish data reduction environment.

These applications were written and managed by different individuals who were not required to share the same technical or conceptual foundations. Additionally, there were several gaps in functionality at the GBT. For example, the system could not be configured without communicating directly to the control system through CLEO, which required expert knowledge of the devices and underlying electronics. The system was not easily accessible to novice users. Because the Green Bank technical infrastructure was also incompatible with longer term Observatory-wide technical goals (such as remote observing and dynamic scheduling) and Observatory-wide organizational goals (to increase collaboration and reduce duplication of effort), the holistic approach to software development and implementation supported by the development of an **enterprise architecture** became necessary.

Contact Information: N. Radziwill, Phone 304-456-2139, nradziwi@nrao.edu. M. Mello, Phone 304-456-2143, mmello@nrao.edu.

2. ENTERPRISE ARCHITECTURE DEFINED

Enterprise Architecture (EA) is defined in many ways. The Commonwealth of Virginia calls it "the bridge between business and technology." [2] Scott Ambler, author of *The Practical Guide to Enterprise Architecture*, defines it as "the models, documents, and reusable items that reflect the [technical] architecture." [3] The Federal Government considers an EA to be the critical linkage between mission, strategy, processes, and technology implementations, captured "using multiple architectural models or views that show how the current and future needs of an organization will be met." This definition continues and asserts that "by focusing on strategic differentiators and working across the enterprise, there is a unique opportunity to create leverage and synergies and avoid duplication and inconsistencies across the enterprise. [4]

2.1. A Definition for Astronomical Systems Development

As software specialists in the astronomical field, our business is enabling science. In order to do this, we must effectively bridge the gaps between the observer domain, the telescope-specific domain, and the science domain. The observer domain encompasses the translation of scientific intent to a proposal, and submitting the proposal to an observation preparation system, which necessarily has a telescope-specific component. The execution of the observation, and production of data, are all telescope-dependent. The reduction and analysis of data are all in the science domain, since the approaches and algorithms used can be the same from instrument to instrument, and are not necessarily dependent upon the telescope. [5]

Drawing from the wealth of available definitions of EA, then, we propose that for astronomical systems development an EA is described as "**the group of models and documents that concisely illustrates the bridge between the observer, the telescope-specific and the science domains, in order to create leverages and synergies and avoid duplications and inconsistencies**." These can be derived from Observatory-wide directives, site-level goal statements and plans, requirements documents, design documents, or implementation details as necessary. These models must fully describe, at a conceptual level:

- how people are to interact with programs
- how programs are to interact with programs
- how varying degrees of abstraction are logically and accurately addressed through systems implementation

One critical point is the definition of an enterprise. Because the GBT project operates independently of other NRAO projects in many ways, but is required to coordinate with other projects to achieve shared results, Green Bank should thus be thought of as an enterprise within the greater enterprise (which is the Observatory itself). This means that there can be a unique enterprise architecture for the GBT, however, it must be integrated and aligned with an Observatory-wide approach for it to be truly useful as a strategic tool. For Green Bank, producing an EA has been a necessary **precursor** to productive participation in Observatory-wide software planning exercises.

An enterprise architecture is the product of systems integration work. Systems integration has been defined as "a progressive and iterative cycle of melding technologies, human performance, knowledge, and operational processes together." [6] In the case of Green Bank software development, two levels of systems integration are necessary. First, software applications local to the site must be normalized according to a consistent framework. Those applications must then be made consistent with an Observatory-wide approach, reflecting an "enterprise within an enterprise" structure.

2.2. Critical Success Factors

For an enterprise architecture to be successful at Green Bank, it must achieve several goals. First, the GBT must be easier to use when software systems are implemented according to an enterprise architecture (that is, any user should be able to interact with the telescope system at the level of abstraction that best meets their needs). Software should become easier to maintain, as it becomes clear which layer (and thus which products) should include which functions. The learning curve for programmers should be shortened because programmers instintively know which layer to look in

if prepackaged functionality is required for their development activities. The learning curve for users should be shortened, because they can access the telescope in novice mode or expert mode as necessary. Software products should become algorithmically consistent as duplication of code is minimized. Migration paths should be evident for all observer-facing applications which are out of date (e.g. GO, IARDS).

With respect to strategic goals, an EA should provide the clarity required to meet them. For example, two key goals for the GBT are ease of use and reducing the learning curve for GBT software systems. This is a common end result of an EA process, in which technical architecture is blended with scientific drivers and a simple, unified application architecture. According to an online source for EA professionals, "an architected system will usually provide a common look and feel that makes all systems using it seem more familiar, and therefore easier to learn and use." [7]

2.3. Levels of Abstraction and the Applications Programming Interface (API) Concept

The key to a successful EA is in defining the levels of abstraction that will be required. The GBT, and possibly other telescopes, require five levels: one for the astronomer new to the telescope, one for the expert or staff astronomer, one for the applications programmer, one for the lower level utilities programmer, and one for the hardware engineer.

One concept germane to an EA is the application programming interface, or API. According to Wikipedia, "a good API provides a 'black box' or abstraction layer, which prevents the [user or] programmer from needing to know how the functions of the API relate to the lower levels of abstraction." [8] In the complex telescope environment, it is likely that there will be several APIs created for unique, yet cooperative purposes.

An important philosophical consideration is the difference between a "User's API" and a "Programmer's API". Within the term itself, application "programming" interface implies that this is a tool for use by programmers. However, in astronomy in particular, users may be sophisticated programmers themselves and might desire the level of functionality that is usually reserved for software engineers. For this expert user case, software engineers and astronomers must carefully negotiate exactly how simplified an API is to be, with detailed consideration of the pertinent levels of abstraction.

3. COMPONENTS OF THE GREEN BANK ENTERPRISE ARCHITECTURE

Recalling that an EA is simultaneously a product and a process, the Green Bank Enterprise Architecture consists of several documents and models, some of which are in place already, and others which are still under development. This process has been underway for approximately two years. The artifacts include:

1. **Science Drivers**. At present, the primary science drivers for the GBT include making the telescope easier to use, shortening the learning curve for new observers, enabling remote observing, enabling automated dynamic scheduling by implementing Scheduling Blocks, [9] and delivering new instrumentation that will extend the scientific capabilities of the instrument. Within the next few years, pipeline processing will become a primary science driver. Understanding the key science drivers now and in the future provides individuals with the ability, at design time, to accommodate for future needs and prevent the requirement for complex rework and redesign in the future.
2. **Standard Observing Modes**. A comprehensive document outlining the standard observing modes for the GBT is in progress. By understanding which modes are to be supported for every completed software package, designers and developers will be better able to plan their work and test their results.
3. **Configuration/Setup Cases**. Drawing from the standard observing modes, there should be several predefined telescopes setups that are applicable, at least one per observing mode. Configuration cases have already been established for typical scientific experiments on the GBT. [10]

4. **Observing Cases**. For each of the standard observing modes and configuration cases, there should be one or more recipes for carrying out an observation. These are currently under development for the GBT, and will include considerations such as whether the observer is targeting a strong or weak source. A Scheduling Block will consist of project identifiers, constraints for how and when the Scheduling Block should be executed, and most importantly a collection of paired configuration cases and observing cases. More than one observing case may be linked to a single configuration, if applicable for an observer's experiment.

5. **Data Reduction Cases**. Once there is a complete set of configuration cases and observing cases which can be chained together, a suite of data reduction cases will be compiled. These will illustrate, in a technology-neutral way, what steps are required to reduce the data set produced when a Scheduling Block (consisting of a configuration case-observing case pairing) is executed on the telescope. This analysis deliverable, scheduled for completion during the summer of 2004, will be used to test out data reduction capabilities within multiple analysis packages.

6. **Calibration Strategies**. In a mature telescope it is expected that there will be a clear recommended calibration strategy for each pairing of observing mode, configuration case, and observing case. However, the GBT has not yet reached the stage where it can provide calibrated data products for all observing modes in an automated fashion. This component of the Green Bank EA is a document describing the steps required for calibrating data, the algorithms available for each step, and initial recommendations for how to chain together the algorithms for each standard observing mode in an astronomically meaningful way. This addition is deliverable in May 2004.

7. **Performance and Scalability Requirements**. It is important to identify whether designs are feasible or not early enough in the development process so that appropriate technical paths are chosen to implement software solutions. This year, these constraints will be outlined for all software applications for the first time. By anticipating how the telescope will be used in the future, software designs can be adjusted accordingly.

8. **Technical Architecture & Application Architecture** (Definition of Layers). Described in §4 below, the technical architecture is the set of principles in place to provide consistency between applications and reduce rework. The application architecture shows which software products and modules fall within which layers of abstraction.

9. **Unified Data Storage**. Currently, there are several disparate databases used by the telescope control system, the configuration utilities, and the Scheduling Block executor which is under development. To reduce rework and encourage reuse of code, the information architecture underlying all applications within Green Bank should be consistent and unified. Although this is required future work, it will not be scheduled until 2005 or 2006, as it will follow the data archive model currently under development by the NRAO e2e Oversight and Architecture Committee. [11]

10. **Organizational Process**. In order to deliver any functionality congruent with science drivers, it is important to have a management process in place for the review of technical deliverables within the strategic context. At Green Bank, this important function is carried out by the Project Planning Committee, and is described in more depth elsewhere in this conference. [12]

It is anticipated that additional models will augment or be used to adjust the Green Bank EA as more is learned by the software engineering staff about future instrumentation, such as the Penn Array Camera. Other models, under development by the NRAO e2e Oversight and Architecture Committee, will also be important parts of the Green Bank EA. These new Observatory-wide models will provide the framework for the GBT to integrate its EA effectively into the context of the larger Observatory enterprise.

4. LAYERS

A layered application architecture is the foundation for providing functionality at varying levels of abstraction. Green Bank uses the structure outlined in Figure 2.

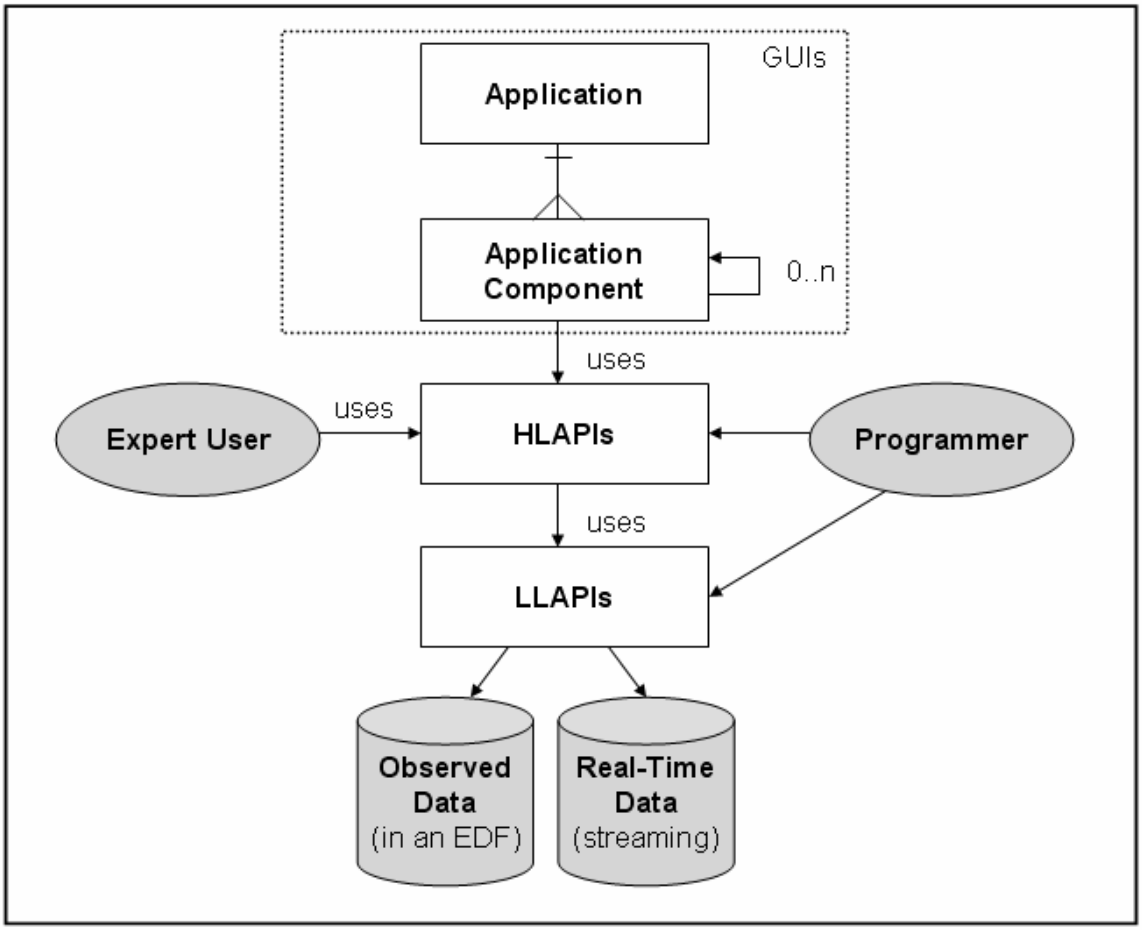

Figure 1: Five layers of the technical architecture in Green Bank, from Applications (top) to Data Systems (bottom). (EDF stands for "Export Data Format".)

4.1. Applications

Applications consist of a logical grouping of application components. All applications use a resource configuration, which dictates how the application components are joined together to provide useful services for a particular user or type of user. Applications should have no functionality on their own, other than to organize application components in ways that are logical to various kinds of users, such as astronomers, engineers or operators.

4.2. Application Components

An application component provides a unique set of functionality. Examples of application components at the GBT include Logview (which is used to view and plot telescope monitor data), the Dynamic Corrections Monitor, and GBT FITS Monitor (GFM) which is used for real-time quick looks at data. Application components can be executed stand-alone, or within the context of an application management system. They may consist of plugins, which provide subsets of functionality. Plugins are also application components, with the constraint that they are only useful within the context of a container which is an application component itself. Individual Graphical User Interface (GUI) applications are crafted at this level. Although it was developed nearly five years ago and not in the context of the EA, the CLEO application was designed with similar principles in mind. Its developer has found that this approach results in high productivity and low maintenance costs. [13]

4.3. High Level APIs

The High-level APIs are the programming infrastructure used to create GUIs and other application components, including the Scheduling Block executor which is currently under development (see Figure 2). They supply a conceptual context to lower-level telescope-specific functionality in an astronomically meaningful way. This is the primary layer at which the shift in abstraction from the observer domain to the telescope domain becomes evident. Expert users may wish to interact with the Scheduling Block executor more directly, by writing their own observing procedures, for example, using the Observing API. Typical users, however, will be able to run their experiments and reduce their data without any programming, by relying upon the functionality provided to them at the application component and application layers.

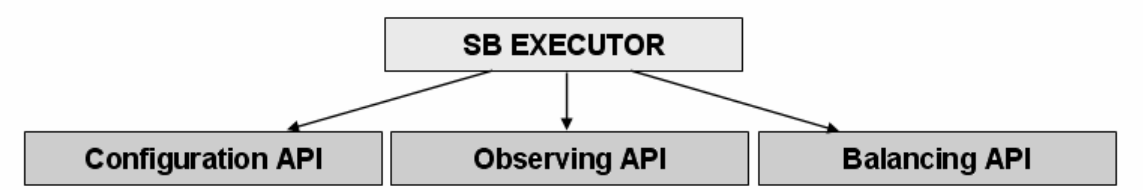

Figure 2: The Scheduling Block executor is designed as an application component which uses High-Level APIs.

4.4. Low Level APIs

The Low-level APIs provide the means for accessing data independent of a structured data format, and for accessing the core parameters of the telescope control system without requiring knowledge of how the control system is implemented. Data access is accomplished using FITS Query Language (FQL), a custom implementation of the industry standard Structured Query Language (SQL) for database access. A group of one or more FITS files can be used to "simulate" a database from which data in extracted in a non-sequential manner. [14] Standard access to the telescope control system is being provided by a new SOAP interface called Grail,[15] which brokers all observing-related transactions with the underlying distributed software systems.

High-level APIs package the use of Low-Level APIs for specific purposes.

4.5. Data Systems

Data systems are the cornerstone of the application and technical architectures. Either the data will already be in existence on disk as observed data, which can be extracted from an Export Data Format (EDF), or it will be accessible in real-time, streaming from some part of the telescope control system. Low-level APIs such as FQL and Grail are typically used to access data at this level.

By abstracting away the data access from the underlying data format, software in Green Bank is much less sensitive to changing data formats. No code changes are required at the High-level API level and above when there is an adjustment to the format of the output data.

5. EXAMPLES OF NEW APPLICATIONS

Examples of applications at the different layers are shown in Figure 3. Many of the following developments are at the alpha or beta stage, and will be made available for general use by visiting astronomers at various times throughout 2004 and 2005.

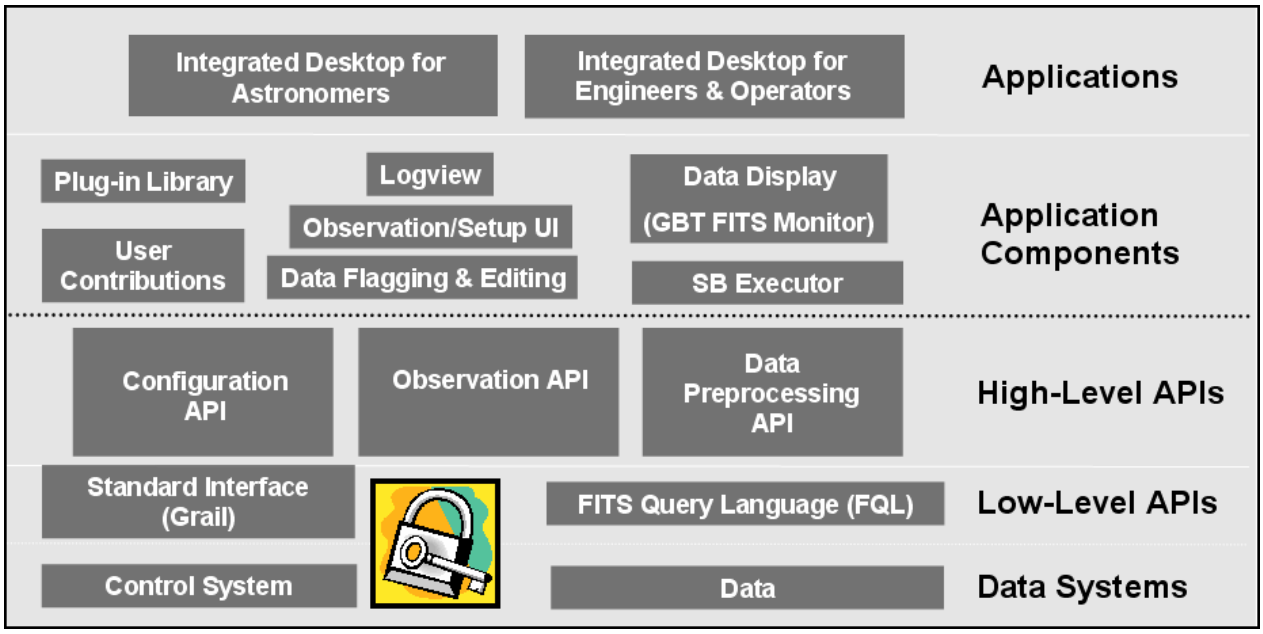

Figure 3: This application architecture "cartoon" shows examples of some applications at the various layers.

5.1. Typical User Scenario

The enterprise architecture is only useful if it helps software engineers deliver benefits to astronomers. By using the components of the EA described earlier, the following experience for the typical visiting observer to the GBT is expected in the future:

- User sets up their **User Preferences** file, if desired
- User defines a **Scheduling Block** to conduct their observation, either by using a text editor or the Scheduling Block builder application. They are able to validate their telescope setup and observing procedure offline.
- At observation time, the **Scheduling Block Executor** sends the observation to the telescope.
- The Scheduling Block Executor looks at the User Preferences file to determine how the **Online Filler** should preprocess data using the **Preprocessing API**
- The observer may use **Astrid**, the Integrated Desktop, to ease real estate management on the observer's display
- The observer and/or operator use the **GBT FITS Monitor (GFM)** for a real-time quick look at the incoming data. GFM **Plugins** apply "least common denominator" data reduction recipes for real-time look, and operates on raw GBT data. It detects which type of scan is incoming & processes accordingly.

After the observing session, the observer has a number of choices available for data reduction: AIPS++/DISH, CLASS, or IDL can be used depending upon the observer's science goals and package preferences. [16]

5.2. User Preferences

The default behavior for GBT applications is contained in a global configuration file. Individual astronomers do not need to have their own user preferences file to run GBT applications, but can override the default behavior by having their own stored in their $HOME directory.

Preferences include: Which data directory to store a project in, and eventually, which output formats are desired (online filler), which data display plugins an observer would like to use, and in the future, which data preprocessing functions are desired (e.g. the Van Vleck correction). Live Data *overrides* User Preferences *overrides* Global Preferences. For

example, focus correction coefficients produced dynamically are stored in the Antenna FITS file. These values will override the default coefficients in your User Preferences file, but if no values are available in the FITS file or your User Preferences file, the coefficients in the global preferences file will be used.

```
[Algorithms]
gbt.calibration.Continuum:    gbt.calibration.KaContinuum
gbt.calibration.SpectralLine: gbt.calibration.SpectralLine

[Data]
DefaultDirectory: /home/gbtdata
ScanLogName:      ScanLog.fits

[Plugins]
Sort Order:   Pointing Focus DCR Spectrometer
DCR:          plugins.dcr.DCRGenericPlugin
Focus:        plugins.dcr.FocusPlugin
Pointing:     plugins.dcr.PointingPlugin
Spectrometer: plugins.spectrometer.SpectrometerPlugin
Tip:          plugins.dcr.TipPlugin

[Pointing]
LPC File: /home/gbt/etc/config/LPCs.conf

[Focus]
Antenna Config File: /home/gbt/etc/config/antenna.conf
LPC File:            /home/gbt/etc/config/LPCs.conf
Y_A: -180.630
Y_B:   66.189
Y_C:  196.949
```

Figure 4: Example of a user preferences file showing how a user can customize their experience of GBT software.

5.3. Configuration API

The Configuration API has been in use at the GBT since November 2003. It can be used through the Python command line interactively, or by preparing a script and executing the entire script at once. The utility is intended for expert users (novice users would simply prepare their Scheduling Block and never interact with the API itself). This important utility at the GBT provides the ability to simply configure hundreds of parameters on the telescope by specifying a small subset (<15) of astronomical metakeywords. It also provides the ability to directly set control system parameters, for the experts among the experts, and it provides diagnostics including the ability to check devices, the state of the IF path, frequencies and velocities of the LO1 and LO2, channels, switches, drivers and modules in use.

5.4. Observing API

The Observing API, nicknamed “Turtle”, is modeled after the LOGO graphics builder utility of the same name. Currently being prepared for a beta release, this provides a series of building blocks that observers may utilize to create complex movements of the beam across the sky or even create their own observation procedures. Turtle will also provide a pre-defined suite of commonly used procedures created by on-site astronomers for observers, which in turn may also be used as building blocks to create even more complex beam movements. Within a Scheduling Block, if a user specifies that they wish to perform a Peak scan, it is the Observing API that will execute that movement and ensure that data is taken.

5.5. GBT FITS Monitor (GFM)

The GBT FITS Monitor (GFM) was initially developed in response to the needs of the PTCS project for analyzing continuum data, and has been in use since early 2004. It is intended to become a complete replacement for IARDS, which has been used previously for real-time data display. GFM's non-astronomical components (plotting, zooming, etc) are accomplished through the 3rd party integration of DEAP, an application developed by candidates for the Master's in Software Engineering at the University of Maryland. [17] This application provides scan-based display and analysis of GBT data, either in real-time as the data is being collected, or in an offline mode where it can be used to simply step through the scans from an observation. The same algorithms used for "quick look" are applied during the playback. Because it can work either in either online or offline modes, it defaults to offline mode. There is an option under the "File" menu to connect to data streaming in from the telescope. The code base for online and offline data analysis is identical, only offline users have access to far greater functionality.

5.6. Astronomer's Integrated Desktop

The Astronomer's Integrated Desktop (astrid) is a single, unified workspace that incorporates, but does not replace, the suite of applications that can be used with the GBT. The goal of astrid is to provide users with easy, managed access to the applications that they need for a successful experience with the GBT (see Figure 5).

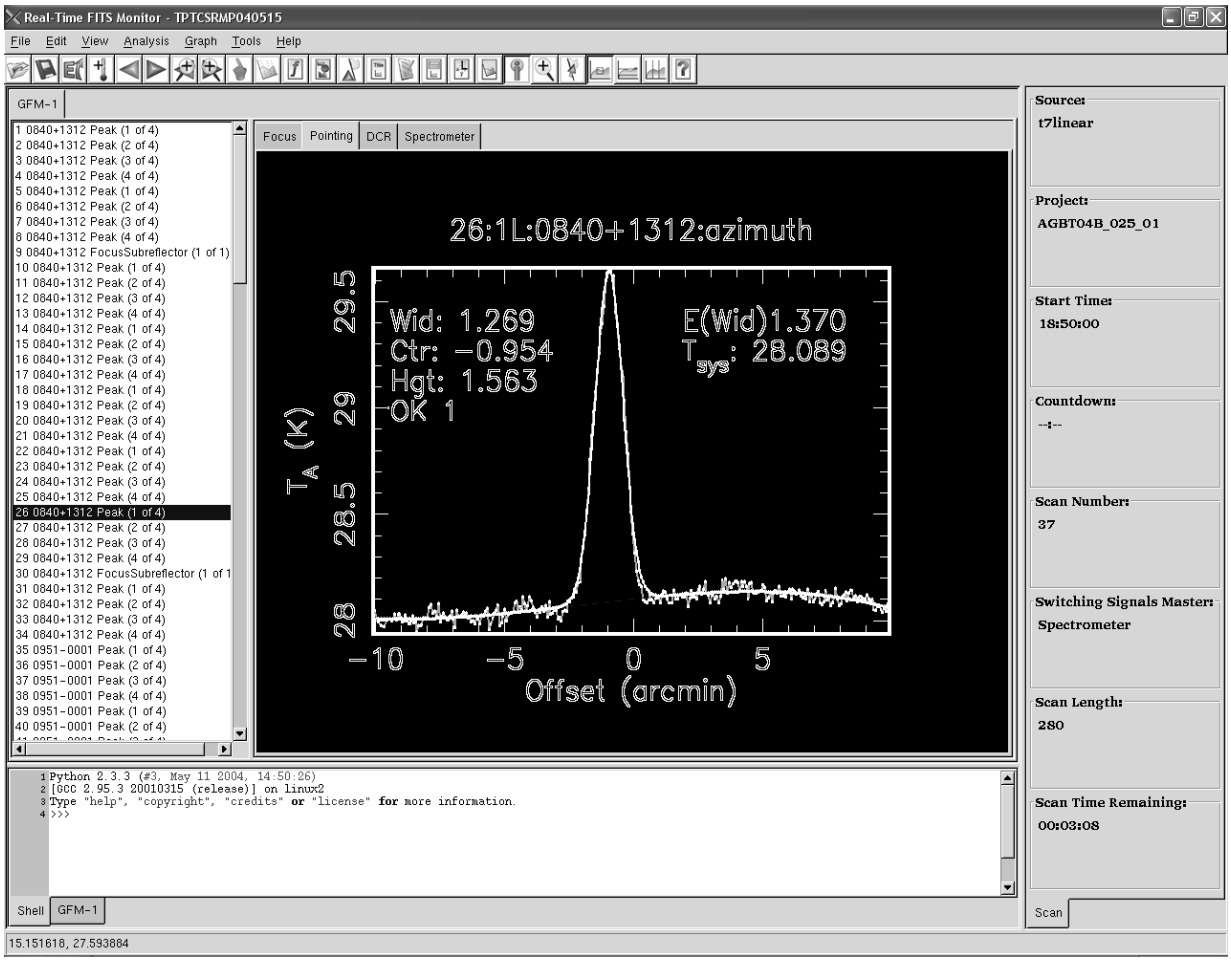

Figure 5: Screen shot of Astrid, showing two contained application components: GFM and a status panel. GFM is shown with four plugins, which are themselves application components: Focus, Pointing, DCR and Spectrometer.

The current prototype of astrid can be used to manage multiple instances of GFM, the dynamic corrections monitor, DEAP, status screens, and to access online help and other documentation. This is an application manager, which means that its sole purpose is to manage on-screen real estate, and all of the underlying science is taken care of by the component applications.

6. CONCLUSIONS

The preparation of artifacts that collectively make up an enterprise architecture has been a valuable experience, and has directly contributed to greater programmer productivity and a more solid technical vision for future work than was present in the past. Now, the user's experience of the GBT can be managed, rather than just the software applications themselves.

Most importantly, we are moving towards a paradigm in which all users can access the system at the level of abstraction that is most comfortable to them. New users can rely on the applications and application components, experts can gain more functionality by using the High-level APIs and Low-level APIs, and software engineers and commissioners can mix, match, and configure at will to arrive at scientific results faster. This will ultimately lead to quicker delivery of software utilities in support of new instrumentation.

The enterprise architecture has also helped GBT software align better with ALMA work. For example, the GBT Scheduling Block has identical entities and attributes as the ALMA Scheduling Block, with the possible exception of setup and configuration. Although these will be eventually archived in XML with the observer's calibrated data products, right now we simply aim to store them in plain text, and have chosen a format similar to the user preferences file.

Much more work is required, however, and the Green Bank EA will continue to grow and mature as new challenges are undertaken, such as the pursuit of an automated data reduction pipeline. Experiments are planned for the fall to see how some standard GBT observing modes could be pipelined, and if and how the architecture should be adjusted to accommodate such development within the upcoming two to three years.

7. ACKNOWLEDGEMENTS

This presentation reflects the work of several people. Ron Maddalena and Frank Ghigo provide scientific guidance as Project Scientists for the Data Handling Improvements and Ease of Use projects, respectively, which draw extensively from the concepts and principles outlined in this presentation. Eric Sessoms has served as the chief architect for many of the systems from which key aspects of the EA have emerged. Melinda Mello designed, developed and now manages the Configuration API and associated interactive tool. Ramon Creager has been directing the work for Grail, the standard telescope control system interface that has recently been developed. Mark Clark and Amy Shelton have been working on the Observing API, and under the technical guidance of Eric Sessoms, have also contributed to GFM. Paul Marganian crafted the original dynamic corrections application component. David Fleming, Jim Braatz and Bob Garwood have been instrumental in making GBT data accessible to multiple analysis and reduction packages.

Ron Maddalena is the designer and builder of CLEO, which has been a highly successful interface to control system parameters and processes. The Interferometry Software Division (ISD) originally built and supported the aips++ products IARDS and DISH prior to June 2003, which are still in use at the GBT today. We also wish to thank Jim Braatz and Bob Garwood who are now responsible for much of the coding and current support for these two products.

The National Radio Astronomy Observatory is a facility of the National Science Foundation, operated under cooperative agreement by Associated Universities, Inc.

REFERENCES

1. P.R. Jewell and R.M. Prestage, "The Green Bank Telescope", in Proc. SPIE 5489 (this conference), 2004.
2. http://www.cots.state.va.us/ea/eaovervw/tsld004.htm
3. See various technical papers and discussions at http://www.ambysoft.com/
4. "Updating the Clinger-Cohen Competencies for Enterprise Architecture." Retrieved on May 12, 2004 from http://www.cio.gov/documents/FINAL_White_Paper_on_EA_v62.doc.
5. Tody, D. NRAO Observatory Model v0.1 (internal draft document). Available to NRAO employees at https://wiki.nrao.edu/pub/Software/E2EMeetingMar312004/ObservatoryModel_v0.pdf
6. Mische, M. A., 2004. "Defining Systems Integration." *Enterprise Systems Integration*, J. Wyzalek, ed. Auerbach Best Practices Series, Boca Raton FL, 2000. pp. 3-10.
7. Enterprise Architecture Community, http://www.eacommunity.com/articles/art10.asp
8. http://en.wikipedia.org/wiki/API
9. Farris, A., 2004: The NRAO e2e Project Model (in progress).
10. http://wiki.gb.nrao.edu/bin/view/Data/ConfigurationCases
11. The NRAO e2e Oversight & Advisory Committee was established to coordinate the software activities of sites and projects throughout NRAO to achieve end-to-end deliverables, for example, pipelines. "The product of the Committee will be a proposed high level system design and roadmap for the data systems at NRAO, including both an NRAO-wide view and a view for how each telescope project fits into the overall system." From http://wiki.nrao.edu/bin/view/Software/E2ECharge (NRAO internal only).
12. Radziwill, N. M. and A. L. Shelton, 2004: TWiki as a Platform for Collaborative Software Development Management. Proc. SPIE 5496, Glasgow Scotland, June 2004.
13. Maddalena, R., 2002: The user interfaces for the NRAO-Green Bank Telescope. Proc. SPIE 4848, Waikoloa HI, August 2002.
14. http://wiki.gb.nrao.edu/bin/view/Data/FitsQueryLanguage
15. http://wiki.gb.nrao.edu/bin/view/Data/GbtInterface
16. O'Neil, K., N. Radziwill, and R. Maddalena, 2003: Increasing the Accessibility of Green Bank Telescope Data, ADASS XII, Strasbourg FR, October 2003.
17. http://deap.sourceforge.net