A Rational Plan for AIPS++

Chris Flatters

12 March, 1992

1 Introduction: The Failure of the Green Bank Meeting

In February 1992 a workshop was held in Green Bank that had the objective of analysing the requirements for processing radio astronomical data in AIPS, including calibration and imaging. The final report of this workshop is an attempt at an object-oriented analysis of this problem domain along the lines suggested by Coad and Yourdon in [?]. As such it is a dismal failure.

Requirements analysis has several goals (see, for example, [?]).

- It enables the system engineer to specify software function and performance, to indicate the software's interface with other systems elements and to establish design constraints.
- It enables the software engineer to build models of the process, data and behavioural domains that will be treated by the software.
- It provides the software designer with a representation of the information and functions dealt with by the system that can be translated into a design.

The Green Bank report does none of the above. The functions provided by the system are left vague as is the nature of the data that the system can deal with.

Although the authors admit that the analysis is incomplete in many areas[1] they claim suggest that it could be used as a framework for further analysis and design. Examination of their model shows that it is not adequate even for this purpose.

Despite the fact that the operations that can be carried out on data are strongly conditioned by the form of that data the authors of the Green Bank model have gratuitously separated operations from data. This has led to an overly complex model. Consider the steps that must be taken to obtain the calibrated value of an interferometer measurement. We need to make three requests to three different objects.

[1] This is an understatement: the analysis is incomplete in all areas

1. Send a request to the *Measurement*[2] to obtain the *Telescope* object associated with it.

2. Send a request to the *Telescope* object to find the current *TelescopeModel*.

3. Send a request to the current *TelescopeModel*, with the original *Measurement* as a parameter, to return the calibrated value.

Most experienced object-oriented programmers would have provided the class that represents interferometer measurements with a service that returns a calibrated value. In this case we only need to send one request to a single object. The interferometer measurement will need to request services from other objects in order to provide the calibrated value but this complexity is hidden from us[3]. Furthermore, this model ensures that the service which is used to calibrate interferometer data is always appropriate to that kind of data (since the service is associated with that class); the Green Bank model appears to provide no mechanisms to prevent attempts to calibrate interferometer data using a telescope model designed for single-dish, total-power measurements.

It is also worth examining the methods by which calibration models are updated in the Green Bank model: this is an outstanding example of confused thinking. The *SolveForModel* service of *TelescopeModel* must operate on a collection of measurements but *TelescopeModel* objects are associated with single measurements and not with collections of measurements: how then do we determine which *TelescopeModel* should update the calibration model for a given set of data? Perhaps the measurements supplied to *SolveForModel* should all be associated with the same *Telescope*. In this case the model needs to ensure that this is the case: the Green Bank model appears to contain no mechanisms to ensure this.

The Green Bank report is so muddled that it serves no useful purpose. It should be discarded in favour of a fresh start.

2 The Plan: A Fresh Start

Object-oriented analysis aims to produce a model of the problem domain in terms of classes of objects with the intent that these classes will form the basis for an object-oriented software design. This requires rigour and precision. Both of these qualities are notably lacking from the Green Bank document.

It is clear that we need a fresh analysis of the problem areas involving the handling of astronomical data that will provide a firm foundation for AIPS++. Without this fresh start we will find ourselves with a system that is overly complex. This may not show up in simple prototypes but will exact its price later. The longer the fresh start is delayed the worse matters will become: the

[2]The term *Yeg* is a poor joke that has suffered through overwork.

[3]The hiding of complexity is a major goal of object-oriented programming.

more effort that is invested in building a system based on the flawed Green Bank model the harder it will be to throw that model away.

In order to avoid repeating the failure of the Green Bank meeting, I suggest that the following conditions be met.

- It is difficult to produce a rigorous model of a large ill-defined system. We should procede under the working assumption that AIPS++ will be a set of collaborating programs, as was the original AIPS. This is sensible since we require a system that is extensible and such a system is easily extended by adding new programs to it. Each program will carry out a well defined operation on a particular kind of data, either modifying that data or producing some result. Operations should be identified and analysed separately. Since the operations are well-defined it is obviously possible to carry out a full analysis. After a number of operations have been analysed in detail it will be possible to identify recurring class structures. These should be extracted and developed into application frameworks[4] for AIPS++.

- The analysis team or teams should be small: no more than 3 people should be in any one group. Larger groups, such as that that produced the Green Bank report, are not conducive to rigorous analysis. Given the well-determined nature of many operations one-person teams and may be quite productive.

- Analysis teams should not work in isolation. There is a constant need to consult with working astronomers to clarify what they require. Problems in the analysis should be solved by research rather than guesswork.

- Analysis teams should exercise good taste in developing classes in the model that results from the analysis. Complexity should be hidden behind class walls wherever possible and not exposed to public view.

The first suggestion leads us into the "fountain" model of object-oriented software development. This has been demonstrated to work in several programming projects and will lead to early development of a working, if incomplete, system. It is also more likely to uncover flaws in our thinking at an early than the construction of prototypes since we are dealing with real data reduction problems from the outset rather than playing with simplified models.

References

[1] Coad, Peter and Yourdon, Edward. *Object-Oriented Analysis, Second Edition.* Prentice Hall, New Jersey, 1991.

[4] An application framework is a specialised type of class library that forms a skeleton form which programs may be derived by adding additional classes

[2] Pressman, Roger S., *Software Engineering: A Practitioner's Approach.* McGraw-Hill, New York, 1992.