Software Requirements Specification

for

SSC Portal for Offline Reprocessing

Versions 2.0-5.0 draft

Prepared by Ted Hesselroth

SIRTF Science Center

6-24-02

Table of Contents

Index of Figures

Revision History

Name	Date	Reason For Changes	Doc Version
Ted Hesselroth	4-23-02	Original Version Draft	0.1
Ted Hesselroth	6-18-02	Released Version for SPOR 1.0	1.0
Ted Hesselroth	6-20-02	Draft of SPOR Version 2.0	1.5
Ted Hesselroth	6-24-02	Draft of SPOR Versions 3.0, 4.0, 5.0	1.7

1. Introduction

1.1 Purpose

SPOR is a tool for running SIRTF data reduction pipelines. The main purpose of SPOR is to provide a user interface to downlink software which is normally invoked by the automated processing system. With SPOR, a user can select the data to be processed, specify processing parameters, and invoke the pipelines. This document describes the software requirements for SPOR.

1.2 Document Conventions

Source code will be listed in courier typeface, with emphasized sections in bold type.

1.3 Intended Audience and Reading Suggestions

The intent of this document is to apprise prospective users of the proposed features of SPOR for the purpose of solicitation of comments and suggestions. The current design is a product of preliminary discussions of use cases and desired features. This document covers only the current release; further functionality can be added to future releases.

1.4 Product Scope

This document describes the Java user interface to the SIRTF Science Center data reduction pipelines.

1.5 References

Documentation for downlink can be found in the SSC Downlink Webpage. An overview of the downlink automated processing system is given in Automate Pipeline Executive for SIRTF (APES). Information on various specific pipeline modules can be found in the Subsystem Design Specification documents, Downlink SDS path.

2. Overall Description

2.1 Product Perspective

There is presently a software subsystem at the SSC for automatic processing SIRTF data. The subsystem consists of three main software elements: binaries compiled from C, C++, or Fortran code for performing numerical operations on the data, wrapper scripts written in perl for executing the binaries and making supporting database calls, and the executive infrastructure for invoking a sequence of wrapper scripts comprising a data reduction pipeline for a given input data file. In automated processing, the infrastructure commences the pipelines upon receipt of data by the ingest process, and the only interaction by the pipeline operator is to view the processing results.

There may be times when it is desired to run pipelines “manually”, outside of the automatic processing system. Products so obtained would not be deliverable outside the SSC, but would be created in support of the standard pipelines. Various use cases have been identified:

1. All pipeline products will pass through an inspection, upon which a status is set. One status value designates that reprocessing is to be done. For some data it may be possible to specify the reprocessing parameters by a rule, in which case reprocessing can be performed via the automated processing system without further intervention. In other cases, the data may be referred to an expert. If it is not evident what reprocessing parameters are suitable, the expert may wish to reprocess parts of the data offline to test various scenarios before making a recommendation.

2. Some data may be known to require a specific pipeline not supported by the automatic processing system. For example, if an artifact appears that is not removed by the current downlink version, it may be necessary to remove it offline in order to accurately evaluate the performance of the instrument.

3. Proposed software modifications for future downlink versions need to be tested, preferably on real data. Similarly, testing of delivered pipelines capabilities is necessary. The SPOR tool will reduce the effort required to test pipelines.

4. In many cases, tuning of pipeline parameters before commissioning of the automated processing system is desired.

2.2 Product Functions

There are four main functions that SPOR must support:

1. SELECTION: The user shall be able to choose a set of SIRTF data to be processed.
2. QUE: A que of selected data must be maintained.
3. PARAMETERS: The user shall be able to specify the set of parameters that control the data reduction algorithms.
4. RUN: The pipelines shall be started from within SPOR.

A user interface shall be provided for each function for control and display of status. A fifth function, INSPECTION, which shall allow inspection of the data at various stages of processing, will be available in a later version. Details of the SPOR functions are provided in Section 4.

2.3 User Classes and Characteristics

The main class of user for this program is the Instrument Scientist. It is assumed that the user is very familiar with the form of the raw data and the operation of the pipelines. The user should know how to find the raw data on the file system, what pipeline needs to be run for it, what parameters should be set for it, and where the output products should be placed.

SPOR will be used frequently during IOC, to determine pipeline parameters and to estimate the efficacy of pipeline stages. It will be used often during the initial survey, GTO, and legacy observations, until full confidence in the parameters and stages of the automated pipelines is reached. SPOR will continue to be in use for custom processing of anomalous raw data and for pipeline verification of post-IOC deliveries.

SPOR may be used by GTO, legacy teams, and general observers if the SSC decides to make it available for use in their own possible reprocessing. SPOR would then need to support generic pipelines, a capability which is planned for future versions.

2.4 Operating Environment

SPOR is a Java application and the stated requirement for the JVM shall be Java 1.4. Presently there is Solaris, Linux, and Windows support for Java 1.4. The runtime java virtual machine will be bundled with SPOR if necessary. (N.B. This version of SPOR will actually run in Java 1.2.2 or higher). The pipelines require perl 5.0 or higher. Any pipeline requirements, such as the existence of libraries, database access, and filesystem access, are inherited by SPOR for the pipelines which are to be run. In addition, any environment variables expected by the pipelines must be pre-set in the environment in which SPOR is started. The environment is passed along to the wrappers and modules that comprise the pipelines. SPOR interfaces with the pipelines by making a system call which invokes the same perl function, run_pipeline.pl, which is invoked by the automated processing system to run the pipelines. Details are given in section 4.4.

The reprocessing is to be done on non-Operations machines. For SSC use, it is assumed that the data have already been processed at least once by the automated processing system. Therefore, the raw data files, calibration files, control data files exist within a directory structure on the Operations file system. In addition, pipeline database entries and FITS header keywords will have been written.

In the likely temporal window in which reprocessing shall be desired, the necessary files can be assumed to reside in the sandbox area of the Operations file system. A utility will be provided within SPOR to copy the needed files to a local directory structure of the user, where the reprocessing will occur.

Thus the user will need read permission in the sandbox and write permission in a local area from within the same application. In the current version of SPOR, the raw data is found by direct file system search. Future versions of SPOR may use the SSC database to locate the data, in which case read permission in the database will be needed.

System resource use by SPOR, namely memory and clock cycles, will be light, small in comparison to those used by the pipelines. Therefore SPOR does not specify additional hardware requirements. The Java 1.4 virtual machine requires a 166 MHz processor with 32 MB RAM on a

PC. For Sun workstations, the Java 1.4 requirement is that the operating system be Solaris 2.6 or above.

2.5 Design and Implementation Constraints

SPOR shall be written as a Java application, as opposed to a Java applet. Applets are limited by security constraints against doing such things as obtaining directory listings and making system calls. SPOR could run as an applet if special permissions were set on each host machine, or if a java server architecture were deployed. These steps are not deemed necessary at this time.

The SPOR source code shall be kept under configuration management, but SPOR deliveries will not follow the same schedule as pipeline deliveries.

2.6 User Documentation

There will be an offline SPOR manual, and floating tooltips will be employed in the application.

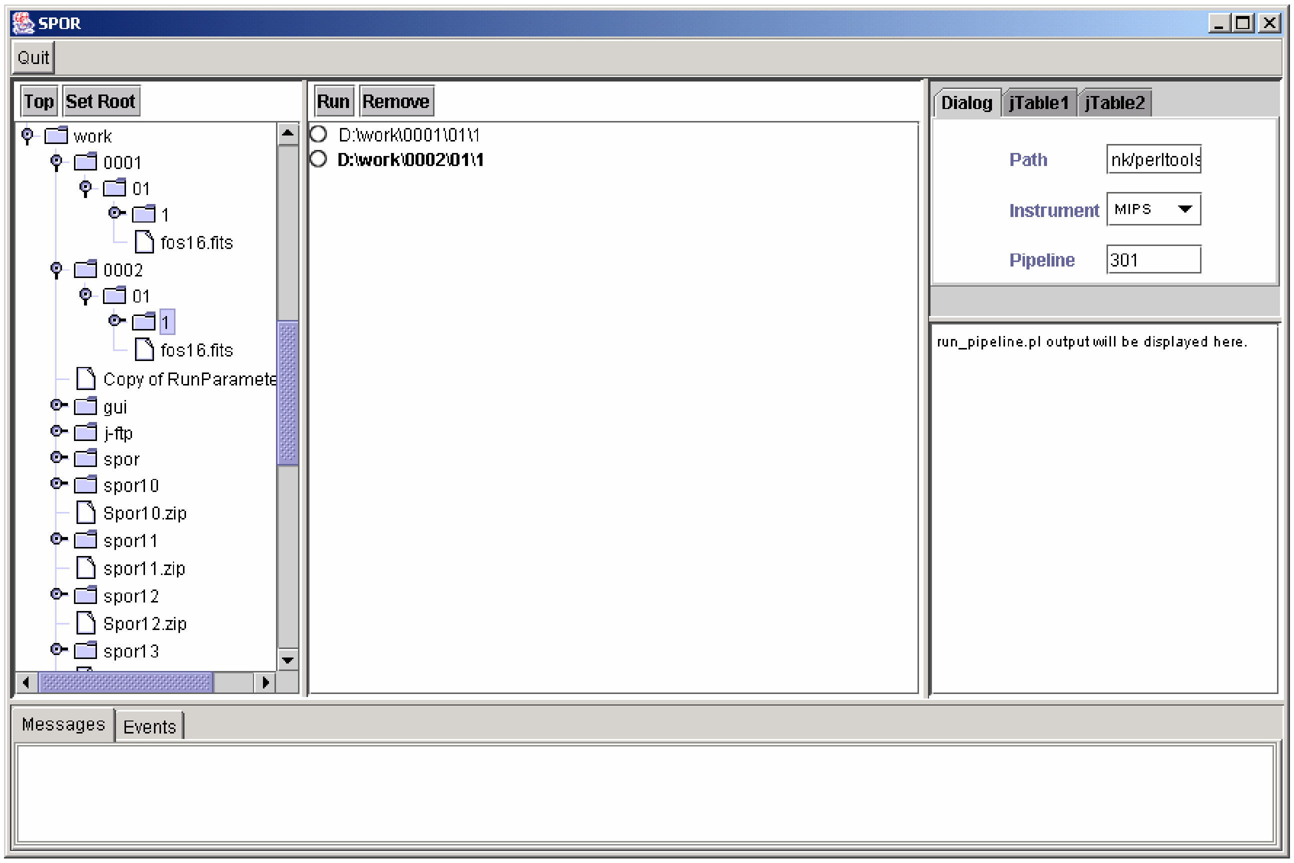
2.7 Assumptions and Dependencies

SPOR will need the support of the ISG regarding deployment of the software and of Operations regarding file system and database permissions granted to users. It is necessary that downlink Cognizant Engineers write the pipelines in such a way as to support offline reprocessing. In particular, writes to the sandbox or database will not be possible. It will be necessary to have options to turn off or redirect such writes during offline reprocessing.

3. External Interface Requirements

3.1 User Interfaces

Below is a screenshot of SPOR's user interface. Most of SPOR's functionality can be accessed via mouseclicks. There is a toolbar for the SPOR application itself, containing a Quit button (top). Each SPOR function has its own view in the form of a panel. In the main display window are seen, from left to right, the tree view of the filesystem for the Selection function, the list of files or directories selected in the Que function, the dialog panes of the Plan function (upper right), and the output of the pipelines from the Run function (lower right). Each function is described in detail in section 4. Finally, text panels for displaying SPOR messages and events are placed in a tabbed pane (bottom).



D:\work\0001\01\1
D:\work\0002\01\1

Path nk/perltools
Instrument MIPS
Pipeline 301

run_pipeline.pl output will be displayed here.

Figure 1. SPOR version 1.0 screenshot

3.2 Hardware Interfaces

There are no hardware interfaces specialized to SPOR. As described in section 2.7, access to the hard disks on which the raw data is stored is a requirement.

3.3 Software Interfaces

SPOR will run in a Java Virtual Machine and invoke perl functions through system calls. The current version will access the filesystem to read directory information and to save property files. Future versions will access the database in order to select and acquire data to be reprocessed. Inspection capabilities will also be added to future versions, and these may actually be interfaces to other software packages or services that provide the functionality.

3.4 Communications Interfaces

SPOR does not use communications interfaces such as ftp or http.

4. System Features

4.1 The Selection SPOR Function

4.1.1 Description

The first step in reprocessing is to choose what data to reprocess. It shall be assumed that the data reside in files on the filesystem, accessible to the user. It is assumed that at least one round of processing has been done on the data, and that the files exist in the directory structure used in automated processing, or a copy thereof.

With the current pipelines, data files are specified via the path. For DCE data, the filenames themselves are common, namely “fos16.fits”. Therefore the selection process consists of choosing the paths to the data. The current pipeline executive script, run_pipeline.pl, has the ability to recursively search for fos16.fits files under the path that is specified to it. So for the case of processing multiple DCEs, it is only necessary to specify a high-level directory under which all the fos16.fits files exist. The directory structure used in automated processing has been designed to accommodate this.

4.1.2 User Input Sequences

4.1.2.1 **Selecting from the tree view.** The filesystem is displayed in a tree view similar to that of Windows Explorer. The user can display a portion of the directory hierarchy by successively clicking on nodes of the tree. When the desired processing file or directory is seen, (V1.6 double-)clicking on its folder icon causes the item to be queued (see section 4.2 for the Que function). To avoid having to drill down the entire directory structure each time SPOR is used, a button is provided which allows the root of the tree view to be changed to the currently selected directory. To see the higher levels again, another button is provided to reset the tree view root to the top filesystem directory. V2.8 Holding down the control key while clicking on a folder or file icon allows multiple selection of items. The items can then be queued by dragging the selection to the Que panel, or by pressing the provided “Que” button. The items will by default belong to the same Que display group(see section 4.2.2.5).

V1.6 The filesystem can alternatively be displayed as a textual list of full paths by pressing the “Paths” button. One can return to the tree view by pressing the “Tree” button.

4.1.2.2 V1.6 **Selecting with wildcard expansion.** There is also a text box in the toolbar of the Selection panel, which displays the full path of the current position in the tree or path view. V1.7 The user can modify or extend the path via keyboard input, including the wildcard symbol (*). When the “Search” button is pressed, the display changes to a full-path listing of the wildcard expansion result. Double-clicking on an item in the list causes the file or directory to be queued.

4.1.2.3 V2.3 **Selecting by BCD name.** BCD products and some ancillary files will be renamed to include the data product ID in the filename before archiving. Furthermore, they may be delivered in a tar file with other BCDs from the same Astronomical Observation Request or Engineering Request. SPOR will provide the capability of unpacking and/or renaming such files to a form suitable for running the pipelines.

4.1.2.4 V3.1 **Selecting by MIPL name.** BCD products and some ancillary files may be renamed following the convention used to name the raw data, that is, a period-separated series of fields containing, Request Key, Exposure, etc. SPOR will provide the capability of renaming such files to a form suitable for running the pipelines.

4.1.2.5 V4.1 **Selecting by Database Query.** BCD products that have been archived may only be available by a call to a database. There will be a panel in the Selection function that will serve as a user interface to the database and allow queing of the data and its retrieval to the local filesystem, both at the DCE and ensemble level.

4.1.2.6 V4.2 **Selecting by Sky Coordinates.** The user will be able specify a range of coordinates on the sky and retrieve data of a selected type from that part of the sky.

4.1.2.7 V4.2 **Selecting by Sky Object.** The user will be able specify an object in the sky and retrieve data of a selected type from that object.

4.1.3 Functional Requirements

REQ-1: A tree view of the filesystem allowing the selection of any directory.
REQ-2: A button to set the root of the filesystem view.
REQ-3: A button to reset the root to the top filesystem directory.
V1.6 REQ-4: A path view of the filesystem allowing the selection of any directory.
V1.6 REQ-5: A button to set the view to the tree view.
V1.6 REQ-6: A button to set the view to the path view.
V1.6 REQ-7: An editable text box to display the current path.
V1.7 REQ-8: A button to cause wildcard expansion of the path.
V1.7 REQ-9: A path view of the wildcard expansion results.
V2.8 REQ-10: A button to queue multiple selections.

4.2 The Que SPOR Function

4.2.1 Description

Once the datasets for reprocessing have been selected, two further steps are needed: planning the runtime parameters of the reprocessing, and invoking the pipelines. These two functions are described in sections 4.3 and 4.4. The que function serves as the gateway to the remaining functions. It also interfaces with the Selection function in providing a view of the selected datasets. Thus the Que function is the central component of the user interface. When a queued item is selected, the Plan function is invoked for that dataset. The Run function can be subsequently invoked by a button press in the Que panel.

4.2.2 User Input Sequences

4.2.2.1 **Selecting a queued item.** The que of data to be reprocessed is represented by a list of the paths chosen in the Selection function. Clicking on an item in the list causes the panel of the Plan function to display a dialog box by which the runtime parameter of the dataset can be modified.

4.2.2.2 **Button actions.** A toolbar at the top of the Que panel contains a “Run” button. When this button is pressed, Que calls the Run function once for each item in the list. There is also a “Remove” button to delete items from the list, which V2.8 supports removal of multiple selections, and V1.9 a “Save” button, which saves the current que as a properties file, a “Save As” button to save the que under a new name, and a “Load” button to load a previously-saved que.

4.2.2.3 V2.1 **Creating an Ensemble.** An ensemble may be defined by holding the control key down while selecting multiple items from the que, then pressing the “Ensemble” button. A list of to be considered an ensemble is then created and stored internally to spor. The items are then combined one queued dataset and displayed in the Que view accordingly. The Plan view of the ensemble will then display the ensemble processing parameters.

4.2.2.4 V2.2 **Status Indicator Actions.** Next to each queued item is a status indicator, the color of which indicates the processing status of the item. Yellow indicates that the dataset has been queued. Orange indicates that the files needed to process the data cannot be found. Dark Yellow indicated that the dataset is now being processed. Green indicates that the processing has proceeded to completion for that dataset. Red indicates that processing for that dataset exited with an error message. White indicates that the dataset will not be run in the next invocation of the pipeline. Clicking on the status button has various effects depending on the color displayed. Clicking on a Yellow or Red status button causes the pipeline to start for that item. Clicking on an Orange status button has no effect. Clicking on a Dark Yellow status button causes the pipeline to be suspended for that item. Clicking on a Green status button has no effect. Clicking on a White status button causes the dataset’s runnable flag to be set to true; it will be processed if the Run button is pressed;

4.2.2.5 V3.3 **Truncated view option.** The display of the full path for each queued dataset may clutter the Que panel. The “View” button on the toolbar of the Que view panel, when pressed, causes the display of the dataset’s path to be shorted to display only the DCE number. Furthermore, datasets can be arbitrarily grouped by making a multiple selection using the control key, then pressing the “Display Group” button. The items selected will then be displayed as a group in the list. The members of the group can be displayed individually by a press of the “+” button next to the group display. An “Ungroup” button will be provided to cause the items to be displayed individually.

4.2.3 Functional Requirements

REQ-1: A displayed list of the paths of each dataset in the Que.
REQ-2: A “Run” button to invoke the Run function.
REQ-3: A “Remove” button to remove the currently selected item from the list.
V1.9 REQ-4: A “Save” button to save the current que.
V1.9 REQ-5: A “Save As” button to save the current que under a new name.
V1.9 REQ-6: A “Load” button to load a saved que.
V2.1 REQ-7: An “Ensemble” button to combine selected items into an ensemble.

V2.2 REQ-8:	A status indicator for each queued item and associated actions.
V3.3 REQ-7:	A "View" button to truncate the display of the datasets' paths.
V3.3 REQ-7:	A "Display Group" button to group the currently selected items.
V3.3 REQ-7:	A "+" button in the display to expand the display of the group.
V3.3 REQ-7:	An "Ungroup" button to ungroup the members of a display group.

4.3 The Plan SPOR Function

4.3.1 Description

The value of reprocessing lies in the ability to adjust the runtime parameters of the pipelines. There are four main categories of pipeline inputs that control the processing:

Calibration Files characterize the instruments and are used by the pipeline modules for removing artifacts from the data.

Control Data Files are text-based files read by pipeline modules for setting parameters and defining input and output filenames.

Environment Variables are read by wrapper scripts and modules, usually to obtain path information.

FITS headers are read to obtain information on the data that is being processed and IDs of files to be used in processing.

The Plan function provides for modification of the pipeline parameters.

4.3.2 User Input Sequences

The user clicks on an item in the list displayed in the Que function. The panels in the Plan function then display the runtime parameters for that dataset. A tabbed panel is used to switch between the various Plan panes. Changes to the parameters made through the Plan function are saved internally by SPOR and by default are not applied to any of the files in the datasets' working directory until Run time.

4.3.2.1 V1.2 The **Files** panel shows the directory structure at the path of the selected dataset. This panel is used to set up the contents of the pipeline working directory and its subdirectories. The user wants a certain set of cdf and cal files to exist in the path at Run time; these may or may not be present at Plan time. The panel contains a "Set Files" button on its toolbar. When the user presses the button, a dialog box appears which allows the selection of files. To allow renaming, there are fields for both the source and destination filenames. Through the dialog box, the user creates a list of files to be replaced or added. This information is maintained in the SPOR object representing the dataset, and the actual moving of files is done at the beginning of the Run process, though a move can be done at Plan time if the user presses the "Copy" button. Subdirectories that don't exist can be created by including them in the destination path of a file that is to be copied, or if an empty directory is to be created, by entering a blank for the source and the directory path for the destination. V4.4 A dialog box for setting up other directories, such as user cache areas, can be invoked by pressing the "Setup" button.

4.3.2.2 **Database query**. V2.4 When selecting files, the user has the option of either using the file chooser or the database to obtain their locations. When the “DB” button is pressed, a dialog box containing fields for specifying the file is displayed. When the “Find” button of the dialog box is pressed, the database returns the path to the file in a text box, and the fully-qualified path to the file is entered into the source field. If the file does not exist on the user’s filesystem, pressing the “Copy” button causes it to be copied to the working directory. Pressing the “Cache” button causes the file to be copied to the user’s cache area. The cache area is specified by an entry into a text box.

4.3.2.3 **Caltrans query**. V3.1 The caltrans module may be called during the Plan step. When the “Caltrans” button is pushed, a dialog box appears which collects the input parameters for caltrans. Pressing the “Copy” button causes the results of caltrans to be copied into the working directory. Pressing the “Cache” button causes the results of caltrans to be copied into the user’s cache area. The cache area is specified by an entry into a text box.

4.3.2.4 V1.4 The **Namelist** panel displays the contents of the namelist file or files, which must reside in the “cdf” subdirectory of the pipeline working directory, and end in “.nl”. The view is in an editable table format, and rows can be added or deleted by the user. This allows complete control of the namelists. The panel can cycle through multiple namelists via press of the “Next” button.

4.3.2.5 V1.4 The **Environment** panel allows the specification of environment variables, since the settings inherited by the Run function from the shell which invoked SPOR may not be those desired for pipeline processing. The variable definitions are shown in an editable table format. Java cannot read environment variables. It can read properties files, so SPOR will save the environment variables in the dataset’s properties file in the cdf directory whenever a change is made to the displayed table. The environment panel first searches the cdf directory for a previous properties file, otherwise a default set is displayed. An “Open” button is also provided, which when pressed opens a file chooser dialog through which an existing properties file can be loaded. V2.9 Each line of the table contains a button labeled with the elipses symbol, which, when pressed, invokes a filechooser. When the dialog is closed, the selection from the file chooser becomes the value of the environment variable.

4.3.2.6 V1.5 The **FITS header** panel displays the header of, by default, a file named “fos16.fits”, which is expected reside in the same directory that contains the working directory. An “Open” button is provided which invokes a file chooser dialog to select an alternative file. The view is in the form of an editable table. Pressing the “Insert” button opens a file chooser which allows the insertion of a text file into the FITS header at the current location. The text file must contain only legal FITS cards. Deletion of rows in provided for by the “Delete” button. Pressing the “Save” button causes the fits header to be updated to correspond to the edited view.

4.3.2.7 The **Command** panel contains the fully-qualified V1.3 name of the run_pipeline.pl perl script which invokes the pipelines. (see section 4.4), the instrument, the desired pipe number, and V1.3 a text box for modifying the command line arguments or entering additional ones. The instrument can be specified by a selection from a pulldown menu; the others by typing in the displayed text fields. V1.3 Any executable can be specified in place of run_pipeline.pl, and the contents of the arguments box will be applied to it. V4.2 In place of a command, a database function can be specified, in which case the database server will invoke the pipelines. . V4.3 Alternatively, an external package such as IDL can be specified in place of a command, in which case the pipelines will be run in the external package.

4.3.2.8 V1.3 The **Pipelines** panel displays definitions of the pipelines as determined from parsing the "plinexyz.cdf" file. By default the file is looked for in a predetermined path relative to the executable defined in the Command panel. If it is not found there, the pipeline definitions are read from the datasets' properties file, if it exists. The user can specify pipeline definitions to be read from a cdf or properties file by pressing the "Open" button and selected the file in the file chooser that then opens. Pipeline definitions are displayed in the form of an editable table, with checkboxes to turn stages on and off. Any edits are immediately saved in the properties file. The edited pipeline definitions can be saved as a cdf file by pressing the "Save" button and using the file chooser to specify a file name. V4.3 A pipeline stage can be specified as a database object by entry of the corresponding database serial ID in place of the wrapper script filename. V4.3 A pipeline stage can be specified directly in terms of an algorithm, provided syntactical rules are followed.

4.3.2.9 V1.3 The **Properties** panel allows viewing and editing of the datasets properties file. All the above parameters, namely the listing of files in the working directory at Run time and their sources, the contents of the namelists, the environment variable settings, and the FITS header contents, are saved by SPOR in a properties file in the cdf directory. If a properties file exists when the Plan function is invoked, the default is to derive the properties from the other files in the working directory and its subdirectories, not the properties file. In the Properties panel, the "Open" button allows the use of an existing properties file. The properties file can be edited directly in this panel, and is saved whenever a change is made to the contents of its view. The "Insert" button allows the insertion of a text file, which must be in the format of a properties file, at the current location in the view. As mentioned previously, by default the properties are not applied to the files in the working directory until Run time. The "Apply" button is provided to cause propagation of the properties to the contents of the files in the working directory. Considerable editing may be done on the pipeline parameters for a given dataset, and it may be desirable to propagate its properties to other datasets. When the "control" key is pressed, multiple selections can be made in the Que panel without changing the contents of the Plan panels. Then, by pressing the "Export" button in the Properties panel, the current properties are duplicated to all items selected in the que.

Planning is done for all the items in the que before the Que "Run" button is pressed. V2.2 The Plan function checks for the existence of the necessary files before Run time; if not all the files are available, a status is set for the dataset to indicate it, and the status is displayed in the Que list, as a Orange.

4.3.2.10 V1.2 The **Ensemble** panel displays the ensemble list of files used for ensemble processing. This list is displayed as an editable text field. "Open" and "Save" buttons are provided for loading a previously-prepared list and for saving the contents of the current view of the list to a file in the working directory. V3.1 Pressing the "DB" button causes a dialog box to appear which allows querying the database for the list of ensemble files. Upon entering information in the fields of the dialog box and pressing the "Find" button, the view displays the list as returned from the database.

4.3.3 Functional Requirements

A dialog box containing the following editable items:

V1.2 REQ-1: A "Files" panel providing a view of the files in the working directory.
V1.2 REQ-2: A "Set Files" button in the panel of REQ-1.
V1.2 REQ-3: A "Modify Files" dialog box for selecting initial pipeline files.

V1.2 REQ-5: A “Copy” button in the dialog box for starting the file copy.
V2.4 REQ-6: A “DB” button to invoke a database lookup dialog box.
V2.4 REQ-7: Fields for specifying the file to be found by the database.
V2.4 REQ-8: A “Find” button to initiate the database query.
V2.4 REQ-9: A “Copy” button to copy the file to the working directory.
V2.4 REQ-10: A “Cache” to copy the file to the user’s cache.
V2.4 REQ-11: A text box to specify the path to the user’s cache.
V3.1 REQ-12: A “Caltrans” button to invoke a caltrans dialog box.
V3.1 REQ-13: Fields for specifying the file(s) to be found by caltrans.
V3.1 REQ-14: A “Find” button to initiate the caltrans query.
V3.1 REQ-15: A “Copy” button to copy the file(s) to the working directory.
V3.1 REQ-17: A “Cache” to copy the file(s) to the user’s cache.
V3.1 REQ-18: A text box to specify the path to the user’s cache.
V1.4 REQ-19: A “Namelist” panel for displaying/editing the pipeline namelist(s).
V1.4 REQ-20: An “Open” button in the panel for choosing a namelist.
V1.4 REQ-21: A “Next” to display the next namelist.
V1.4 REQ-22: An “Environment” panel to display/edit runtime env. variables.
V1.4 REQ-23: A “Open” button to open a properties file for environment variables.
V2.9 REQ-24: A elipses button for each line of the environment variables table.
V2.9 REQ-25: A file chooser dialog to be invoked by the elipses button.
V1.5 REQ-26: A “FITS Header” panel to display the header of fos16.fits.
V1.5 REQ-27: An “Open” button to select a file for header display in the panel.
V1.5 REQ-28: An “Insert” button to insert rows contained in an external text file.
V1.5 REQ-29: A “Delete” button to delete rows.
V1.5 REQ-30: A “Save” button to update the header.
REQ-31: A “Command” panel for construction of the pipeline Run command.
V1.3 REQ-32: A text box to set the command.
REQ-33: A drop-down menu to set the instrument.
REQ-34: A text box to set the number of the pipeline to be run.
V1.3 REQ-35: A text box for setting additional command line arguments.
V1.3 REQ-36: A “Pipelines” panel to display/edit the contents of plinexyz.cdf.
V1.3 REQ-37: An “Open” button to select the plinexyz.cdf file.
V1.3 REQ-38: A “Save” button to save the plinexyz.cdf file.
V1.3 REQ-39: A “Properties” panel to display all the SPOR properties of the dataset
V1.3 REQ-40: An “Open” button to allow the selection of a properties file.
V1.3 REQ-41: An “Insert” button to insert properties.
V1.3 REQ-42: An “Apply” button to cause modification of the pipeline files to match the dataset’s properties.
V1.3 REQ-43: An “Export” button to copy the dataset’s properties to all items currently selected in the que.
V1.2 REQ-44: An “Ensemble” panel displaying the list of files in the ensemble.
V1.2 REQ-45: An “Open” button to select an ensemble list file.
V1.2 REQ-46: A “Save” button to update or create the ensemble list file.
V3.1 REQ-47: A “DB” button to invoke a database ensemble lookup dialog box.
V3.1 REQ-48: Fields for specifying the ensemble list to be found by the database.
V3.1 REQ-49: A “Find” button to initiate the database query.
V4.4 REQ-50: A “Setup” button in the Files panel to start a dialog box for creating and populating non-dataset directories needed by the pipelines.

4.4 The Run SPOR Function

4.4.1 Description

The pipelines are invoked by a system call to the run_pipeline.pl perl script (by default, see section 4.3.2.5 for other options), which is the same program that is invoked in the automated pipelines. The run_pipeline.pl script is called for each item in the que with the path and command line arguments that were set in the Plan function, and the input path chosen in the Select function. It is useful to know the progress of the pipelines, therefore the standard output of run_pipeline.pl is displayed in a text window in the Run panel. Pipeline invocation is run in a thread to prevent Run from blocking the rest of SPOR while the pipelines finish.

4.4.2 User Input Sequences

4.4.2.1 **The Run Button**. The processing parameters for each dataset should have already been set up in the Plan function. The user then presses the "Run" button in the toolbar of the Que function, and processing begins, as seen in the standard output shown in the text window. V1.9 The standard output can be saved to a file by a press of the "Save" button, which opens a file chooser dialog to set the name of the file. V2.2 Since the run_pipeline.pl invocation runs in its own thread, other SPOR actions during processing are possible, namely adding new datasets to the que and modifying their run parameters with the Plan function. If the "Run button is pressed again, the new datasets are processed in another thread. Up to three pipeline threads are allowed. Each will display its standard output to its own text window; a tabbed panel is used to select which thread to view.

4.4.3 Functional Requirements

REQ-1: A function to implement a system call to run_pipeline.pl.
REQ-2: A text window to display standard output from run_pipeline.pl.
V1.9 REQ-3: A "Save" to allow the contents to be saved to a file.
V2.2 REQ-4: A tabbed panel to select among threads' output to view.

4.5 The Inspection SPOR Function

4.5.1 Description

The user may wish to inspect the data at various stages of processing, so SPOR provides capabilities for viewing the data. The Inspection function will contain various panels to provide multiple inspection options.

4.5.2 User Input Sequences

4.5.2.1 V3.4 The Inspection function's view consists of a tabbed panel, so that each inspection option has it's own panel.

4.5.2.2 V3.4 **View Selection Context Menu**. There is a universal menu of view selections for SPOR, which is displayed when the user right-clicks on a file icon, whether it belong to the Selection, Que, or Plan/File view. When the user makes a selection from the menu, the corresponding view of the file is shown in the Inspection Panel. Switching to another view of the same file is accomplished by selecting the corresponding tab in the Inspection panel.

4.5.2.3 V3.4 The **Text** panel allows the viewing of text-based files. The contents of the file are displayed in a scrolling panel.

4.5.2.4 V3.4 The **Image** panel displays a plane of the FITS file as an image. The plane is selectable via a slider bar with associated text box. Pressing the "Map Color" button invokes a dialog box by which a mapping of pixel value to display color space can be set.

4.5.2.5 V3.4 The **Header** panel displays the header of the FITS file in a scrolling text box.

4.5.2.6 V3.5 The **Mask** panel is used to display mask files. A bit in the mask can be specified via the slider bar with associated text box. The display shows black at pixels where the bit is not set, and white where it is set. Alternatively, a set of bits can be displayed by assignment of a color to each bit. When the "Assign Colors" button is pressed, a dialog is displayed, which displays a list of the bits, with a color selection slider bar for each one. A button to invoke the Color Chooser dialog is also provided for each bit.

4.5.2.7 V3.6 The **Cuts** panel displays a chart of a cut through the data in the file, if it is in the FITS format. The type of cut, whether by row, column, or depth, is selected via a set of radio buttons, and the cut location is selected via a mouse click on the desired column, row, or pixel.

4.5.2.8 V3.7 The **Pointing** panel reads pointing information from the FITS header and displays it in a chart.

4.5.2.9 V3.7 The **History** panel gathers processing history from filenames, fits headers, and log files, and displays it in tabular form.

4.5.2.10 V4.2 The **DB Info** panel provides read access to the database for viewing of previously entered data.

4.5.2.11 V4.5 The **Interactive** panel provides controls for processing the data as well as viewing it, and automatically updates the view in response to the processing.

4.5.2.12 V5.2 The **External** panel provides an interface to external programs, such as IDL, that can deliver inspection and processing capabilities.

4.5.3 Functional Requirements

V3.4 REQ-1: A tabbed panel to contain Inspection views.
V3.4 REQ-2: A context menu for every SPOR panel that displays a file icon.
V3.4 REQ-3: A "Text" panel for viewing ascii text files.
V3.4 REQ-3: An "Image" panel for viewing FITS images.
V3.4 REQ-3: A dialog box for choosing the color mapping of image values.
V3.4 REQ-3: A "Map Color" button to invoke the color mapping dialog.
V3.4 REQ-3: A "Header" panel to display the FITS header.
V3.5 REQ-3: A "Mask" panel to display integer-valued FITS files.
V3.5 REQ-3: A slider bar to select the bit to display.
V3.5 REQ-3: An "Assign Colors" button to invoke the color assignment dialog.

V3.5 REQ-3: A dialog to assign a color to each bit.
V3.5 REQ-3: A slider bar to select a bit's color.
V3.5 REQ-3: A button to invoke the Color Chooser to select the bit's color.
V3.6 REQ-3: A "Cuts" panel able to display charts.
V3.6 REQ-3: A radio button set to select the direction of the cut.
V3.6 REQ-3: A mouse click action to select the location of the cut.
V3.7 REQ-3: A "Pointing" panel to display the pointing information.
V3.7 REQ-3: A "History" panel to display the processing history.
V4.2 REQ-3: A "DB Info" to display database information.
V4.5 REQ-3: An "Interactive" panel to allow interactive data reduction.
V5.2 REQ-3: An "External" panel to provide a user interface to an external viewing and/or data reduction program.

4.6 SPOR Text Panes

4.6.1 Description

V1.1 For diagnostic purposes, SPOR prints text to a pair of panes placed in a tabbed panel. In the "Messages" pane are printed miscellaneous strings to track what SPOR is doing internally. V1.1 In the "Events" pane are printed string representations of the events sent from one SPOR function to another.

4.6.3 User Input Sequences

V1.1 Clicking on a tab causes the messages for that pane to be displayed.

4.6.3 Functional Requirements

REQ-1: A tabbed panel to contain the two text panes.
V1.1 REQ-2: A text window to display standard messages from SPOR.
V1.1 REQ-3: A text window to display SPOR events.

4.7 SPOR Layout Control

4.7.1 Description

V2.7 There are many SPOR functions, and although the splitpane dividers between SPOR panes allow resizing of the views, the layout may become too large to be practical. Furthermore, as the users' work progresses, some SPOR functions may no longer be needed, while others not previously needed may come into play. For example, the user may choose all the data sets before proceeding to the Plan and Run stages. In that case, the Selection function view is no longer needed after the initial selection sequence, but an Inspection pane may become useful. To save on screen real estate and encourage simplicity of view, SPOR allows dynamic repopulation of pane contents.

4.7.3 User Input Sequences

V2.7 Each panel displayed in a SPOR pane has a close button. When this button is pressed, the pane's contents become empty, and the panel is iconified in the main SPOR toolbar, next to the "Quit" button. The user can then drag another iconified SPOR function into the empty pane, upon which the function's view is shown in the pane. Furthermore, panes can be added by the tool bar "Add Pane" button, or removed by pressing the panes closing "X" button that appears when the pane is empty. The layout selection dropdown menu in the SPOR toolbar allows selection among various vertical and horizontal splitpane configurations.

4.7.3 Functional Requirements

V2.7 REQ-1:	A close button on every SPOR function view.
V2.7 REQ-2:	An iconification of every SPOR function upon closing.
V2.7 REQ-3:	Drag and Drop cabability of populating SPOR panes from the icons.
V2.7 REQ-4:	A button in each empty pane to remove it from the layout.
V2.7 REQ-5:	A dropdown menu to select among splitpane layouts.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

SPOR must run rapidly enough so that user input is not slowed. Methods which take a long time to execute must not block user input. This is being satisfied in the current implementation by causing the pipeline invocation to run as a thread.

5.2 Security Requirements

Any security issues concerning offline operation of the pipelines are inherited by SPOR, especially the potential of overwriting the results of previous pipeline runs, since in general the pipelines invoked by SPOR read from and write to the filesystem. It is recommended that reprocessing be done in a private copy of the automated processing directory structure. Policies for accessing data from the sandbox or archive should be followed in the context of using SPOR. SPOR itself reads and writes property files to remember user settings, but these files are uniquely named and will not interfere with other software.

5.3 Software Quality Attributes

The main attribute of SPOR must be ease of use, for it is currently possible, though difficult, to run the pipelines offline without SPOR. While constrained by the previously determined pipeline design, details of the pipeline implementation should be handled by the internal workings of SPOR, leaving as its outward form the clear and rapid facilitation of user input.

SPOR is being designed via the use of software patterns wherever possible, for robustness and for clarity of code to facilitate future maintenance. Each SPOR function is written as a package contains its own view class contained in a panel, so that it can be inserted into SPOR or perhaps another application as a discrete component.

6. Other Requirements

It is expected that user feedback will result in requests of increased functionality. To aid in extensibility, SPOR is being designed using an event-driven software architecture model. The design is based on that used in the SSC uplink software. Further information can be found in the source code documentation, in preparation.

Appendix A: Glossary

SPOR	SSC Portal for Offline Reprocessing
SSC	SIRTF Science Center
SIRTF	Space Infrared Telescope Facility
MIPS	Multi-band Infrared Photometer for SIRTF
FITS	Flexible Image Transport System
IOC	In-Orbit Checkout
GTO	Guaranteed Time Observer
JVM	Java Virtual Machine
ISG	IPAC Systems Group
IPAC	Infrared Processing and Analysis Center
DCE	Data Collection Event

Appendix B: Analysis Models

Class diagrams for all the packages in SPOR have been produced and can be found in the source code documentation, in preparation.

Appendix C: To Be Determined List

There are no TBDs for the current version. The exact use and form of features to be added in future versions are yet to be determined.