Astronomical Data Analysis Software and Systems XII
ASP Conference Series, Vol. 295, 2003
H. E. Payne, R. I. Jedrzejewski, and R. N. Hook, eds.

Refactoring DIRT

N. S. Amarnath, Marc W. Pound and Mark G. Wolfire

Astronomy Department, University of Maryland, College Park, MD 20742, Email: amar@astro.umd.edu

Abstract. The Dust InfraRed ToolBox (DIRT - a part of the Web Infrared ToolShed, or WITS, located at http://dustem.astro.umd.edu) is a Java applet for modeling astrophysical processes in circumstellar shells around young and evolved stars. DIRT has been used by the astrophysics community for about 4 years. DIRT uses results from a number of numerical models of astrophysical processes, and has an AWT based user interface. DIRT has been refactored to decouple data representation from plotting and curve fitting. This makes it easier to add new kinds of astrophysical models, use the plotter in other applications, migrate the user interface to Swing components, and modify the user interface to add functionality (for example, SIRTF tools).

DIRT is now an extension of two generic libraries, one of which manages data representation and caching, and the second of which manages plotting and curve fitting. This project is an example of refactoring with no impact on user interface, so the existing user community was not affected.

1. Rationale for Refactoring

DIRT is a powerful tool for searching pre-calculated models to fit observed data. DIRT provides access to over 500,000 models, parameterized over a large set of physical parameters and dust types. DIRT has been used by the astronomy community since 1999. For a more complete description of DIRT and its user interface, refer to Pound et al. (2000).

DIRT had a user interface that was closely coupled to the data representation, both on disk and in memory. Essentially, DIRT did not use a Model-View-Controller (MVC) pattern.

At the present time, DIRT is being extended to

- support missions such as SIRTF by providing instrument based models,
- support models for disk shaped dust regions around young and evolved stars.

Such extensions are less effort to implement if data storage is decoupled from data representation in memory, and the user interface makes minimal assumptions about data representation and manipulation.

© Copyright 2003 Astronomical Society of the Pacific. All rights reserved.

2. DIRT - Application Structure & Modifications

DIRT is an application that is made up of

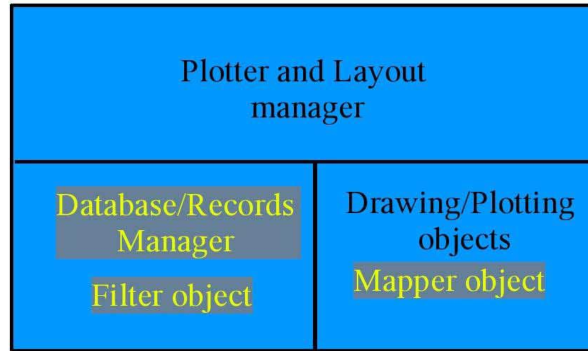
1. model files, containing pre-calculated data from parameterized models,
2. a directory tree containing the model files, where each level in the tree represents a specific parameter, and a value of that parameter is represented by a node at that level,
3. Perl scripts for retrieving model data from the disk,
4. a Java applet using the Perl script that provides an AWT-based interface for plotting and fitting.

The model data files store data as ASCII columns of numbers with human readable column headers. A model may have multiple data files, each with its own columns and rows.

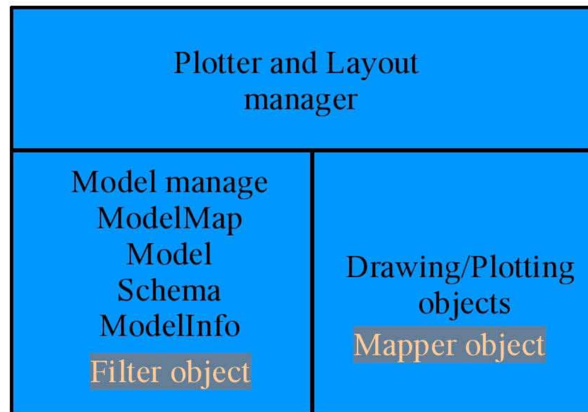
Before refactoring, model data was represented within the user interface using Database and Record objects. The Database object (a Java class), is a set of records, where each record is a row in the model data file represented by the Database object. Each plot window was associated with an array of database objects, representing the model data of interest for a set of plots. A column within a Database object was accessed by name, and these column names were explicitly used within the user interface. Database objects were associated with specific files by name. Datasets for plotting were selected using an class Filter object, and the chosen datasets were mapped to plot axes using a Mapper. Adding model data files, adding new types of models, and modifying the user interface required significant development effort with this design.

In order to simplify future upgrades, we decided to decouple the user interface and data representations and move to a more MVC compliant design for the application. However, any redesign (refactoring) had to be accomplished without affecting DIRT's user community. This implied that anything directly visible to the user had to remain unchanged, and behavioral changes to the user interface had to be minimal. Since our resources were limited, we also had to pick redesign options that would minimize the effort required to make the necessary code changes. These constraints meant that we could only change the representation of model data and the interface to data storage. To simplify the process of change, we decided to change only the Java code, and leave the Perl scripts mostly intact.

We decoupled data storage representations from the user interface by replacing the Database and Record classes by ModelMap, Model and ModelData classes. ModelMap is an ordered collection of Model objects. Each Model object is a collection of ModelData objects, which represent the data in one model file. Data in a ModelData object are read from the disk file and parsed using a description of the model data file contained in a schema, or Schema object. Information about the nature of the model, for example, the number of model parameters and the structure of the directory tree, are managed by an object of type ModelInfo. In this design, a ModelMap object now contains all models of interest to the user. Data from all data files associated with a specific model are managed by the associated Model object. Filter was slightly modified to filter datasets out of ModelMap objects, and Mapper changed to manage mapping and scaling of data to plot axes, which was formerly handled by Database.



Unreconstructed DIRT – Shows structure of DIRT prior to refactoring.



Reconstructed DIRT – Shows structure of DIRT after refactoring.

Figure 1. A block diagram showing changes to DIRT.

Figure 1 provides a picture of the modifications to DIRT - grayed out objects represent classes that were modified, or eliminated.

3. The New, Refactored DIRT

As a result of our refactoring, we can now add new model types and instrument based model data without changing any code. The process of designing the addition of SIRTF based model data was considerably accelerated - in fact, apart from the additional model data files, it involved the addition of just three text files.

In addition, source code was greatly simplified, making it more readable and comprehensible. For example, referring to Figure 2, which describes a routine

for displaying the details of a point on a plot, the refactored code on the right is more understandable.

Figure 2. Sample difference in source code between the original version, on the left, and the refactored version on the right.

The Java applet in DIRT is now composed of 3 packages, one for plotting and fitting numeric data, a second for managing data in memory, and a set of DIRT-specific classes.

Acknowledgments. Refactoring of DIRT was supported by NASA Applied Information Systems Research Program (AISRP) grant NAG 5-10751.

References

Pound, M. W., Wolfire, M., Mundy, L., Teuben, P. J., Lord, S., 2000, in ASP Conf. Ser., Vol. 216, Astronomical Data Analysis Software and Systems IX, ed. N. Manset, C. Veillet, & D. Crabtree (San Francisco: ASP), 628