Enhance NWSRFS Deterministic Verification Tools for AWIPS OB8.2 Software Design Document OSIP 06-023

Hank Herr

Table of Contents

1.0 Overview

This document provides design information at the level of a developer being tasked with implementing some of these changes. The enhancements to be made to the verification software for OB8.2 are documented in the HOSIP *CONOPS and Requirements Specification Document*, and the design components needed to satisfy these new requirements can be broadly classified into the following groups:

- Additional data types
- Pairing with processed data
- New verification metrics
- Special plots
- ChartDirector change
- Comparison variable
- New variables used to select pairs
- Confidence intervals
- Vfyruninfo Editor enhancements
- IVP Batch Builder upgrades
- Design improvements
- Miscellaneous changes

Section 2.0 provides the general, overall design of the verification software. Sections 3.0 and later describe the design of software changes to be made for each group of requirements.

The complete design diagrams can be found in Appendix A. In the following sections, references to the design diagrams will be provided based on their diagram numbers.

> *NOTE: In the following Sections, whenever physical element, duration, extremum, or type source are used, it is the SHEF physical element, SHEF duration, SHEF extremum, and SHEF type source that are being referred to.*

2.0 High-Level Design

The high-level design of the Interactive Verification Program (IVP) for the AWIPS OB8.2 release does not differ much from that for the AWIPS OB7.2 release. An outline of the designs for OB7.2 and OB8.2 are provided in the next two sections.

2.1 Current OB7.2 Design

The system design of the OB7.2 verification software includes the following:

- Written in Java 1.5
- Forecast, observed, and processed data, verification forecast-observed pairs, and some location parametric information are stored in the archive database on the AWIPS RAX machines.
    - Forecast data is stored in the pedfsep table.
    - Observed data is stored in the pecrsep table.
    - Processed data is stored in pehpsep table.

    - Forecast-observed pairs are stored in the vfypairs table.
    - Verification location parametric information is stored in the vfyruninfo table.
    - Critical stages are stored in the rivercrit table.
    - RFC identity is stored in the location table.
    - Verification add-adjustment factors are stored in the vaddadjust table.
- Database access is provided by the JDBC API
    - Incorporates a postgres driver.
- The batch file interface, via the IVP Batch Program, provides the user the ability to (1) build pairs, (2) set parameters of a verification run, (3) compute verification statistics, and (4) generate graphics.
    - Commands are used to define parameters that specify what data to pair, what data to verify, how to do the verification, and how to build the graphic.
    - Actions are used to perform some kind of action, including building forecast-observed pairs, calculating statistics, and generating a graphic.
- The graphical user interface, via the IVP, provides tools to manage the same parameters provided in the batch file interface.
    - Internally, the GUI defines command token values corresponding to batch file commands and then executes actions as needed.
    - For a particular graphic, the GUI can be used to create a batch file for the IVP Batch Program that, when executed, will generate the same graphic and save it to a file.
- Chart rendering is done using JClass DesktopViews.
- Charts may be saved to files in PNG or JPEG format.

2.2 Proposed OB8.2 Design

The system design of the OB8.2 verification software includes the following (changes from OB7.2 are in red):

- Written in Java 1.5
- Forecast, observed, and processed data, verification forecast-observed pairs, and some location parametric information are stored in the archive database on the AWIPS RAX machines.
    - Forecast data is stored in the pedfsep, pehfsep, and peqfsep tables.
    - Observed data is stored in the pecrsep, pedrsep, and peoosep tables.
    - Processed data is stored in pehpsep, pedpsep, peqpsep, and peoosep tables.
    - Forecast-observed pairs are stored in the vfypairs table.
    - Location parametric information is stored in the vfyruninfo table.
    - Critical stages are stored in the rivercrit table.
    - RFC identity is stored in the location table.
    - Verification add-adjustment factors are stored in the vaddadjust table.
- Database access is provided by the JDBC API
    - Incorporates a postgres driver.
- The batch file interface, via the IVP Batch Program, provides the user the ability to (1) build pairs, (2) set parameters of a verification run, (3) compute verification statistics, and (4) generate graphics.
    - Commands are used to define parameters that specify what data to pair, what data to verify, and how to do the verification.
    - Actions are used to perform some kind of action, including building forecast-observed pairs, calculating statistics, and generating a graphic.

- The graphical user interface, via the IVP, provides tools to manage the same parameters provided in the batch file interface.
    - Internally, the GUI defines command token values corresponding to batch file commands and then executes actions as needed.
    - For a particular graphic, the GUI can be used to create a batch file for the IVP Batch Program that, when executed, will generate the same graphic and save it to a file.
- Chart rendering is done using ChartDirector Version 4.0.
- Charts may be saved to files in PNG or JPEG format.

3.0 Additional Data Types

The following new or changed requirements are part of this group:

- 1.1.1
- 1.1.2.2 – 1.1.2.6
- 1.2.1.2.2
- 1.2.1.3
- 1.2.1.4
- 1.2.1.11
- 1.2.1.14
- 1.2.2.1.1.1
- 1.2.2.1.1.2

3.1 Design

3.1.1 Archive Database Changes

The following fields will be added to the vfyruninfo table:

- dur: The forecast point's duration code
- extremum: The forecast point's extremum code
- active: A binary field defining if the forecast point is active or not.
- national: A binary field defining if the forecast point is to be used in computing national statistics.

The dur and extremum fields will become part of the primary key, since they, along with the location id and physical element, uniquely define a forecast point or location. Furthermore, the forecast type source will also become part of the key, so that it is possible to have different sensor preferences for each forecast type source. Previously, all forecast type sources for a given forecast point had to have the same sensor preference list.

3.1.2 Vfyruninfo Table Classes

The VfyruninfoRecord and VfyruninfoTable classes will be enhanced to read in and process the new dur, extremum, active, and national fields of the vfyruninfo table. The VfyruninfoTableModel class will be enhanced to display columns for all of the new fields in the table within the Vfyruninfo Editor. Other classes of the Vfyruninfo Editor will be enhanced, as needed, to account for the new columns. Lastly, the rows of the VfyruninfoTableModel class will be defined so that the forecast type source becomes part of the key for a given row, implying each row will have a single forecast type source.

3.1.3 Other Database Classes

Each table that will be accessed for forecast, observed, or processed data will have a class created for it that is associated with a record in that table. The required tables are as follows:

- Forecast: pedfsep, pehfsep, peqfsep
- Observed: pecrsep, pedrsep, peoosep
- Processed: pehpsep, pedpsep, peqpsep

For forecast data, this record class will be a subclass of the IVPForcastRecord class, which will be newly created. For observed or processed data, this record class will be a subclass of the IVPObservedRecord class, which was created for OB4.

Each table will also have a class created for it that is associated with the entire table. These classes provide methods that query the table and return appropriate record classes containing the results. Each table class will be a subclass of DbTable, which is part of the OHD-Common Java library. Furthermore, in order to facilitate the need to query for forecast records in small chunks at a time, a special interface will be created that all forecast table classes must implement that includes a method to read in blocks of records resulting from a query, instead of all records at once.

Though automatically generated code does exist for all tables of the archive database, this code does not provide all of the required functionality. In particular, the ability to access pseudo-array table columns via an array interface is not provided. For example, the pecrsep table stores stage values for every 15 minutes in a day with each value being in a different column, implying that there are 96 value columns and 96 SHEF qual code columns in one record of the pecrsep table. The automatically generated code provides access to these columns individually. Array-like access is necessary for coding simplicity. Hence, new table and record classes will be written.

Appendix A: See the PairingProcessor diagram (Diagram 2) and data flow diagram (Diagram 7).

3.1.4 Data Handler Classes

The data handler classes associated with forecast and observed data must be enhanced to use the new record and table classes. Furthermore, a new FcstDataRecord class will be created so that the FcstDataHandler returns instances of this class containing forecast data. Previously, since only the pedfsep table was accessed for forecasts, the FcstDataHandler returned instances of its record class, PedfsepRecord.

Appendix A: See the PairingProcessor diagram (Diagram 2) and data flow diagram (Diagram 7).

3.1.5 Pairing Algorithm Class

The PairingProcessor class will be enhanced to use the new FcstDataRecord class being returned by the FcstDataHandler.

Appendix A: See the PairingProcessor diagram (Diagram 2).

3.1.6 Verification Location Classes

Added to the VerificationLocation class will be the ability to determine if two locations are compatible (i.e. they can be placed in the same group of locations for verification statistic computation) based on the data types. This will be done by using a new ArchiveDataType class to determine the unit for the locations' physical elements and making sure the units are identical.

Also, the VerificationLocation class will be changed to read in both critical stages and critical discharge (flow) values from the rivercrit table of the archive database. This will allow for categories to be defined relative to critical discharge values in addition to stages.

Appendix A: See the VerificationLocation diagram (Diagram 3).

3.1.7 Other Batch Classes

Tokens `DUR`, `EXTREMUM`, and `ACTIVE_STATUS` will be added to both the GroupBatchProcessor and PairingBatchProcessor classes, allowing for commands with those names to be specified by the user. These commands provide the ability to restrict locations involved in verification and pairing based on duration, extremum, and whether the point is active. Furthermore, the `PE` token, which already exists for the GroupBatchProcessor, will be added to the PairingBatchProcessor, allowing for users to restrict locations involved in pairing based on SHEF physical element.

Appendix A: See the BatchProcessor diagram (Diagram 1).

3.1.8 GUI Classes

Within the Verification Location Manager window, the VerificationLocationJTableRowData class will be enhanced to include the columns dur, ext (for extremum), ts (forecast type source), and act (active status) in the table. The Category Editor Manager will be updated to allow for setting categories relative to critical discharge values as well as critical stages. The VerificationLocationMgr class will be updated to split the existing table into two components: a table for locations that have not been chosen for display, and a table for locations that have been chosen. Each table will have its own **Select All** and **Select None** buttons. The **Show** button will be removed, as this functionality will be moved to the IVP Data Display. The tables will be instances of a new class called VerificationLocationJTable. The class LocationFilterPanel will be created providing tools for selecting rows from the tables based on the pe, dur, ext, and ts columns, as well as whether or not any of the critical values are undefined (missing).

The IVP Data Display will be enhanced to include a list of verification locations (by location id and SHEF code). From this list, one location can be selected, and this location will be "shown", just as if the **Shown** button had been clicked in the Verification Location Manager of the AWIPS OB7.2 IVP.

The Vfyruninfo Editor will also undergo significant changes to allow for users to set the dur, active, and national columns of the vfyruninfo table. The exact design of those changes has not yet been determined.

Appendix A: See the VerificationLocatioMgr diagram (Diagram 4).

3.1.9 Chart Classes

All drawn charts must be enhanced so that the labels match the data types. Previously, all data types were assumed to be stage. Now, it must be flexible based on the SHEF physical element of the data. Additionally, the units must be displayed for all axes. The class ArchiveDataType will be created which provides unit information based on the physical element of the pairs data.

4.0 Pairing With Processed Values

Pairing with processed values refers to forecast-observed data pairs that are generated where the observation comes from one of the tables accessed by the processed decoder on the RAX machines. This includes the pehpsep, pedpsep, and peqpsep tables. The following new or changed requirements are part of this group:

- 1.2.1.12

4.1 Design

4.1.1 Archive Database Changes

A new table called vfyprocpairs will be added to the archive database. It will have the exact same schema as vfypairs, but will store only those pairs constructed from processed (as opposed to raw) values. In other words, it will store pairs when the command `OBS_TYPE` set to `'PROCESSED'`.

This is being done to address the problem that pairs constructed using observed values must not overwrite those constructed using processed values in order for requirement 1.2.1.12 to be satisfied. However, they still need to be able to overwrite pairs constructed using other observed values, in case the new observed value is preferable (based on its sensor type) or the old observed value is removed. There were two identified solutions: (1) add a column to vfypairs that records if the pair is build using observed or processed values, and (2) create a new table to store processed values. Option 2 has been selected due to simplicity of development.

4.1.2 Software Changes

The following changes will be made to facilitate this requirement:

- The PairingProcessor class will be updated so that the insert statement to insert new pairs will use the appropriate table based on the `OBS_TYPE` token value.
- The PairingBatchProcessor will have the new batch command `OBS_TYPE` added, which must be either "`PROCESSED`" or "`RAW`".
- The VerificationLocation class will be changed to include a new attribute to be associated with a verification location. The new attribute is _observationType and will have one of three values:

    - "`PROCESSED`"
    - "`RAW`"

  This field will tell the VerificationProcessor whether to load pairs constructed using processed or raw table data.

- The LocationBatchProcessor will have the new batch command `OBS_TYPE` added, as well, which will be used to specify the value of _observationType within the VerificationLocation class.
- The VfypairsDataHandler class will be changed to determine which table to query based on passed in parameters or the new _observationType attribute of the VerificationLocation class.
- The VerificationLocationJTableRowData classes will have a column added to the table with the name ‘obstype’. The column will have one of two values, depending on the location’s _observationType attribute setting: “proc” (for “`PROCESSED`”) or “raw”.
- A new button will be added to VerificationLocationMgr that will spawn a GenericRadioButtonSubCommandPanel. This panel will be used change the setting of the obstype column for all of the currently selected rows of the table.

Appendix A: See the BatchProcessor diagram (Diagram 1), PairingProcessor diagram (Diagram 2), VerificationProcessor diagram (Diagram 3), and VerificationLocationMgr diagram (Diagram 4).

4.1.3 Description

For pairing, through the new command `OBS_TYPE`, the user will be able to specify if pairs are to be constructed using either raw values, processed values, or both, independently (requiring two separate executions of the pairing algorithm, with the `OBS_TYPE` being different).

For statistics computation, both raw and processed pairs can be used independently (by defining the verification location twice, with the `OBS_TYPE` being different). However, since each location can only be defined once within a verification group, raw and processed pairs for one location may not be grouped or lumped together to compute one set of statistics.

For graphics, because building the plots depend upon identifying a location uniquely by its lid and SHEF forecast pedtsep, a single graphic built using the GUI cannot display statistics computed for the raw pairs and the processed pairs for one location simultaneously. Only one or the other can be used. In batch mode, if the location is defined twice for the different `OBS_TYPE` values, then the later definition takes precedence.

5.0 New Verification Metrics

The following new or changed requirements are part of this group:

- 1.2.3.1

5.1 Design

A new subclass of VerificationStatistics will be created, called MomentsVerificationStatistics, which will be responsible for computing the mean, variance, correlation, and bias metrics.

Another subclass of VerificationStatistics which will be created is SkillVerificationStatistic. This will be a class capable of taking a list of forecast type sources to be used as the reference forecasts and calculating skill scores. For each type of reference forecast, this class will be subclassed. Since the super class will be responsible for calculating the scores, the subclass is only responsible for naming the columns in the output, so that each type of reference forecast will have different named columns in the output. For persistence forecasts, the name of the subclass will be PersistenceSkillVerificationStatistics class. The only current statistic calculated by SkillVerificationStatistic is the RMSE-SS. It will also provide the forecast RMSE, reference RMSE, and sample sizes for both RMSEs.

The Gilbert score and ROC area will be added to the CategoryVerificationStatistics class. The ROC area will be calculated via a new class to be created for the ROC special plot (see Section 6.0), but output via the CategoryVerificatonStatistics class.

The existing VerificationProcessor class will be enhanced to include the new subclasses of the VerificationStatistics class.

The IVPStatisticInfo class will be updated to include new StatisticGroup instances for the new groups of statistics, and the IVPStatisticChooserMgr class will use a new MultiCheckboxPanel class for each of the new StatisticGroup objects.

Appendix A: See the VerificationProcessor diagram (Diagram 3) and the IVPStatChartMgr (Diagram5).

5.2 Description of Metrics

As with current computation, all metrics are computed for a specific analysis interval, lead time interval, etc., as specified by the user. The new metrics are as follows

- Bias (%): The fraction (forecast mean)/(observed mean). This will not be expressed as a ratio, as opposed to percentage, within the output.
- Pearson correlation coefficient: Computed for the forecast and observed values.
- Forecast or observed means: The mean of the forecast or observed value.
- Forecast or observed standard deviation: The standard deviation of the forecast or observed values.
- Gilbert Score: ($hits - hits_{random}$)/($hits + misses + false\ alarms - hits_{random}$), where $hits_{random}$ = ($hits + misses$)*($hits + false\ alarms$)/$total$. Note that, for this software, a category of values always has a lower and upper bound, so that a forecast/observed value can be below, within, or above a category. With that in mind, a *hit* occurs when the forecast is within or below a category of values, and the observed is as well. A *miss* occurs when the forecast is within or below a category, the observed is above. A *false alarm* is when the forecast is above a category, but the observed is within or below.
- ROC Area: The estimated area under an ROC curve.

More details on these statistics can be found in the *National Weather Service River Forecast Verification Plan*. This document can be found at:

http://www.weather.gov/oh/rfcdev/docs/Final_Verification_Report.pdf

6.0 Special Plots

The following new or changed requirements are part of this group:

- 1.3.6

6.1 Design

For each special plot, a subclass of VerificationStatistics (which is a subclass of DataSet) will be created. These classes will calculate the numbers needed for the special plot and store them internally.

The VerificationProcessor will be enhanced to allow for construction of the diagram. When it is called to produce a special plot, it will call the appropriate new subclass of VerificationStatistics to compute the numbers. Then when the VerificationGroupOutputLine is acquired from the VerificationStatistics subclass, instead of storing Double objects (the values of the statistics), it will store a single DataSet object, which specifies points on the plot.

A subclass of IVPStatChartDrawer will be created called IVPSpecialChartDrawer, which will be responsible for transforming DataSet objects found in the VerificationGroupOutputLine outputs into a single PlotData object ready for plotting via ChartDrawer methods. Additionally, a subclass of IVPStatChartProperties will be created called IVPSpecialChartProperties. This class will do mostly what the super class does, except for the computation of axis limits and the construction of plot labels.

The StatisticGroup class will be enhanced to make it so that only one special diagram plot can be selected at a time. The MultiCheckboxPanel class will be enhanced so that, if the StatisticGroup only allows for one special diagram to the selected at a time, then the master checkbox, which is used to select all checkboxes in the group at once, will be disabled.

The VerificationPlotDefinitionMgr class, associated with the Plot Definition Manager window, will be enhanced to react to special plots in a special way. If a special plot is selected by the user, certain fields of the window will be disabled, since those fields will have default values assumed for them. For example, with a ROC diagram, the user defined categories have no meaning, since they are varied in order to calculate the points of the diagram. Also, whenever any special plot is selected, the variable for the x-axis is known and fixed, so that the user will not be able to select an x-axis variable.

Appendix A: See the VerificationProcessor diagram (Diagram 3) and the IVPStatChartMgr (Diagram5).

6.2 Description of Special Plots

A special plot differs from a standard plot in that many of the user-adjustable parameters for a standard plot are not adjustable for a special plot. Specifically, only the comparison variable can be specified for a special plot. All other parameters, including the primary and second y-axis variables and plot types, and the x-axis variable, are fixed. For each special plot, the comparison variable is used to break down the pairs in order to compute one plot for each value of the comparison variable, with the value being given in the plot legend.

The special plots are as follows:

- *Cumulative Distribution Plot*: An empirical estimate of the cumulative distribution of the variable for which the category is set to “Do Not Use” in the Verification Plot Definition Manager or “NONE” in the batch language. The points on the function are shown as scatter points connected by lines.
- *Probability Density Function*: An estimate of the probability density function of the variable for which the category is set to “Do Not Use” in the Verification Plot Definition Manager. The

function is estimated using 10 intervals evenly spaced from the largest value to the smallest value of the variable. The number of instances of the variable value within each interval is counted, and that number is scaled such that the total area of the histogram would be 1. Though it is computed as a histogram, in the plot, in order to make it easier to view, the points on the probability density function will be connected by lines, where the x-coordinate of each point is the midpoint of the corresponding interval.

- ROC Diagram: An estimate of the relative operating characteristic curve. It is computed point-by-point by defining some threshold for the observed value such that if it is exceeded some event of interest occurs. For example, using flood stage as the threshold value implies that the event of interest is flooding. Next, a threshold for the forecast value is defined such that if the forecast exceeds the threshold, an action is taken; the action being appropriate for when the observed value is above its threshold. For example, if the forecast threshold is the action stage (corresponding to an observed threshold of flood stage) then the action taking may be preparing for a flood. Then, for any forecast threshold, two numbers are computed:

    - Probability of Detection: Given the observed threshold is exceeded, the probability that the forecast threshold is exceeded.
    - Probability of False Detection: Given the observed threshold is not exceeded, the probability that the forecast threshold is exceeded.

  The ROC is a plot of POD against POFD for many possible forecast thresholds. In this case, 100 thresholds are used, being equally spaced from the smallest forecast or observed value to the largest forecast or observed value.

7.0 ChartDirector Change

The following new or changed requirements are part of this group:

- 11.1.1

7.1 Design

First, the existing ChartMgr class will be broken down into two classes:

- ChartDrawer: This class is responsible for managing ChartDirector's XYChart object that underlies all charts. It creates the chart and formats it based on user settings (via a ChartProperty object).
- ChartMgr: This class provides a GUI to display the chart.

This breakdown is necessary in order to satisfy the need that a chart can be generated via the IVP Batch Program without the ability to open a window. The IVPDataChartMgr and IVPStatChartMgr classes must also be broken down, accordingly.

Next, ChartDrawer will be updated to use ChartDirector instead of JClass Chart.

Then, a new class called AxisLimitsAndSpacing will be created, which is responsible for managing the axis limits and tick mark spacing for a given axis. Additionally, it will provide functionality for computing the limits and spacing given the largest and smallest values associated with the axis. This is necessary because experimentation showed that ChartDirector does not do an adequate job of managing axis limits and spacing.

Lastly, a NOAA watermark will be added to all charts via methods in the underlying XYChart class. The NOAA and NWS logos, in the upper left and upper right corners of the graphics in OB7.2, will be removed.

Appendix A: See the ChartDrawer and ChartMgr design diagram on page 8. Also see the IVPDataChartMgr diagram on page 6 and IVPStatChartMgr diagram on page 7.

8.0 Comparison Variable

The following new or changed requirements are part of this group:

- 1.3.4

8.1 Design

First, the needed command must be added to the batch language and the GUI. The PlotDefinitionBatchProcessor will be enhanced to include the new batch command `COMP_VARIABLE`. The VerificationPlotDefinitionMgr class will be enhanced to allow for specifying the comparison variable in a way exactly analogous to how the x-axis variable is specified. Also, it will be enhanced to make sure the x-axis variable and comparison variable are never identical.

Then, the charts must incorporate the comparison variable. The GraphicsProcessor class will be enhanced to adjust the VerificationGroup parameters according to both the x-axis variable and the comparison variable. Currently, only the x-axis variable is used to make those adjustments. The IVPStatChartDrawer class will be enhanced to read in y-axis values based on both the x-axis variable values and the comparison variable values. The IVPStatChartProperties class will be enhanced to create the legend components according to the statistics displayed and the comparison variable. The title will also be enhanced to mention the comparison variable.

Appendix A: See the BatchProcessor diagram (Diagram 1) and the GraphicsProcessor and IVPStatChartMgr diagrams (Diagram 5).

9.0 New Variables Used to Sort Pairs for Computation

The following new or changed requirements are part of this group:

- 1.2.1.8
- 1.2.2.2
- 1.2.2.6
- 1.3.3.1
- 1.3.4.1

9.1 Design

The new variables include (1) issuance time-of-day intervals (new commands `ISSUANCE_START`, `ISSUANCE_END`, `ISSUANCE_STEP`) and (2) forecast type source.

For each variable to add, the following tasks must be completed

- VerificationGroup will be updated to store the new variable settings in appropriate attributes, include sets and gets for the new attributes, allow for iterating through the values of the variable, generate output header information for the new variable, and filter pairs based on the variable values (i.e. VerificationDataFilter interface method).
- GroupBatchProcessor will be updated to input the new batch command value, store it, and pass it into the VerificationGroup in the doDefineGroup method.
- VerificationProcessor will have a new method created that will iterate through the possible variable values using the tools in VerificationGroup. This will be done either before it reads data from the database, or after. If before, an appropriate doVerificationFor* method will be created. If after, an appropriate generateStatisticsFor* method will be created.
- VerificationGroupOutputLine will be updated to include attributes storing the variable values and output methods will be updated to include new columns of output corresponding to the variable.
- VeriifcationPlotInfo will be updated to add the variable to choices available for x-axis and comparison variable settings in a graphic.
- IVPStatChartDrawer will be updated to add necessary fields to the methods that build x-axis and comparison variable labels.
- An appropriate SubCommandPanel will be written and added to the existing MiscSelectionsCommandPanel, or a new CommandPanel will be written allowing user specification of the variable. Existing generic classes will be used, if appropriate.
- VerificationGroupMgr will be updated, if necessary, to include the new SubCommandPanel or CommandPanel.
- VerificationPlotGroupAdjustor will be updated, if necessary, to adjust the new variable settings based on plot settings.
- GraphicsProcessor will be updated to call the new adjustment routine within the VerificationPlotGroupAdjustor, if any.

Another required enhancement is that every combination of variable values specified in the group must result in a VerificationGroupOutputLine being constructed, even if no data is found. In otherwords, every combination of analysis interval, lead time interval, issuance interval, forecast type source, etc. must yield one VerificationGroupOutputLine per category. In OB7.2, no VerificationGroupOutputLine is produced if no data is returned from the query of the vfypairs table. The effect of this can be seen in Figure 1, in which the title shows that lead times are “0-5 days”, but the plot only shows results out to 3 days. There was no data for days 4 and 5.

Appendix A: See the BatchProcessor diagram (Diagram 1), VerificationGroup and VerificationProcessor diagram (Diagram 3), VerificationGroupMgr diagram (Diagram 4), and GraphicsProcessor diagram (Diagram 5).

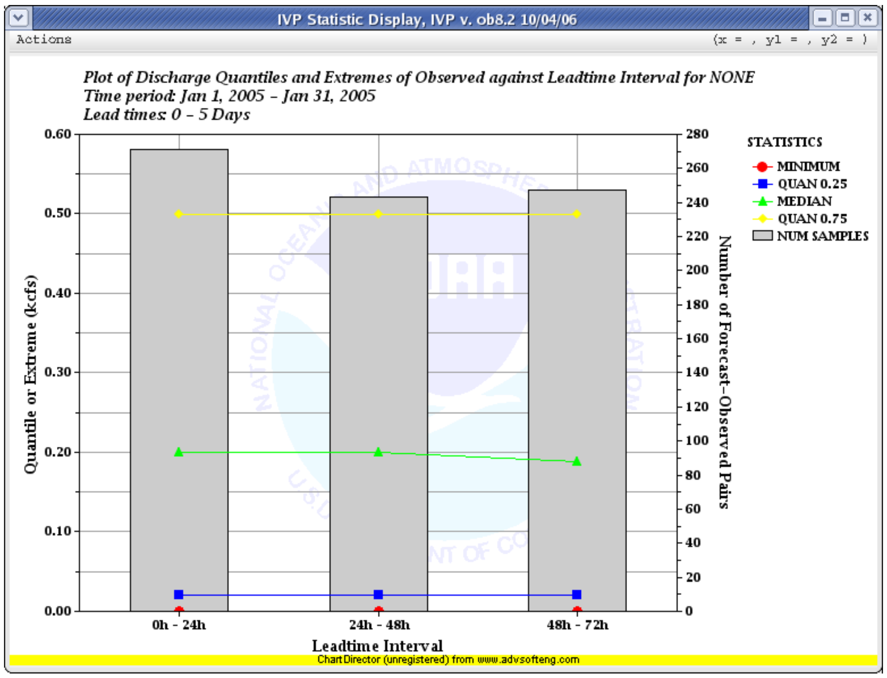

IVP Statistic Display, IVP v. ob8.2 10/04/06
Actions
(x = , y1 = , y2 =)
Plot of Discharge Quantiles and Extremes of Observed against Leadtime Interval for NONE
Time period: Jan 1, 2005 – Jan 31, 2005
Lead times: 0 – 5 Days
Quantile or Extreme (kcfs)
Number of Forecast–Observed Pairs
Leadtime Interval
0h - 24h
24h - 48h
48h - 72h
STATISTICS
MINIMUM
QUAN 0.25
MEDIAN
QUAN 0.75
NUM SAMPLES
ChartDirector (unregistered) from www.advsofteng.com

Figure 1: Plot showing problem with 5 lead time categories being defined but only 3 having data.

10.0 Confidence Intervals

Confidence intervals will NOT be delivered for OB8.2. The design, however, will still be included below. The following new or changed requirements are part of this group:

- 1.2.3.3
- 1.3.5.2
- 5.1.2

10.1 Design

The VerificationStatistics classes will need to be enhanced to be able to produce confidence intervals, if desired, and store the bounds of the interval in a VerificationGroupOutputLine class. This will show up as additional columns in the verification output associated with the lower and upper bounds of the interval.

The batch language will be enhanced to allow for a suffix to be attached to each stat in a `CALCSTATS` command. The suffix, "`_CI`", if attached will mean that confidence intervals will be calculated for each statistic computed.

The IVPStatChooserMgr class will have an on/off switch in it to allow users to specify if confidence intervals are to be included.

The IVPStatChartDrawer class will be enhanced to make statistics that are to include confidence intervals into box-whisker plots (or another appropriate plot type to be chosen). Accordingly, the

VerificationPlotDefinitionMgr class must disable the corresponding Plot Type choice box if confidence intervals are to be calculated, since the plot type is no longer user selectable.

Appendix A: See the VerificationProcessor diagram (Diagram 3) and VerificationPlotDefinitionMgr diagram (Diagram 5).

11.0 Vfyruninfo Editor Enhancements

The vfyruninfo table provides information about what verification locations are available for verifying. The Vfyruninfo Editor GUI was created in the AWIPS OB4 release as a simple tool to populate and change the vfyruninfo table. It was exceedingly simple, providing very few tools to facilitate creating entries in the table. At the time, however, only stage forecasts were verified, the duration and extremum were assumed to be 'I' and 'Z' respectively, and the forecast type sources for a particular location and SHEF pe code would be lumped together in computing statistics. Hence, the simple Vfyruninfo Editor available in OB4 was believed to be sufficient at the time.

That is no longer the case, however. The new vfyruninfo table has a primary key that include location id and the SHEF physical element, duration, type source, and extremum codes. Furthermore, it includes new active and national columns. This justifies the need for a new Vfyruninfo Editor tool.

See Section 3.0 for requirements related to this enhanced design.

11.1 Design

The new VfyruninfoEditorMgr class will be very similar to the new VerificationLocationMgr class, displaying two tables and allowing for selecting rows from the top table for inclusion in the bottom table (see 3.1.8). The top table will contain the contents of the ingestfilter table where the t (type) component of the SHEF pedtsep is equal to 'F'. The user will be able to override this restriction, having the ability to list valid codes for physical element, duration, type source, and extremum components of the SHEF pedtsep. To facilitate this, a new IngestFilterDataHandler class will be written to read in data from the ingestfilter table and return VerificationLocation objects corresponding to the rows. The bottom table will contain the current or edited contents of the vfyruninfo table, as contained by the VfyruninfoTableModel within a list of VerificationLocation objects.

A new VfyruninfoJTable class will be created that builds and maintains the two tables within the VfyruninfoEditorMgr. The rows of the tables will be instances of the new class VfyruninfoJTableRowData. Both of these classes reuse code written for WFO applications.

Tools will be created to allow for the following:

- Selecting entries from the ingestfilter table that are to be part of the vfyruninfo table.
- Creating new records that do not have corresponding entries in the ingestfilter table.
- Removing entries from the vfyruninfo table.
- Editing the sensor preferences, response time, active state, and national flag for each new entry in the vfyruninfo table. This will be done by popping up a window containing an appropriate batch builder SubCommandPanel.
- Saving the contents to the vfyruninfo table.

After creating these classes and testing the new Vfyruninfo Editor, both the VerificationLocationMgr and VfyruninfoEditorMgr will be refactored so that the common components can be abstracted out and code reuse is maximized.

Appendix A: See the Data Flow diagram (Diagram 7). No diagram has been created for the Vfyruninfo Editor, since there are only a few classes used within the GUI.

12.0 IVP Batch Builder Upgrades

12.1 Design

For any batch command added to the IVP Batch Program in order to satisfy a requirement, the IVP Batch Builder will be enhanced. A SubCommandPanel will be created and added to an appropriate CommandPanel within the software. Wherever possible, this should reuse code written for the IVP.

13.0 Design Improvements

13.1 Design

The following design improvements will be made:

- Lists of VerificationLocation instances will be passed around to specify locations to use for verification and pairing. Previously, the VerificationProcessor or PairingProcessor, themselves, were required to find the locations based on parameters of a VerificationGroup object or internal parameters, respectively. For OB8.2, algorithms will be executed within the appropriate BatchProcessor subclass to find the locations within the VfyruninfoTableModel class that correspond to locations that will be processed. These rows will be passed to the processor class for use. This simplifies those classes and allows for reuse of the algorithm to find those rows.
- A DataTable class will be created that stores the double-array table that DataSet manages. This class will allow for samples/variables to be either the first or second index within the double array. It will also provide methods that return data from the tables (samples or variables) by pointer instead of copying it, where possible. This should make the chart rendering quicker and more memory efficient, particularly for large data sets.
- A new ChartPropertyLoader class will be created to handle loading/saving chart properties from/to a template file. Furthermore, a check will be added when loading properties from a template file to make sure the properties are compatible with the graphic. If no such check is made, the plot may not look reasonable. This is particularly problematic if the number of legend items in the plot does not match the number of legend items in the template file.

Appendix A: See the ChartDrawer and ChartMgr diagrams (Diagram 6).

14.0 Miscellaneous Changes

The following new or changed requirements are part of this group:

- 1.1.6.1.2
- 1.1.6.1.3
- 1.2.2.4.1

- 1.3.2.1

14.1 Design

The following miscellaneous changes will be made:

- Tokens for the System Settings file will be added to allow for changing the sizes of shapes shown in scatter and line plots and widths of lines shown in line plots. The IVPStatChartProperties and IVPDataChartProperties classes will be enhanced to read and apply these settings.
- The ChartWindowHeight and ChartWindowWidth tokens will be renamed ChartHeight and ChartWidth. The reason is because these tokens actually control the height and width of the chart, itself, not the window that holds it. The size of the window is based on the chart. This was changed with the ChartDirector change and in order to ensure that the batch and GUI modes produce the same size chart.
- A SubCommandPanel will be added to the VerificationGroupMgr class to allow for specifying a pairs file in the created batch file.
- The persistence radio buttons will be removed from the Verification Group Manager. Instead, users will be required to make sure the forecast type source ‘`FR`’ is included in the `FCST_TS` list if persistence forecasts are to be included. Furthermore, a new possible value for the `FCST_TS` command, “`ALL_BUT_PERSIST`”, will be created allowing for all forecast type sources, except ‘`FR`’, to be selected for verification.
- A progress window will be designed to show the progress of loading data after **Create Display** is clicked in the VerificationGroupMgr. Changes will be made to VerificationProcessor to send messages to this window at appropriate times.
- The ObsDataHandler will be enhanced to reject any observed data with a shef_qual_code of ‘B’, ‘R’, or ‘E’ (AWIPS DR 18419). Furthermore, if a sensor preference list has been defined for a forecast, then an observed value will not be considered as a pairing candidate for that forecast value unless its type source is within the sensor preference list (AWIPS DR 18118).
- The analysis subinterval “`YEARLY`” will be added. If specified, then the years covered by the start time and end time of the overall interval will specify years of analysis to perform, which the month and day of the start time and end time will specify the time period over which to compute statistics for each year. VerificationProcessor and VerificationGroup will be changed to allow for a “`YEARLY`” subinterval.
- The ability to have multiple IVP Data Display windows open simultaneously, with each displaying a different “data source”, will be added. This will require a change to IVPDataChartMgr to add a menu item that builds a new VerificationGroupMgr object, and a mechanism to determine how many data sources are currently open.
- Output and pairs files shall be changed so that a group can have both an output and a pairs file associated with it when the group is defined. Then, when `CALCSTATS` is called, output/pairs will be sent to the appropriate file such that output/pairs will never be overwritten within a single `CALCSTATS` call, even if some of the groups use the same files. To do this, a new VerificationGroupOutputFileDirector class will be implemented to (1) determine which files are already opened, (2) open files if necessary and provide BufferedWriter instances to be used for writing to the corresponding file, and (3) closing the files.
- Add an optional setting for the list of locations provided with an `NATLSTATS` action. The setting will be “`NATIONAL`” and will indicate that only those locations for which the national field of the vfyruninfo table is set to ‘`Y`’ will be used in computing the national statistics. This will become the default setting provided with the national statistics template batch file.

- Special replacement strings will be allowed to be part of labels and file names throughout the software. If a label or file name contains a replacement string, then the replacement string will be replaced with an appropriate string. The list of replacement strings, and the replacement value for each, is as follows:

Replacement String	What Replaces It
`@ANALYSISINT`	The analysis interval; either "`MONTHLY`", "`YEARLY`" or "`# hours`".
`@ANALYSISWINDOW`	The analysis window; equivalent to "`@STARTTIME - @ENDTIME`".
`@CATVAR`	Either "`observed`" or "`forecast`" depending on which variable is used to categorize pairs.
`@COMPVAR`	The variable used for comparison, based on the choices available in the **Verification Plot Definition Manager** choice box.
`@DURS`	Comma separated list of duration SHEF codes within the pairs.
`@ENDLDTIME`	The last lead time in the format "# hours".
`@ENDTIME`	The end time of the analysis window, by default in format "`CCYY-MM-DD hh:mm:ss TZC`". The user can set the format manually; see `@STARTTIME` for how to do this.
`@EXTREMUMS`	Comma separated list of extremum SHEF codes within the pairs.
`@FCSTCAT`	The forecast category in the format "`Cat#`", where # is the number of interval, starting from the smallest (# = 1).
`@FCSTTSS`	Comma separated list of forecast type source SHEF codes within the pairs.
`@LDTIMEINT`	The lead time interval in the format "`# hours`".
`@LEADTIMES`	The lead time interval, equivalent to "`@STARTLDTIME - @ENDLDTIME`".
`@LOCATIONS`	Comma separated list of location ids (lid in the archive database).
`@OBSCAT`	The observed category in the format "`Cat#`", where # is the number of the interval starting from the smallest (# = 1).
`@PEDATATYPE`	The data type: precipitation, temperature, height, discharge.
`@PES`	Comma separated list of physical elements SHEF codes within the pairs.
`@RESPTIMES`	The response time setting.
`@RFC`	Three letter RFC id found for the first location in the working verification group.
`@STARTLDTIME`	The very first lead time in the format "`# hours`".
`@STARTTIME`	The start time of the analysis window, by default in format "`CCYY-MM-DD hh:mm:ss TZC`". The user can set the format manually by passing in a format string within "{[" and "]}" following the reference string. For example, "`@STARTTIME{["CCYY-MM-DD"]}`". Note that "`MON`" corresponds to the three letter month abbreviation.
`@STATVAR`	Either "observed" or "forecast" depending on the variable for which statistics are computed. This is the opposite of `@CATVAR`.
`@UNIT`	A unit identifying string (i.e. "ft").
`@XAXISTITLE`	The title displayed by default for the x-axis, based upon the x-axis variable.
`@XAXISVAR`	The variable used for the x-axis, based on the choices available in the **Verification Plot Definition Manager** choice box.
`@YAXIS1TITLE`	The title displayed by default for the primary y-axis, based upon the user specified primary statistics.
`@YAXIS2TITLE`	The title displayed by default for the secondary y-axis, given the user specified secondary statistics.

This change will make it easier for users to specify graphical template files (created from the GUI) and output file names that are generic, enabling users to generate output for many combinations of locations and parameters within a single run of the graphical processor. This will allow for smaller batch files for use with the IVP Batch Program. IVPGenericStringAdjustor class. An instance of this class will be used to adjust any appropriate string prior to use. The one exception to this is that the y-axis and x-axis labels will not be adjustable. Their values are fixed, though the x-axis values can be changed manually via the ChartPropertyMgr, and the y-axis values can be changed by setting the axis minimum and maximum.

15.0 IVP Data Display Enhancements

15.1 Design

The class responsible for the existing scatter plot view will be redesigned. A class called IVPDataHashMapper will be constructed that is responsible for parsing the pairs data, contained within a List of VfypairsRecords, into a List corresponding to each location. The scatter plot drawing mechanism will be moved from IVPDataChartDrawer to IVPScatterChartDrawer. A new PlotData subclass called IVPScatterPlotData will be created that is responsible for returning the needed scatter plot data from the IVPDataHashMapper when the ChartDrawer mechanism demands it.

A new time series view will be added to the IVP Data Display. The IVPDataHashMapper will parse the incoming pairs data into one List per available time series, where a time series is defined by the corresponding VerificationLocation and the forecast basis time. It will also use the ObsDataHandler to load observed data from the archive database and parse those values in Lists, one for each VerificationLocation. The IVPTSChartDrawer class will be created and will be responsible for drawing the forecast time series, observed time series, and focused time series. The class IVPTSPlotDataGenerator will be created and will be responsible for returning the PlotData objects needed by the new IVPTSChartDrawer.

The user will have the ability to focus on locations and time series within the IVP Data Display. To allow for this, a new JListPanel class will be created. It will be a subclass of JPanel and contain a JScrollPane and a JList within the JScrollPane. It will be made generic, so that the using class can specify a title for the panel and the data within the list, and be notified whenever a selection is made.

16.0 Bad Forecast-Observed Pairs

16.1 Design

Within the archive database vfypairs and vfyprocpairs tables, a bad pair will be any pair whose quality_code field value is less than 1073741824. This value is derived from the IHFS database bad quality threshold. The PairingProcessor will be changed to set the initial quality_code field for all generated pairs to 1879048191. This is derived from the default ‘good’ quality_code value for the IHFS database.

Within the VfypairsHandler class, which is responsible for loading data from the vfypairs and vfyprocpairs table, the loaded pairs will be broken down into two Lists based on whether the pair is ‘good’ or ‘bad’. The VerificationProcessor will compute statistics for only the ‘good’ pairs.

The ability to mark a pair as being 'bad' or 'good' will be part of the IVPDataTableMgr. The IVPScatterChartDrawer and IVPTSChartDrawer classes will draw the plots such that 'bad' data is plotted differently than 'good' data.

17.0 IVP Batch File Creation Wizard

17.1 Design

The IVPBatchFileSaveMgr will be replaced by an IVPStatBatchFileMaker class. This class will implement a free, open-source wizard found at *Java.Net*:

https://wizard-framework.dev.java.net/

Each wizard step will be implemented and will allow the user to create a batch file with any date and any locations. The batch file will also be capable of generating multiple images; not just the image viewed in the IVP Statistics Display.

18.0 Design Constraints

There are some design constraints that must be considered when designing these changes. These constraints deal mainly with system resources and performance.

18.1 Memory Issues

Memory problems within the IVP software have been recognized since the OB7.2 software was checked into AWIPS. At that time, it was recognized that, given the (then) current hardware available on the archive (ax) machines, an upper limit on the number of pairs that can be visualized within the IVP GUI software was around 40,000. If a larger number of pairs needed to be viewed, then steps could be taken to expand this limit, and those steps were outlined in the *IVP User's Manual*. A link to this document can be found at,

http://www.nws.noaa.gov/oh/hrl/verification/verification_doc_ob7.php.

By using the steps provided in the manual, internal testing has shown that the IVP is capable of visualizing greater than 120,000 pairs at one time.

Given the design of enhancements for OB8.2, it is not anticipated that the new features should noticeably affect the memory usage for OB8.2.

18.2 Query Performance Issues

It was recognized with the OB7.2 delivery of IVP that there may be performance issues involving querying the archive database while running the IVP GUI. These issues exist with the IVP Batch Program, as well, but they are not as noticeable since it does not have a graphical interface. The solution proposed within the *IVP User's Manual* was to shut down the two archive SHEF decoders while running the IVP GUI software, if performance seems to be poor. Though this may create a backlog of SHEF messages to the archive database, the decoders should be capable of catching up, so long as they are not shut down for too long a period of time. This problem is likely to be more noticeable with the OB8.2 software, because both the greater number of data types to verify and the longer archive history in the database imply a potential for larger database queries.

18.3 Vfypairs Table Size

With the capability of constructing pairs for any data for which the forecast and observed values are stored within the tables accessed by this software, the vfypairs table is likely to greatly increase in size. This has been partially alleviated by adding a new vfyprocpairs table to store pairs constructed using processed observations, breaking down the one big table from OB7.2 into two smaller tables for OB8.2. The vfyprocpairs table, however, will likely contain a small portion of the total number of pairs. If the vfypairs table size becomes a significant issue, the vfypairs table can be further broken down perhaps based on physical element or some other component of the SHEF code. This change will need to be done through a DR or DCS, however.

APPENDIX A: Design Diagrams

Design diagrams are provided on the following pages, with a legend being given on the next page. The design diagrams are each labeled Diagram #, and are as follows:

1. BatchProcessor: High level diagram of the different BatchProcessor subclasses, which are responsible for reading and understanding batch files.
2. PairingProcessor: Diagram of classes used to generate pairs.
3. VerificationLocation, VerificationGroup, VerificationProcessor: Diagram of classes involved in defining locations, defining groups, and computing statistics.
4. VerificationGroupMgr, VerificationLocationMgr, IVPDataChartMgr: Diagram of classes involved in the generation and display of the Verification Group Manager, Verification Location Manager, and IVP Data Display windows.
5. GraphicsProcessor, VerificationPlotDefinitionMgr, IVPStatChartMgr: Diagram of classes associated with defining and viewing plots, both within the IVP Batch Program and IVP GUI.
6. ChartDrawer, ChartMgr, ChartPropertyManager: Diagram of the low level code that draws the plots, views the plots, and manages the chart properties associated with a plot.
7. Data Flow Diagram
8. Window Interactions

These diagrams are referenced throughout this design document by the diagram number.

Legend

Complex Classes We Manage	A complex class that managed as part of this project.
Simple Classes	A simple class, such as an exception or comparator subclass, or an interface, managed as part of this project.
Classes We Do Not Manage	A class that is not managed as part of this project. It may be a system class, automatically generated class, or a reused class managed as part of another project.

The class at the beginning of the arrow is the “source” class; and the end is the “destination” class.

⇒	“Accesses” or “Sees”; usually linked between a graphical class and “User” or a database class an the archive database.
⇢	“Implements” or “Extends”. The destination class either implements the source interface or subclasses the source class.
→	“Uses”: The destination instantiates and uses the source class in some way.
- - - - - - - - -▸	“Throws”: The source class is thrown by the destination class.

BatchProcessor (Batch)

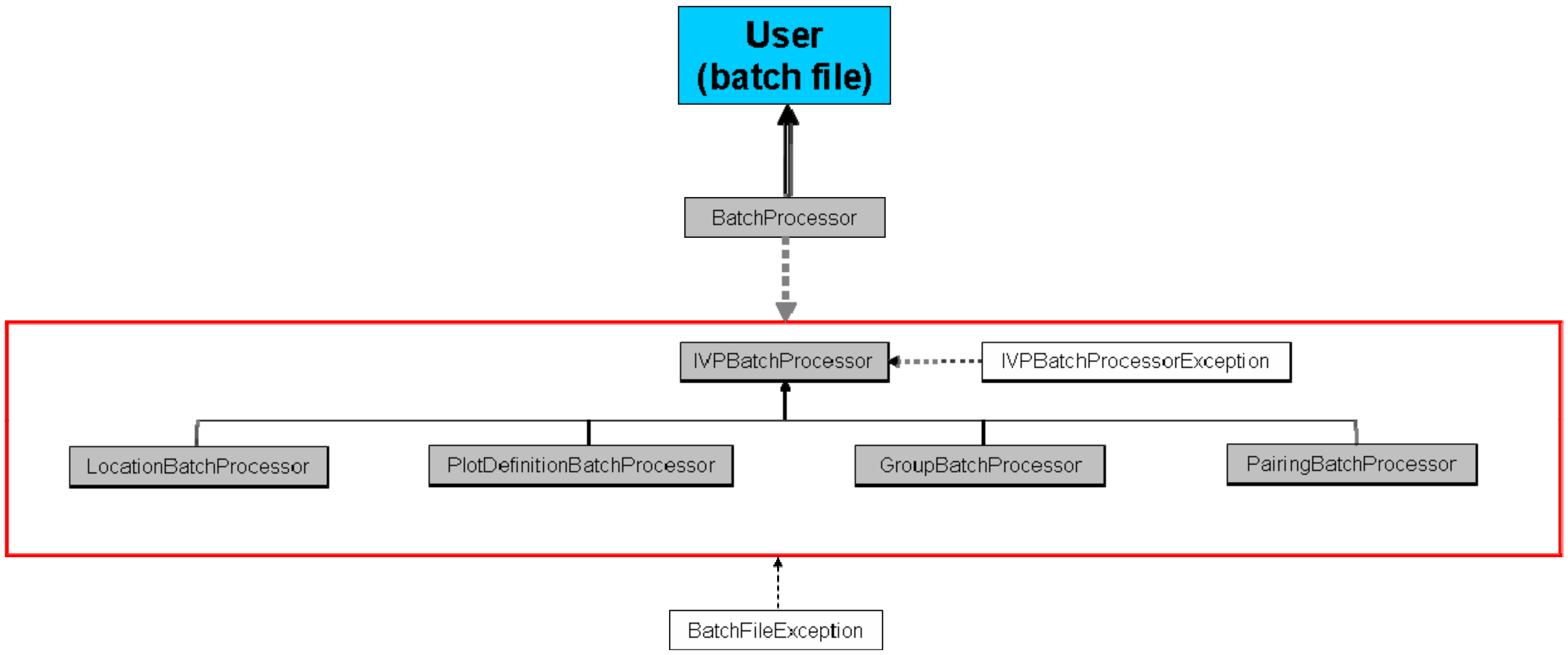


Diagram 1

PairingProcessor (Batch)

- All *Comparator classes are subclasses of Comparator
- All *Exception classes are subclasses of Exception
- All *Record classes are subclasses of DbRecord
- All *Table classes are subclasses of DbTable

* VfyruninfoDataHandler and VfyruninfoTableModel are used throughout the software in MANY classes. It is only included here only so that its class diagram is somewhere and because I do not want to include it everywhere.

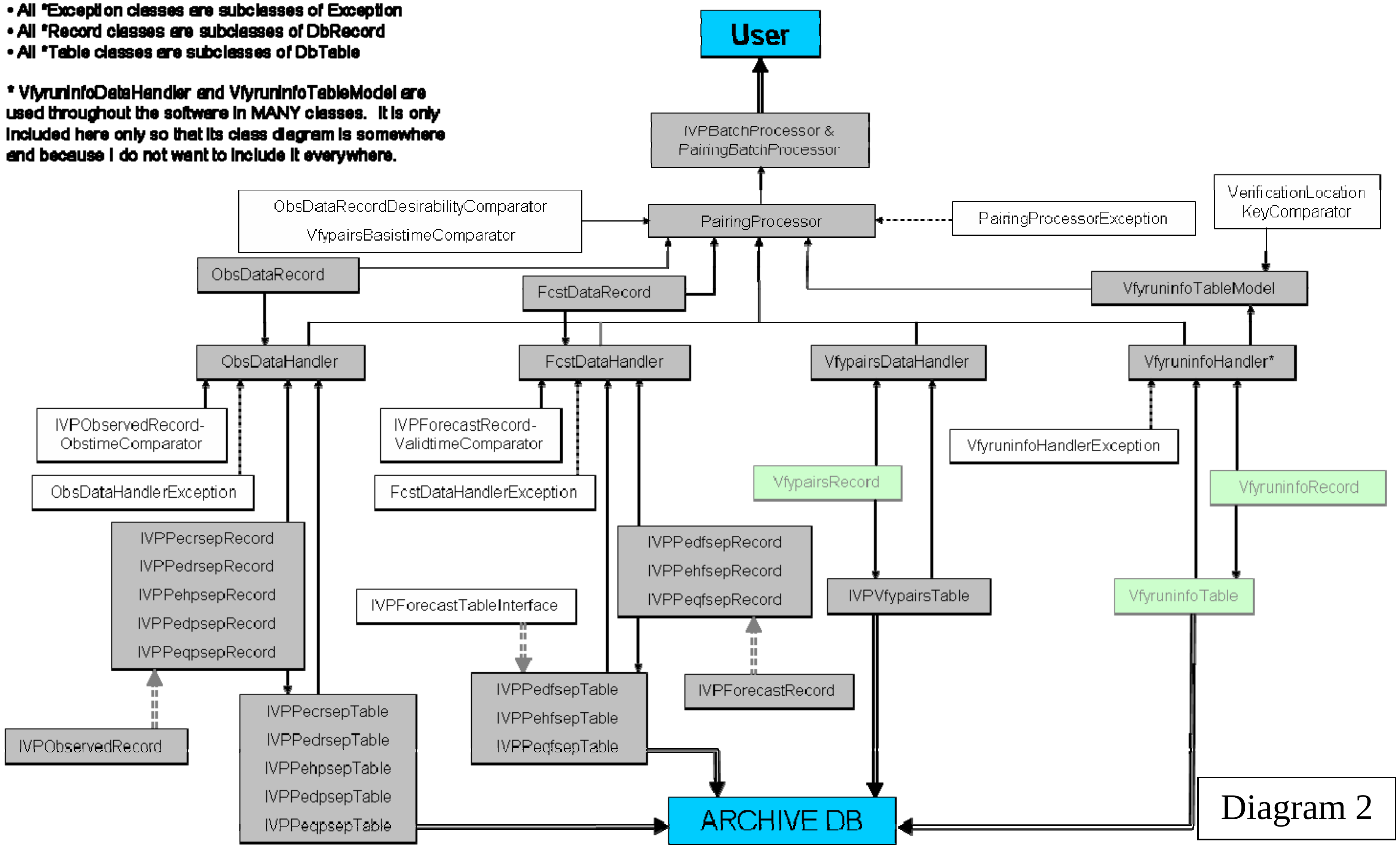
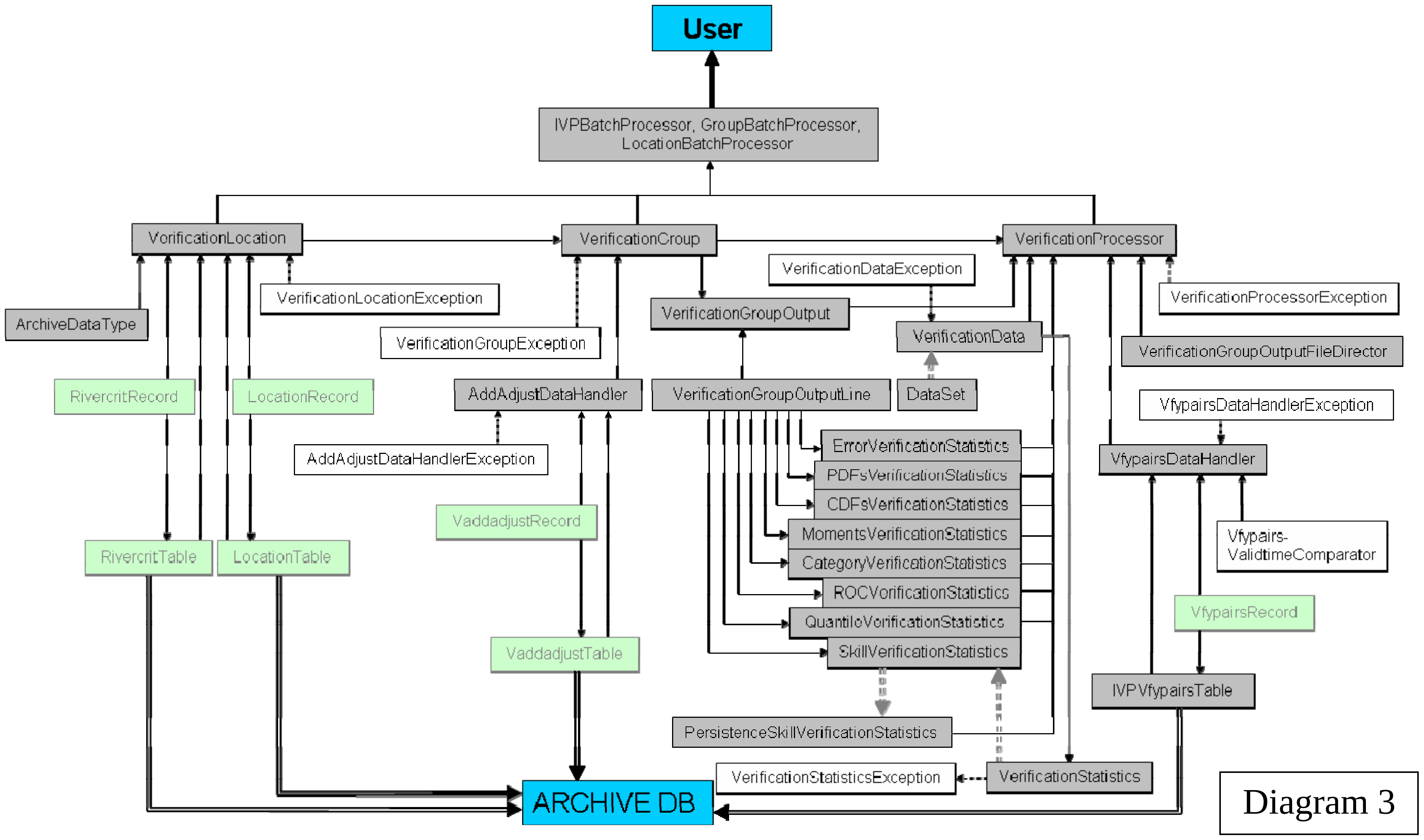


Diagram 2

VerificationLocation, VerificationGroup, VerificationProcessor (Batch)


User
IVPBatchProcessor, GroupBatchProcessor, LocationBatchProcessor
VerificationLocation
VerificationGroup
VerificationProcessor
ArchiveDataType
VerificationLocationException
VerificationGroupException
VerificationDataException
VerificationGroupOutput
VerificationData
VerificationProcessorException
VerificationGroupOutputFileDirector
RivercritRecord
LocationRecord
AddAdjustDataHandler
VerificationGroupOutputLine
DataSet
VfypairsDataHandlerException
AddAdjustDataHandlerException
ErrorVerificationStatistics
PDFsVerificationStatistics
CDFsVerificationStatistics
MomentsVerificationStatistics
CategoryVerificationStatistics
ROCVerificationStatistics
QuantileVerificationStatistics
SkillVerificationStatistics
VfypairsDataHandler
Vfypairs-ValidtimeComparator
VaddadjustRecord
RivercritTable
LocationTable
VaddadjustTable
VfypairsRecord
IVPVfypairsTable
PersistenceSkillVerificationStatistics
VerificationStatisticsException
VerificationStatistics
ARCHIVE DB
Diagram 3

VerificationGroupMgr, VerificationLocationMgr, IVPDataChartMgr (GUI)

[1] See the VerificationProcessor diagram
[2] See the VerificationPlotDefinitionMgr diagram
[3] See the ChartMgr diagram

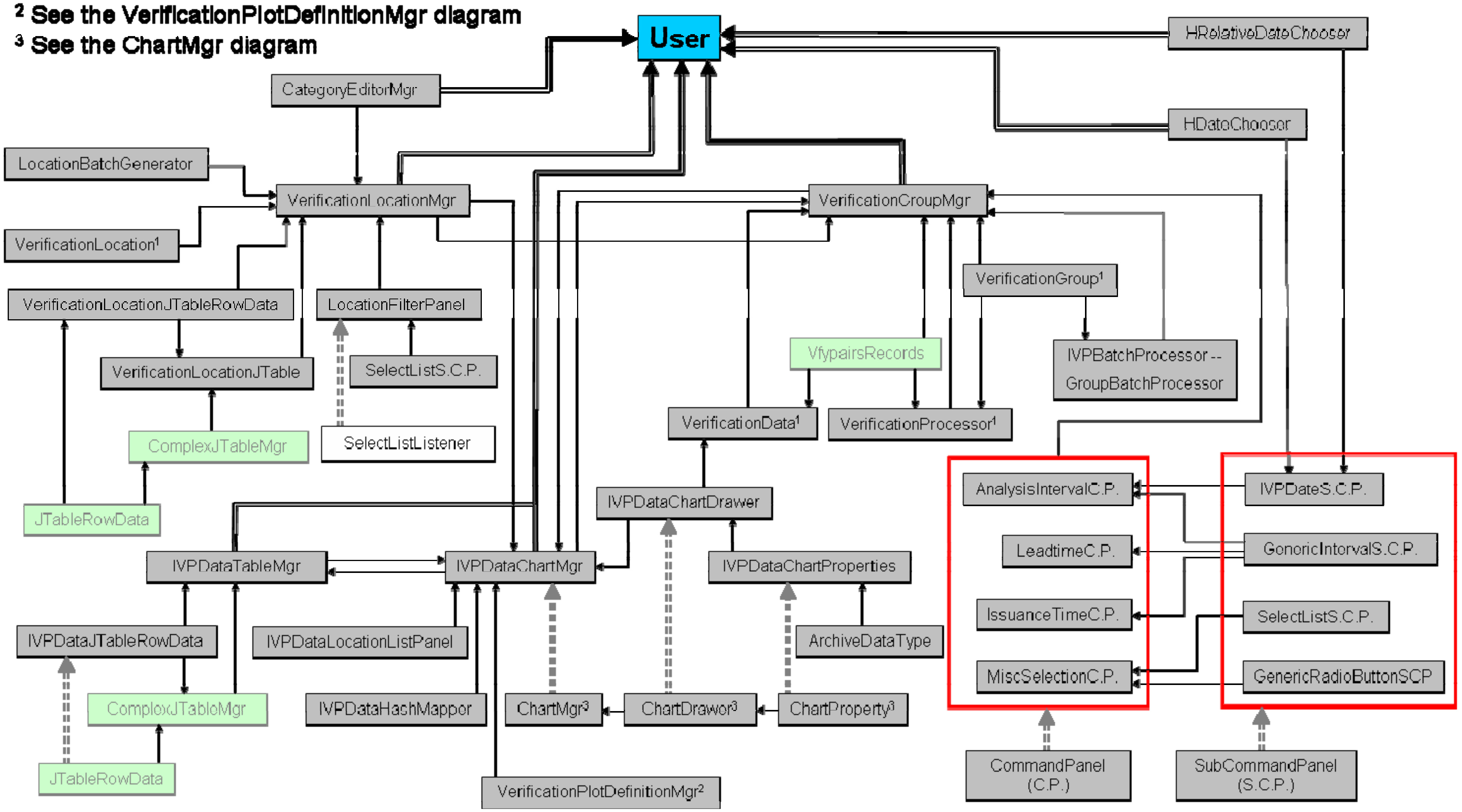


Diagram 4

GraphicsProcessor, VerificationPlotDefinitionMgr, IVPStatChartMgr (Both)

[1] See the VerificationProcessor diagram
[2] See the VerificationGroupMgr diagram
[3] See the ChartMgr diagram

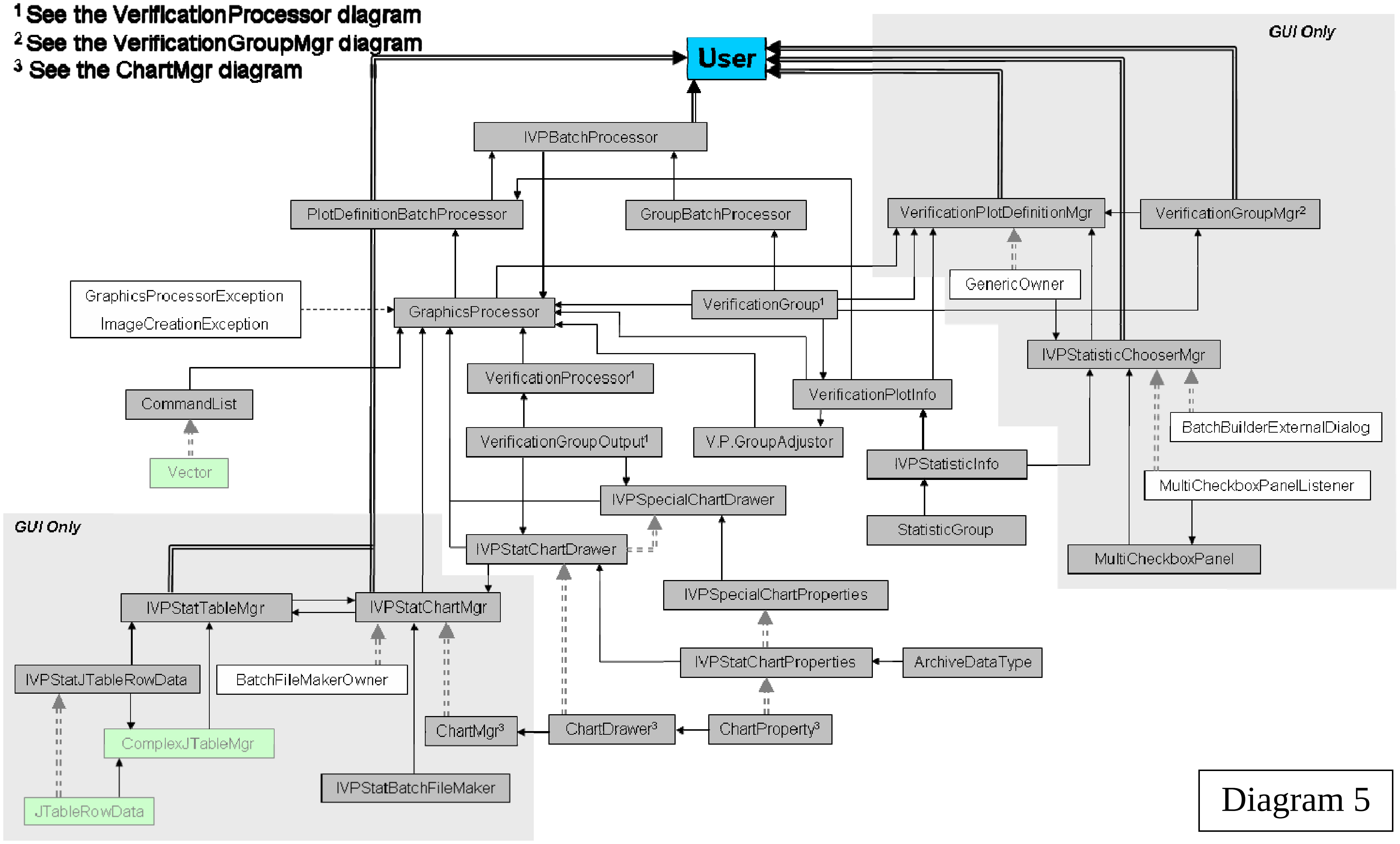


Diagram 5

ChartDrawer, ChartMgr, and ChartPropertyManager (Both)

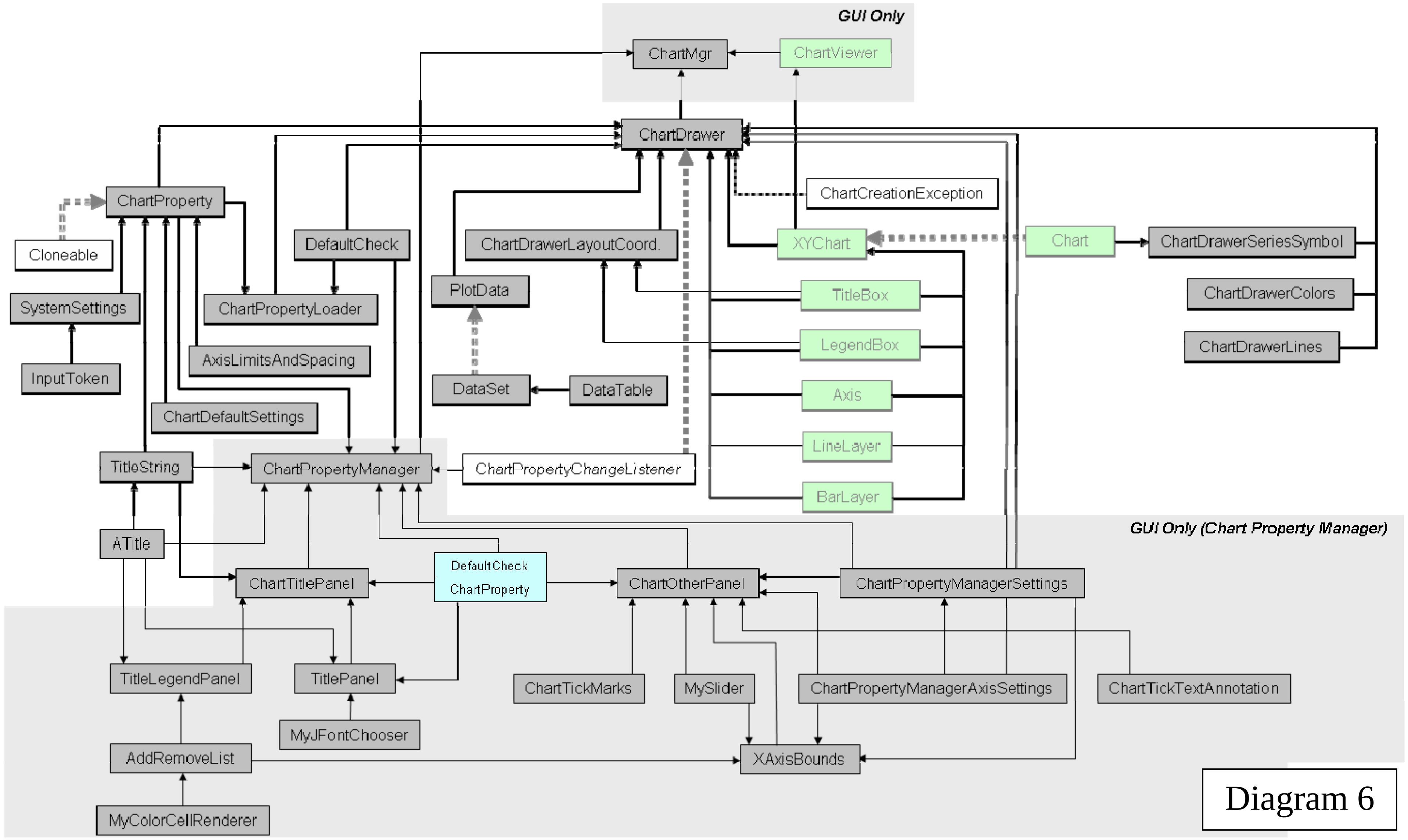
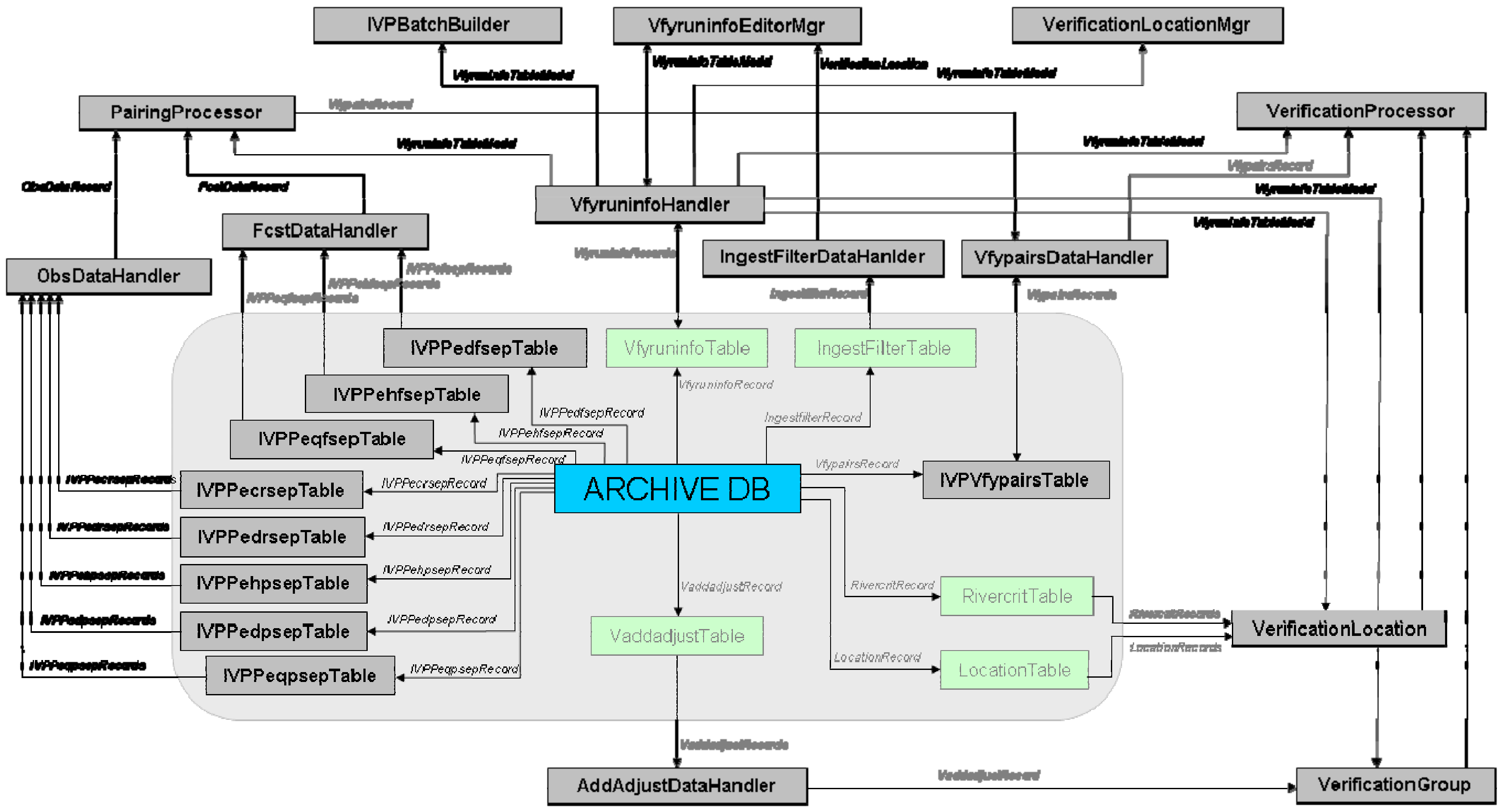


Diagram 6

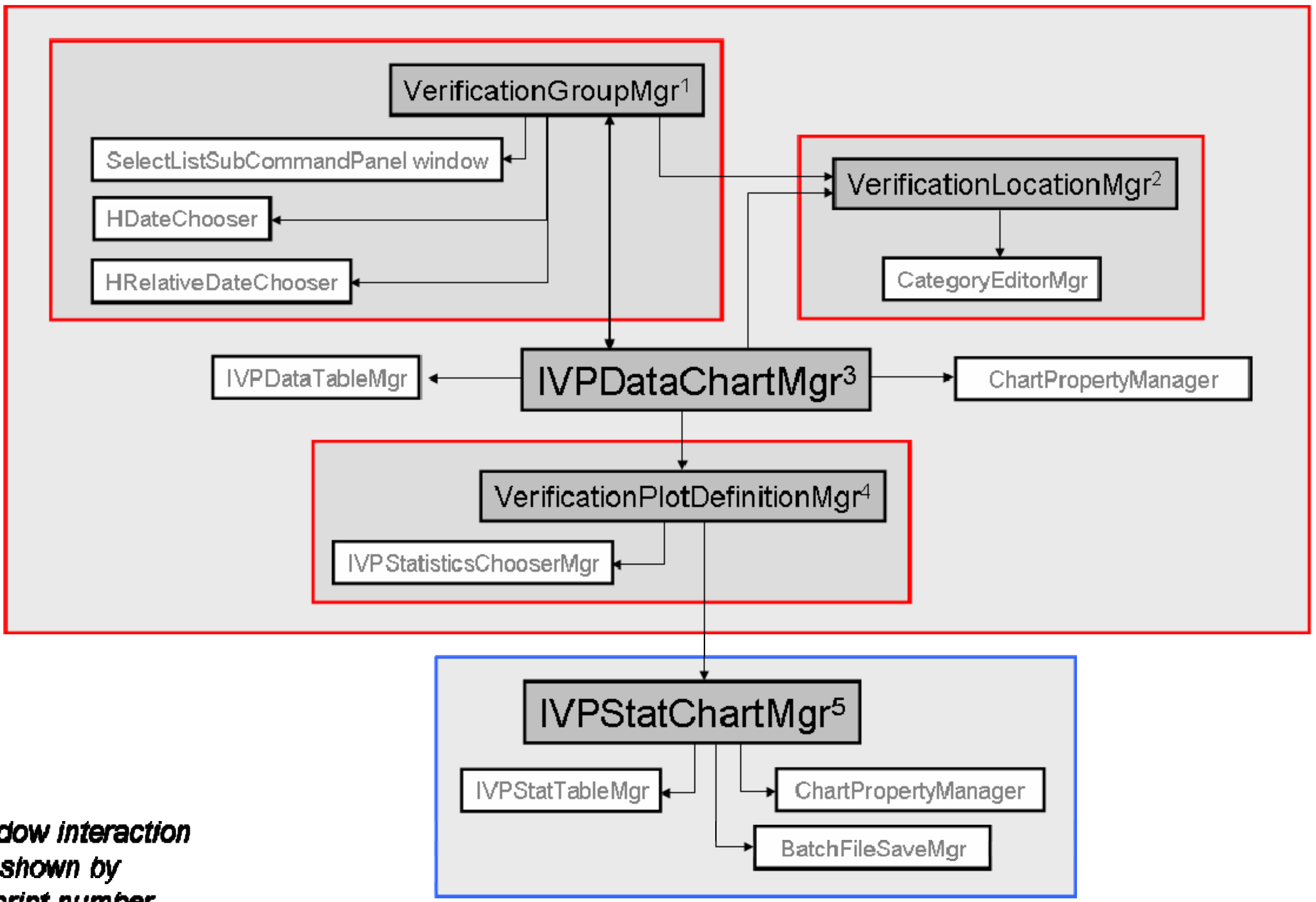
Data Flow Diagram (Proposed for OB8.2)



Data is transferred via instances of data classes. The class used for transfer is provided next to each line and given in *italics*. If the class is part of the automatically generated library, then it is in *italics*.

Diagram 7

Window Interactions



Order of window interaction
For users is shown by
The super-script number
At the end of GUI name.

Diagram 8