AFRL-RI-RS-TR-2010-156

INTERACTIVE VISUALIZATION OF NATIONAL AIRSPACE DATA IN 4D (IV4D)

Aerospace Computing, Inc. (ACI)

August 2010

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88th ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2010-156 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/s/
PETER A. JEDRYSIK
Work Unit Manager

/s/
JULIE BRICHACEK, Chief
Information Systems Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
AUGUST 2010	Final	May 2007 – May 2010

4. TITLE AND SUBTITLE

INTERACTIVE VISUALIZATION OF NATIONAL AIRSPACE DATA IN 4D (IV4D)

5a. CONTRACT NUMBER
FA8750-07-C-0050

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
N/A

6. AUTHOR(S)

Susan Hinton

5d. PROJECT NUMBER
NASA

5e. TASK NUMBER
NG

5f. WORK UNIT NUMBER
01

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Aerospace Computing, Inc. (ACI)
465 Fairchild Drive, Suite 224
Mountain View, CA 94043-2251

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL/RISB
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
N/A

11. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFRL-RI-RS-TR-2010-156

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. PA# 88ABW-2010-4567
Date Cleared: 23-Aug-10

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This Final Technical Report discusses the accomplishments of an effort to support NASA and FAA goals for visualization of the National Airspace System (NAS) to aid the analysis of proposed changes to increase its capacity and meet future needs. The Air Force Research Laboratory (AFRL) is developing a JView-based visualization tool, known as the Viewer, for visualizing objects and results from NASA's ACES (Airspace Concept Evaluation System) simulation application. IV4D (Interactive Visualization in 4-D) was developed as a plug-in to the Viewer. It employs a combination of heuristics and visualization defaults in order to search for entities to visualize, and then to display them with minimal user selection. IV4D makes it easy for aeronautics researches and others to display flight trajectories, airspace boundaries, weather, and more, with data coming from multiple sources, including multiple software and simulation applications. IV4D is designed to work with most forms of NAS state data, concentrating first on data representing the most common elements of objects within the NAS – positioned still and moving objects; that is, concentrating on objects situated by latitude, longitude, elevation, which are present for a defined period of time.

15. SUBJECT TERMS

National Airspace System visualization, airspace visualization, air traffic visualization, air traffic management tools, airspace analysis tools

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Peter A. Jedrysik
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UU	42	N/A

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

ABSTRACT

When AFRL (Air Force Research Laboratory) offered to use part of a Congressional earmark, targeted at advancing the Air Force's JView visualization framework, to consolidate JView capabilities into an API (application programming interface) layer, now known as the Viewer, for visualizing objects and results from NASA's ACES (Airspace Concept Evaluation System) simulation software application, the idea of a user centered tool for studying both the current and theoretical use of the NAS (national airspace) was born. Though IV4D (Interactive Visualization in 4-D) was first conceived of as a way to facilitate the display of data from NASA's simulation software application, aeronautics researchers preferred a more generic approach. Design for IV4D evolved to locate and display two readily assembled types of table data, comma-delimited and SQL (structured query language), representing positioned still and moving objects – that is, objects situated by latitude, longitude, and elevation which are present for a defined period of time. IV4D also needed to be easy to understand and use, and able to visualize a multitude of airspace polygons along with thousands of flights at a time. Additionally, it was envisioned that user filtering and specifications be saved as a kind of meta-data so that subsequent attempts to display similar future data would be accelerated.

Today IV4D is a flexible, easy to understand and use tool that, along with the AFRL Viewer, performs well with a high volume of data, and which provides NASA and FAA researchers with a way to see and understand still and animated NAS data coming from multiple sources.

ACKNOWLEDGEMENTS

In addition to AFRL, Aerospace Computing, Inc. (ACI) would like to thank developers from CACI, Inc. who developed the Viewer API layer that supports IV4D, and with whom the ACI team coordinated during the project. ACI would also like to thank NASA Ames Research Center for providing office space and facilities for the duration of the project, and for providing beta and usability testing while using beta and released versions of the tool for research purposes.

TABLE OF CONTENTS

LIST OF FIGURES

1 SUMMARY

IV4D (Interactive Visualization in 4-D) consists of a set of tools that plug into an API (application programming interface) layer, called the Viewer, which, in turn, utilizes the AFRL (Air Force Research Laboratory) JView graphics engine. All of the software, IV4D/Viewer/JView, is written in Java and is platform independent, meaning that it is able to run on many computing platforms. IV4D has been tested on computers running Linux, Windows, and OS X operating systems.

IV4D was first conceived of as a way to display data from NASA's ACES (Airspace Concept Evaluation System) simulation software application. However, aeronautics researchers made it clear that there are many sources of NAS (national airspace) data, and an application able to assemble displays from disparate sources would be valuable.

IV4D employs a combination of heuristics and visualization defaults in order to search for entities to visualize, and then to display them with minimal user selection: IV4D makes it easy for aeronautics researches and others to display flight trajectories, airspace boundaries, weather, and more, with data coming from multiple sources, including multiple software and simulation applications. To do this, IV4D is designed to work with most forms of NAS state data, concentrating first on data representing the most common elements of objects within the NAS – positioned still and moving objects: That is, concentrating on objects situated by latitude, longitude, elevation, which are present for a defined period of time. Such objects include airports, flights, waypoints, airspace boundaries, weather polygons, etc. However IV4D is not limited to this data, as it is possible to add as many other relevant pieces of data, assembled with or connected to the state data, as desired. It is also possible to add in, on the fly, algorithmic metrics that act on the data to produce a result for display.

Since ACES data originally consisted solely of table data, and since many aeronautics tools read and display table data, and since table data is comparatively easy to assemble, IV4D and the Viewer were designed to work with forms of table data. (Both also work with a small set of other types of data.) As a result, IV4D works with both comma-delimited (CSV) and SQL (structured query language) database data (MySQL), and does not depend upon any single software application. Additionally, it was envisioned that user filtering and specifications be saved as a kind of meta-data so that subsequent attempts to display similar future data would be accelerated.

Throughout the project there were three main challenges. First, there is a linear set of API (application programming interface) dependencies as the Viewer depends directly upon JView and IV4D depends directly on the Viewer, so an obvious challenge involves keeping the APIs for the applications stable and in synch with each other.

Second, IV4D was developed on the West Coast, at NASA Ames Research Center, where key users live and work, while the Viewer and JView were being developed on the East Coast, in Rome, NY at the Air Force Research Laboratory. This meant that planned communications and at least a modicum of CMMI (Carnegie Mellon's Capability Maturity Model Integration) process, targeted at levels 2 to 3, for software development, were necessary. This contrasts with regular software development at AFRL, which adheres to a deliberately impromptu process known as agile software development. Agile development assumes constant communication and access among developers (i.e., working in close proximity).

Third, with the release of version 1.0, at the end of the first year of development, it became clear that software performance was a major impediment to successfully displaying a multitude of airspace polygons and thousands of flights at a time, which are necessary in order to show flight activity across the entire NAS.

Challenges 1 and 2 were closely intertwined and were mitigated during the first year by attempting to follow a streamlined version of NASA's ACES software development process, though following the process came more naturally to developers on the West Coast. In following years, once the Viewer API stabilized, and after East and West Coast developers became more at ease with each other and with the combined software set, it was possible to ease back from strict adherence to process. The timing was fortunate, since solving performance issues for all pieces of the software demanded the attention of the developers on their respective parts of the puzzle.

During the past three years, a number of researchers and aeronautical research developers used both in-development, and released versions of IV4D. Their feedback was essential in making a successful final product. In addition to reporting bugs, their usability and feature requests regularly changed the course of development. These included changes such as: Allowing users to pick the same data source as many times as desired while applying different filters and display specifications; Keeping the last visualization display up while the user creates and starts the next one; And, of course, the wringing out of every possible iota of performance, to meet the explosive growth in aeronautical data begging for display (such as airspace and weather polygons together). These are only of few of many changes made as a result of user testing.

Today IV4D, in conjunction with the AFRL Viewer, forms a successful set of software tools that are being used by aeronautics researchers at NASA Ames and NASA Langley Research Centers. Users have found the application to be flexible, easy to use and understand, that performs well, and provides researchers with a way to study and understand still and animated NAS data. Furthermore, the IV4D/Viewer combination was considered interesting enough to cause NASA Ames Research Center to fund a separate plug-in module for the Viewer, the ACES simulation run-time plug-in.

There is, however, room for product improvement. For example, a considerable improvement would involve standardizing IV4D and Viewer persistence data so users can move more seamlessly between visualizations saved from the Viewer Configuration panel and visualizations saved from IV4D. Other improvements might include making parts of IV4D modules more malleable with respect to each other, improving heuristics for default display results, and new default methods for visually comparing and contrasting sets of data.

2 INTRODUCTION

2.1 IDENTIFICATION AND INTENDED AUDIENCE

This is the final technical report for IV4D (Interactive Visualization in 4-D), a visualization tool that plugs into a layer of software called the Viewer, which consolidates the Air Force's JView graphics engine capabilities into an API layer for visualizing the NAS (national airspace).

It is intended to be read by any software or technical manager or individual interested in 1) building a user centered software application to visualize elements and objects in the NAS, or 2) the experience of working across agencies to build an application for use by researchers at one agency (i.e., NASA) that depends upon a framework being built by another agency (i.e., the Air Force).

2.2 IV4D APPLICATION BACKGROUND AND OVERVIEW

IV4D is a visualization tool, built to leverage and enhance work done for the Air Force Research Laboratory (AFRL) Viewer, and developed over a three-year period for NASA Air Traffic Management (ATM) researchers. The goal was to produce a tool that is easy to understand and use, performs well, and provides researchers a way to study and understand the national airspace using various forms of input and output data, such as data from NASA's ACES (Airspace Concept Evaluation System) simulation software.

Though first designed to read ACES data, IV4D is built to work with simple forms of table data, both comma-delimited (CSV) and SQL database (MySQL) data, and does not depend upon any single software application. IV4D also makes it possible to visualize flight trajectories, airspace boundaries, weather, and more, with data that comes from multiple sources, including multiple software and simulation applications. Finally, it was designed so that user filtering and specifications are saved as a kind of meta-data, called "views," so subsequent attempts to display similar data can be accelerated.

In order to do this, IV4D is designed to work with most forms of NAS state data, concentrating on data which represents positioned still and moving objects – that is, on objects which are situated by latitude, longitude, elevation, and which are present for a defined period of time. These objects include airports, flights, waypoints, airspace boundaries, weather, etc.

Throughout the project, all software upon which IV4D depended was evolving while IV4D evolved: IV4D modules, Create Views and Apply Views, plug into the AFRL Viewer while the Viewer depends upon another actively evolving piece of software, the AFRL JView graphics engine. Additionally, NASA's ACES software application was actively being developed. Changes to ACES included changes to data in form and type. (Originally IV4D targeted display of ACES data.) Plus, IV4D was developed on the West Coast, at NASA Ames Research Center where key users live and work, while the Viewer and JView were being developed on the East Coast, at AFRL in Rome, NY. Furthermore, AFRL utilizes an agile software development process, known to work best when developers work in close proximity. When developers belong to different organizations and are not sitting near each other, agile methodologies become more difficult to follow. Thus, in addition to technical challenges entailed in synchronizing design in order to produce useful software, it was important to foster regular communication and synchrony by defining a process to formalize steps in design and implementation.

2.3 DOCUMENT OVERVIEW

The bulk of this document, below, is divided into two parts:

1. The technical description of IV4D and how it evolved

2. A description of the organizational framework designed to synchronize IV4D and the Viewer, and how that evolved

Following these are sections discussing results and conclusions.

3 TECHNICAL METHODS, ASSUMPTIONS, AND PROCEDURES

IV4D is a user-centered piece of software, designed for use by aeronautical researchers and consumers of aeronautical data. It allows users to quickly take advantage of esoteric and complex functionality supplied by underlying layers of AFRL supplied software. It does this first by hiding, to a great degree, the complexity upon which it relies, and second by creating its own intermediary layer, based on a series of heuristics, to call upon and use the underlying software.

To build on this statement, IV4D is a set of plug-in modules whose forte relies primarily upon understanding the composition of aeronautical data – or at least a broad swath of it – and on familiarity with aeronautical research at NASA, and familiarity with and access to researchers and aeronautical developers. It does not strongly rely upon software design breakthroughs or on cleverness in manipulating scene graphs, compos able software elements, or underlying 3-D to 2-D transformative algorithms. These last areas are well advanced by the AFRL JView graphics engine and by the AFRL Viewer.

One can think of IV4D as second generation software built on top of a set of premier first generation visualization capabilities. IV4D is a highly flexible tool that allows aeronautical researchers to quickly and accurately visualize aeronautical state data and associated elements, while, if desired, employing on-the-fly metrics.

3.1 ARCHITECTURAL VIEW

Below (Figure 1) is a view of the basic architectural design showing how IV4D makes use of AFRL's JView software application capabilities. JView is a graphics engine, invented by AFRL that allows rapid development of highly accurate 3D and 4D platform independent visualizations. The Viewer forms a subsequent layer, consolidating JView capabilities into an API for visualizing objects in the NAS. The Viewer also employs a plug-in framework, allowing independent modules, such as the IV4D modules Create Views and Apply Views, or the separate ACES run-time visualization module, to plug into and to utilize collected functionalities.

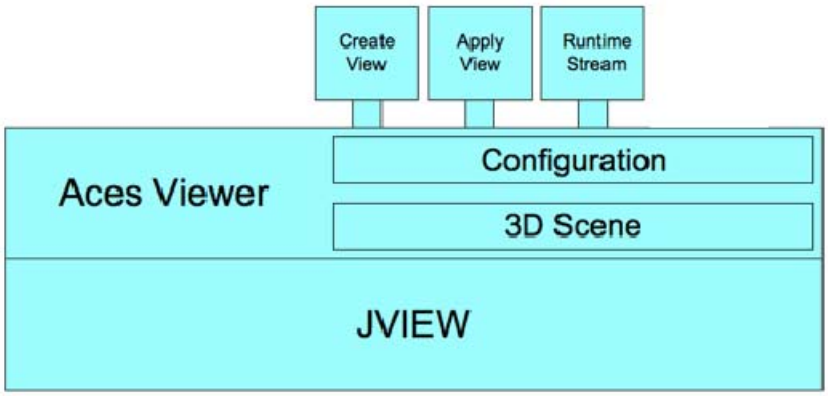


Figure 1: Architectural view

3.2 DATA

IV4D depends upon aeronautical data, and this section describes the data that IV4D utilizes. Specifically, IV4D looks for data that can be visualized, and then uses a series of heuristics for deciding how to display the data.

Though IV4D was first conceived of as a way to facilitate the display of data from NASA's ACES simulation software application, aeronautics researchers preferred a more generic approach. Design for IV4D evolved to locate and display two readily assembled types of table data, comma-delimited (CSV) data and SQL (structured query language) data, representing positioned still and moving objects – that is, on objects situated by latitude, longitude, and elevation that are present for a defined period of time.

In addition to the general forms described below, IV4D recognizes XML ACES airspace boundaries, and a more specialized ACES SQL table containing airport location.

3.2.1 CSV Data

IV4D/Viewer reads comma-delimited data in two static forms and in one dynamic form. Though originally the forms represented sector or center airspace volumes, data in these formats have been used to represent sub-sectors, dynamic sectors and sub-sectors, and dynamic weather events.

All CSV data must have a commented Header line at the start of the file (Figure 2). The Header represents the table schema and, for each object being represented, all forms of CSV data must include

1) A unique Id string

2) A base altitude and a top altitude

3) A space-delimited set of latitude/longitude vertices that describe a closed polygon

4) If animated, there must be a column for time, in milliseconds, seconds, minutes, or hours, in regular, incremented intervals.

   Since the scene is redrawn at every time interval, each maintained volume must have a row for every time step at which it is visible. In other words, if a polygonal shape is present for 4 hours and time advances every minute, there will be 240 rows (60 minutes x 4 hours). Id will be maintained, but base altitude, top altitude, and the closed polygonal shape can change at each time interval.

5) Other, optional, columns may also be present, and the optional data can be included in the display. Viewer specific functionality makes this possible. Once the scene is drawn, users can choose to use Viewer specific functionality either to display aspects of the additional data or to script on-the-fly code snippets In order to manipulate the data for display.

```
**Sector(string id),Time,AltitudeMin(feet),AltitudeMax(feet),Boundary(lat/lon),
2974,1153999230000,0,25294.827600192595,39.3386/-87.6004 39.3386/-87.5070 39.3386/-87.4603 39.3386/-87.4136 39.3386/-87.3670 39.3386/-87.3203 39.3386/-87.2736
```

Figure 2*:* First two rows from file with animated polygonal data

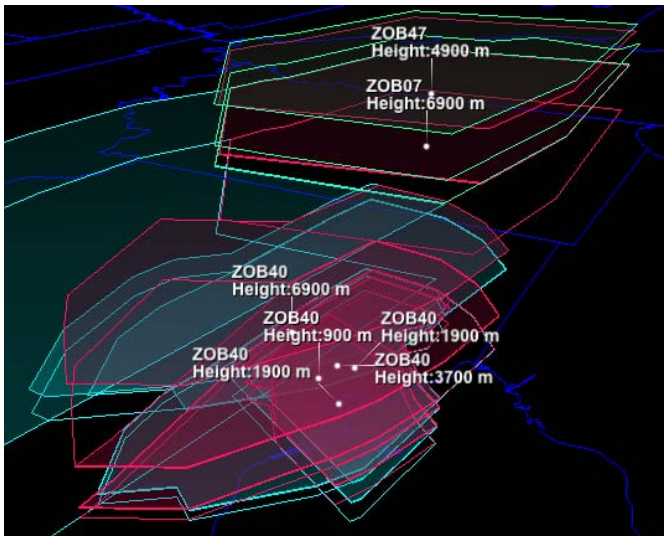


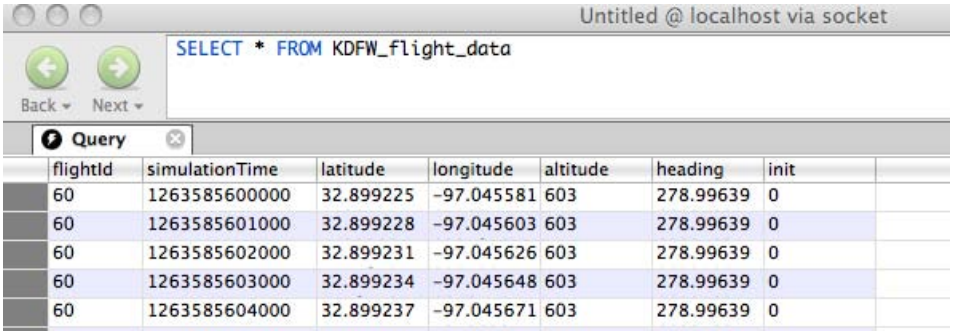
Figure 3: Display of CSV data, with 'extra' data for height shown in labels

3.2.2 MySQL Data

IV4D/Viewer reads MySQL database tables containing columns for flightId (or other string ending with the sub-string of 'Id' or 'Name'), simulation Time, latitude, longitude, altitude, init (Boolean), and heading (an optional column). As with CSV data (Figure 2) there may be other columns as well.

Additional columns, by default, are ignored, though users can choose to employ functionality provided either by IV4D or by the Viewer to access and either display aspects of 'extra' columns or to add on-the-fly code snippets to manipulate data for display. IV4D allows users to filter and manipulate the data through SQL queries, regular expressions, or simple listings while, along with other mechanisms, the Viewer allows users to add short code snippets to manipulate the data during visualization. More is written below on the user interface in section ***User Interface***.

As with CSV data, MySQL data is assumed to represent either static or animated objects. Static objects are drawn as lines, though lines can be connected to represent runways, taxiways, etc., or entire trajectories or other objects. Animated objects are assumed to be flight objects and are represented by models resembling paper airplanes, though users can use functionality provided by the Viewer to select or draw alternate shapes.



flightId	simulationTime	latitude	longitude	altitude	heading	init
60	1263585600000	32.899225	-97.045581	603	278.99639	0
60	1263585601000	32.899228	-97.045603	603	278.99639	0
60	1263585602000	32.899231	-97.045626	603	278.99639	0
60	1263585603000	32.899234	-97.045648	603	278.99639	0
60	1263585604000	32.899237	-97.045671	603	278.99639	0

Figure 4: Table columns and rows needed to animate trajectories (heading is optional)

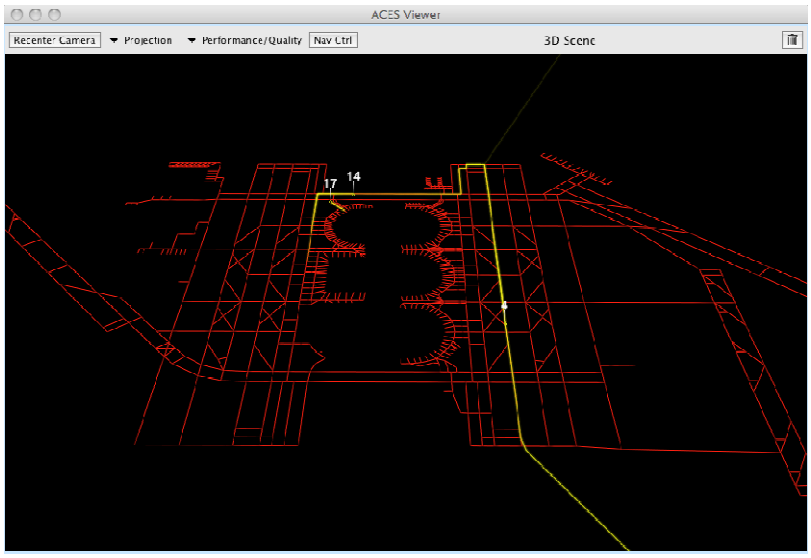


Figure 5: Display of both static and animated MySQL table data, together

3.2.3 Heuristic Data Search

Because locating data is a repetitive and predictable process, which therefore can be automated, as much as possible IV4D tries to locate viewable table data for the user. (The goal is to relieve users of unnecessary tedium.)

Viewable data contains data for columns mentioned previously, in sections ***CSV Data*** and ***MySQL Data***. In some circumstances, IV4D goes further, looking for the most appropriate Id or attempting to associate appropriate latitude, longitude, and altitude columns with each other when there are multiple choices.

Heuristics are particularly helpful when looking at SQL data, since a database can contain tables with and tables without viewable data.

IV4D looks for ‘the most likely’ Id candidate column name in the following heuristic order (an Id name consists of an alpha-numeric string):

1) flightId

2) Anything containing ‘flightId’, such as ‘uniqueFlightId’ or ‘etmsFlightId’ or ‘airlineFlightId’ (pick the first found, if there is more than one)

3) etmsId

4) airlineFlightNumber

5) The first column name that ends in “id”, such as “AIRPORT_ID” or “airportId”

6) The first column name that ends in “name”, such as “fixedWIngName”

The other columns that may have a varying column name are: column names indicating latitude, longitude, or altitude. Each should include the appropriate case-ignored sub-string of ‘latitude’, ‘longitude’, or ‘altitude’, for example IV4D would find all of the following: reference_latitude, altitudeFeet, westernHemiLongitude. If there is more than one column with appropriate sub-strings, IV4D tries to choose those which seem to go together – for example if there is a latitude_1, latitude_2, longitude_1, longitude_2, IV4D would, by default, use latitude_1 with longitude_1. However, the user interface (UI) allows users to select them differently.

In addition to the rules above, IV4D utilizes other heuristics specifically for ACES data. For example, an Id may be connected to latitude, longitude, altitude columns spread across several tables loosely coupled with each other through ACES table naming conventions.

3.3 USER INTERFACE (UI)

Though IV4D tries, as much as possible, to automatically find and display data, there are decisions that remain best made by a user. The IV4D user interface is the mechanism by which users make these decisions.

Researchers working on difficult problems are time constrained. Therefore a key goal for IV4D was to make it as simple as possible to quickly produce a useful visualization – defined as one that may not be as polished as possible but which clearly illustrates desired elements in as esthetically pleasing a way as possible. This meant IV4D needed a user interface requiring 1) little time to learn, but which 2) allows users to pick, choose, save, and reuse data as desired, and 3) which displays immediate results that can be fine-tuned afterward, i.e., interactively.

While aeronautical researchers use a number of software applications, including MATLAB, SPSS, scripting languages, and others, nearly everyone uses Microsoft Excel: The main point is that researchers are used to looking at data in rows and columns, and at applications displaying data in rows and columns. Where reasonable, therefore, IV4D presents data selection and filtering mechanisms in lists or with rows and columns. Other selection and filtering UI elements were designed after considering and choosing the most straightforward standard UI practices: Possibly this makes the UI look somewhat boring (Figures 6 and 7), but it allows users to create useful visualizations very quickly – often on the first attempt.

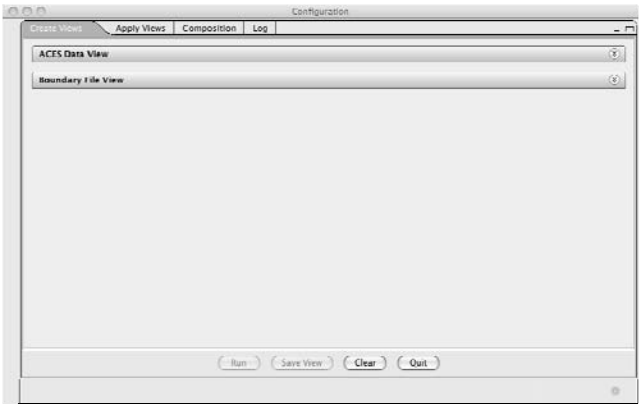


Figure 6: Initial UI on start-up, shows empty Create Views panel

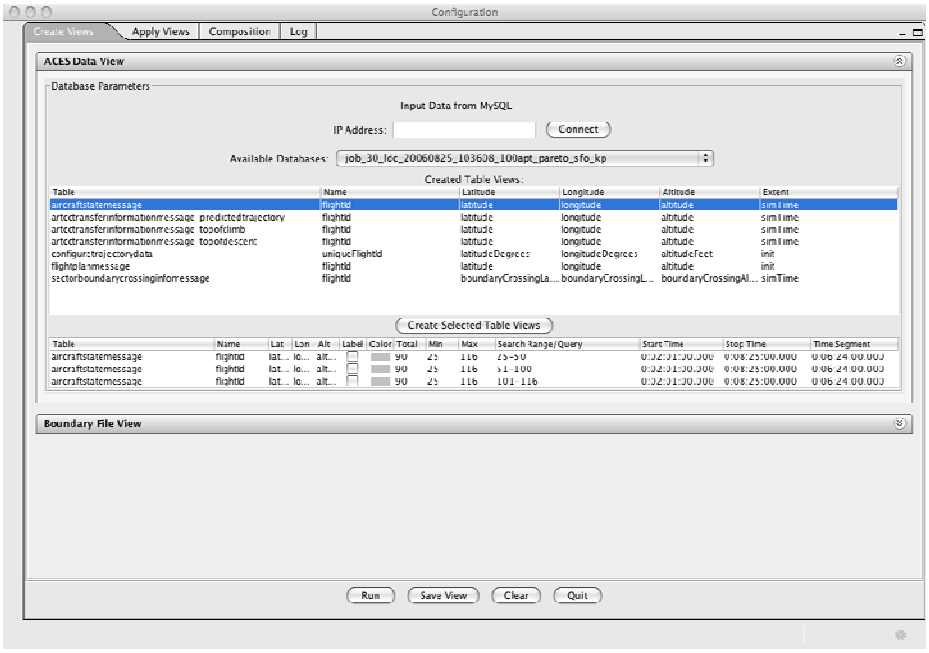


Figure 7: UI for building visualization from MySQL table data

IV4D UI is split into two modules: Create Views, for creating visualizations, and Apply Views, for reusing visualizations and for mixing visualizations together.

3.3.1 Create Views

The Create Views UI, which allows users to build visualization, is divided into two parts: One for building visualizations of airspace volumes, such as sectors or weather, and one for building visualizations of flights, trajectories, and for line drawings, such as airport taxiways, runways, etc. Visualization can be built from either or both parts.

3.3.1.1 Airspace Volumes

Once appropriate CSV or ACES XML airspace boundary files are selected from a standard Java File Chooser dialog, users can add data from each file one or many times, and then filter, label, and choose colors for each data row.

For example to see ***Figure 8***, below, a user would

1) Select low sector data twice

2) Filter for two sets of sectors, one set with Id names starting with “ZID,” the Indianapolis ARTCC (Air Route Traffic Control Center), and the other set from ARTCCs surrounding ZID

3) Give separate colors to each set

4) Turn on labels for ZID

5) Click Run

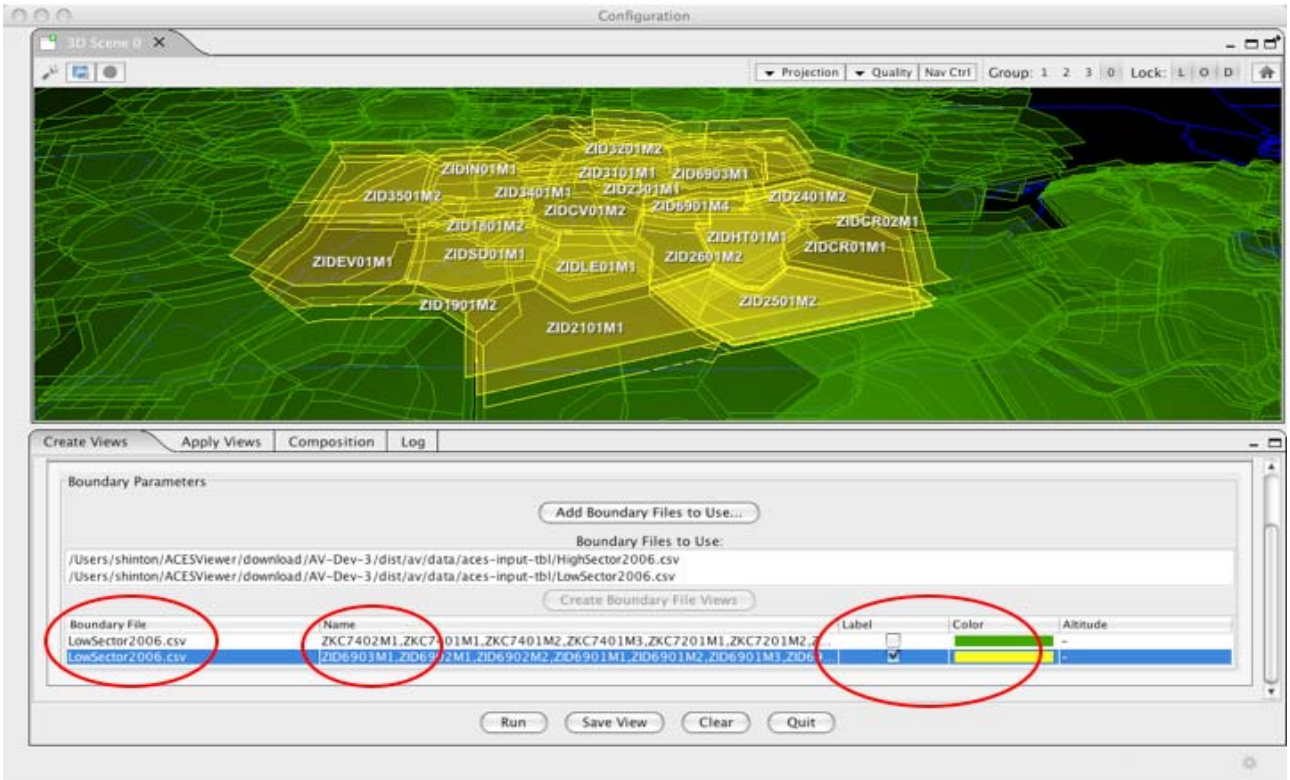


Figure 8: Simple selection yields fast, informative display

To choose a filter or color, the user clicks the cell in the row and column to filter (see ***Figure 9***, below). For example, to pick the yellow color, a user clicks on the 2nd cell down from the column header Color; or to select a set of Id name filters, the user clicks on the appropriate cell under the column header Name.

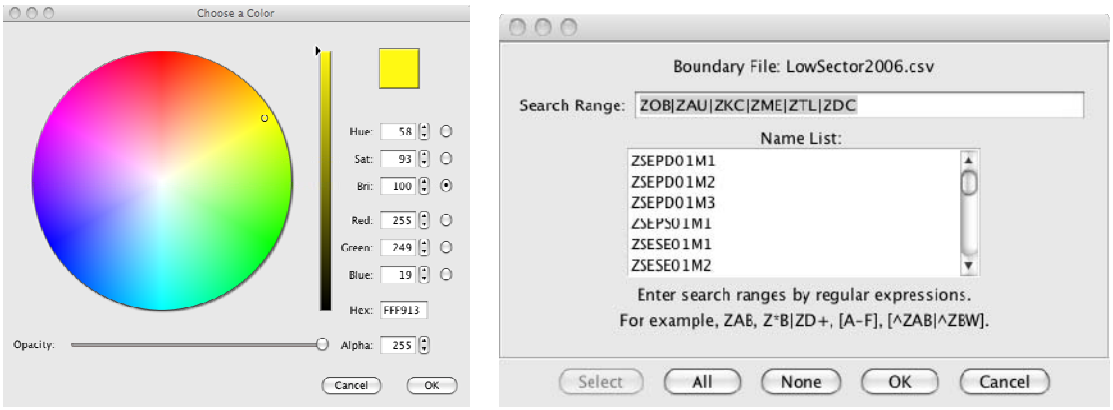


Figure 9: Popup dialogs for coloring or filtering sectors

Most popup dialogs have multiple ways for users to refine their process. For example, in the Boundary File popup above, a user can enter a regular expression into the Search Range text box, or a user can scroll through the list of Id names and click on those to display. This helps to ensure that every user, for example users who are not familiar with regular expressions, will be able to make a useful selection.

3.3.1.2 Flights, Trajectories, Line Drawing

The first step in selecting table data from a database is to connect to the database server and then to select a database. If the database server is on an IV4D user's system, the user simply clicks Connect; otherwise a user enters an appropriate IP Address and clicks Connect.

IV4D then searches for tables in the database that contain the appropriate, necessary columns – see previous section ***Heuristic Data Search*** for more information. As with earlier CSV data, users can add data from each table one or many times, and then filter, label, and choose colors for each data row. In addition, for animated data users can filter by Start and Stop times.

Since there are more columns available in IV4D for MySQL data, users are able to click on more cells and, when there are other appropriate choices, filter more finely.

Normally, as in the example in Figure 10, under column header Name, one would expect to see 'flightId', but here a user has clicked on the cell and made an alternate selection, 'sectorName'. Then, after clicking on another cell under column header Search Range/Query, the user has entered the following regular expression: ^ (ZOB|ZAU|ZKC|ZME|ZTL|ZDC) This expression causes flights that are in sectors beginning with particular ARTCC identifiers (such as 'ZOB') to be selected.

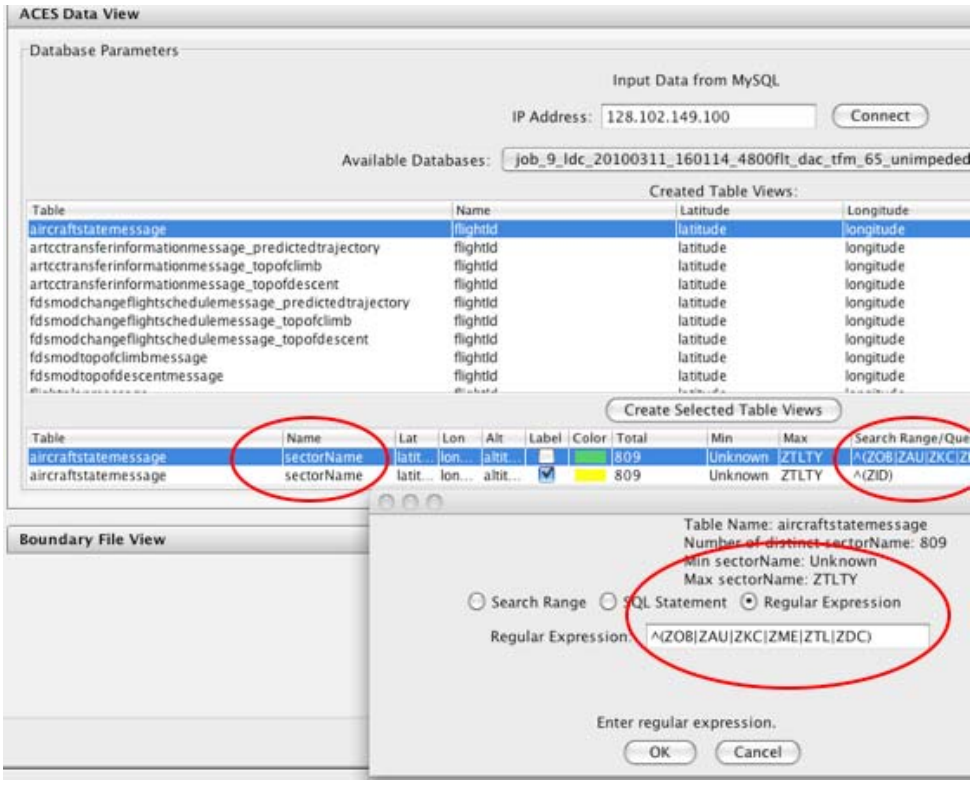


Figure 10: Selecting alternate Name, sectorName, along with regular expression: ^ (ZOB|ZAU|ZKC|ZME|ZDC)

Using the Search Range/Query popup, users can create filters in the most appropriate mode for the task at hand. For example using a SQL statement allows a user to include data from other columns, or other tables – including those with non-visualizable data. The Search Range option is best when a user knows exactly what he or she wants to see, for example 'ZID78 – ZID99.'

In the example above, which results in the display seen in Figure 11, flights appear with green or yellow tails depending upon whether they are currently in ZID or in a surrounding airspace. This is similar to an earlier example showing a result from using regular expressions for airspace volumes, along with color (see above section ***Airspace Volumes***).

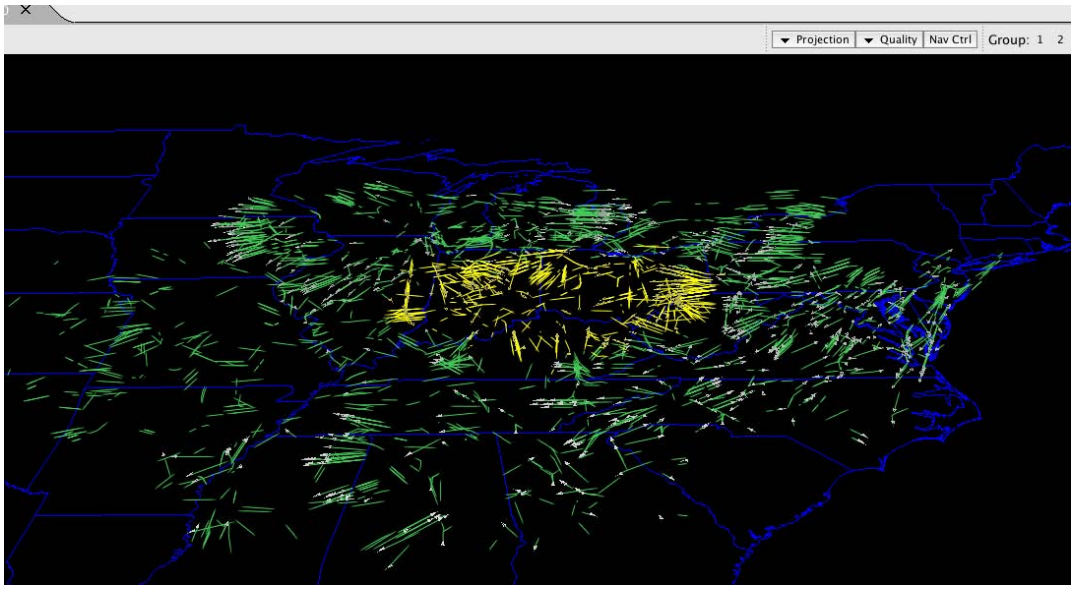


Figure 11: Result from using regular expressions with MySQL flight data

And, of course, users can select flight data and airspace volumes together. Selections shown in Figure 12 result in the display pictured in Figure 13.

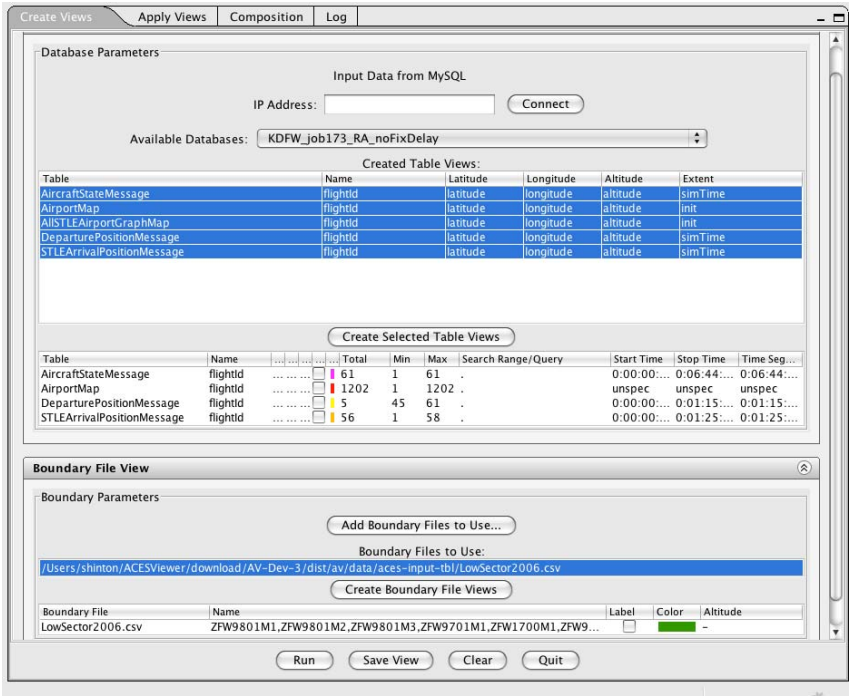


Figure 12: Select low DFW sectors with flights into and out of Dallas-Fort Worth

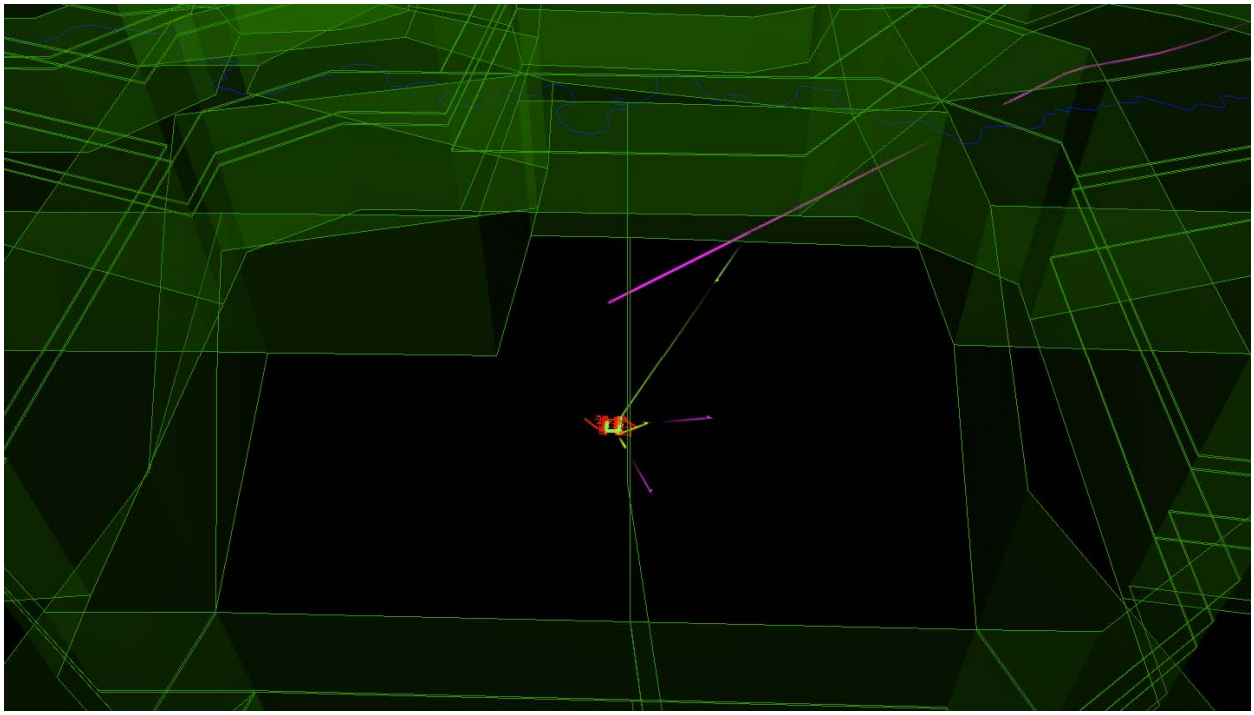


Figure 13: Low DFW sectors with flight tracks into and out of Dallas-Fort Worth

3.3.2 Apply Views

Data saved from the Create Views panel can be replayed or combined with other saved data from the Apply Views panel. In addition to combining visualizations, Applied Views allows users to review settings made in Create Views, and to apply those settings to other data with a similar scope.

A simple example is shown in Figure 14. From Create Views, a user builds and saves visualization. In the case below, flights (from database “job_1_ldc…”) with Id names starting with the number 4 are shown in red, while flights with Id names starting with the number 5 are shown in yellow. Labels are turned on.

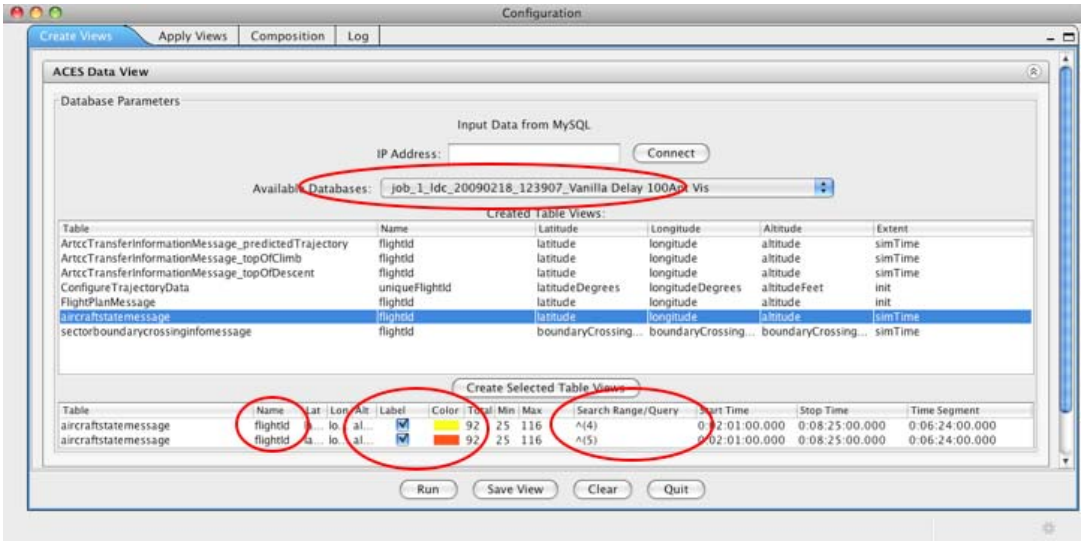


Figure 14: A user creates, and then saves a view, so that flights with Ids starting with '4' are red and flights with Ids starting with '5' are yellow

Later, from Apply Views (Figure 15), a user picks and loads the view saved above, from a list. The Apply Views UI is designed to echo the Create Views look, making it easy for users to recall the settings. If desired, a user can 'Browse' to apply the view to a new database.

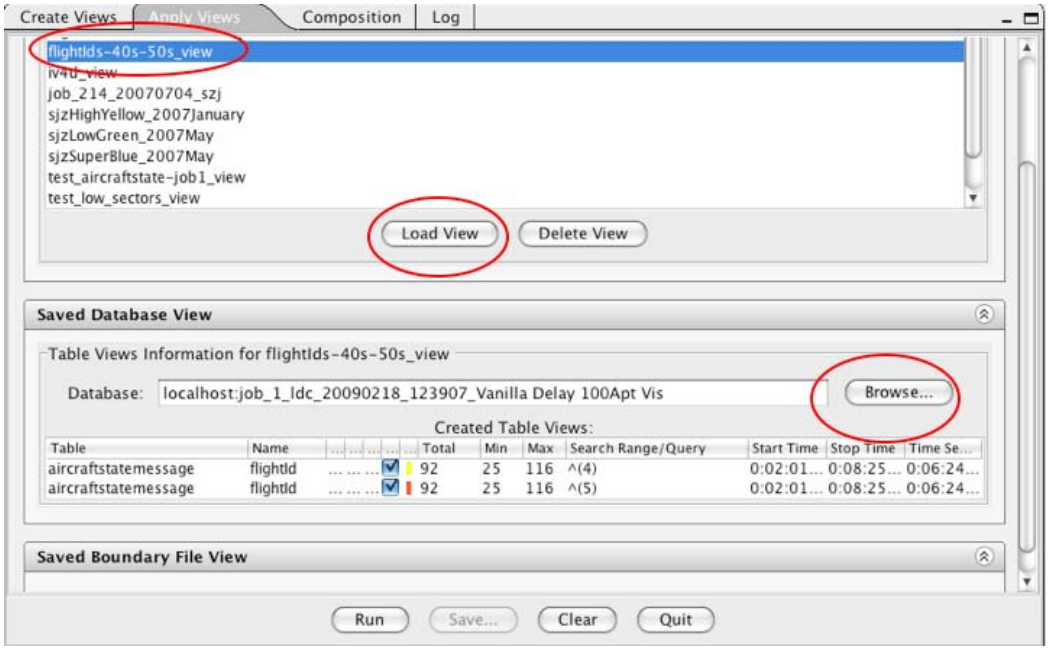


Figure 15: Apply Views shows saved view, from Figure 14

If the new database contains appropriate data, a user can save, or save and immediately run, the new view (Figure 16). In the case below, the user applies the view created for the original database to a new database named “job_9_ldc…” (The original database started “job_1…”)

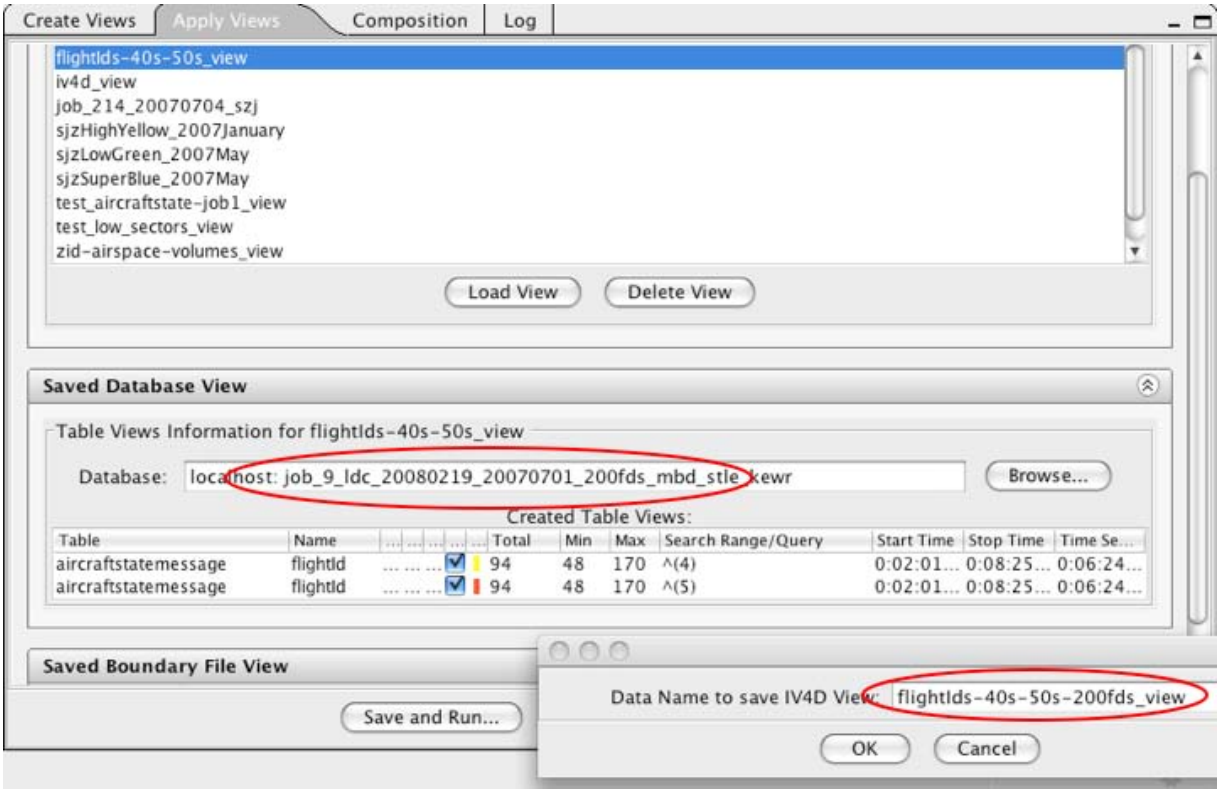


Figure 16: User applies view to a new database, and saves result to a new name

This makes it easy to quickly use the same settings over and over again for, say, a series of related simulations, or for simulations that may not be at all related.

A user can also use Apply Views to put two or more views together. The picture in Figure 17 shows the result from putting together the flight Ids view with an earlier airspace volumes view (see earlier section ***Airspace Volumes***). This means that data from entirely differing sources can be quickly visualized together. For example a user might choose to show animated weather polygons from one day or one experiment with flight data from an entirely different source or time.

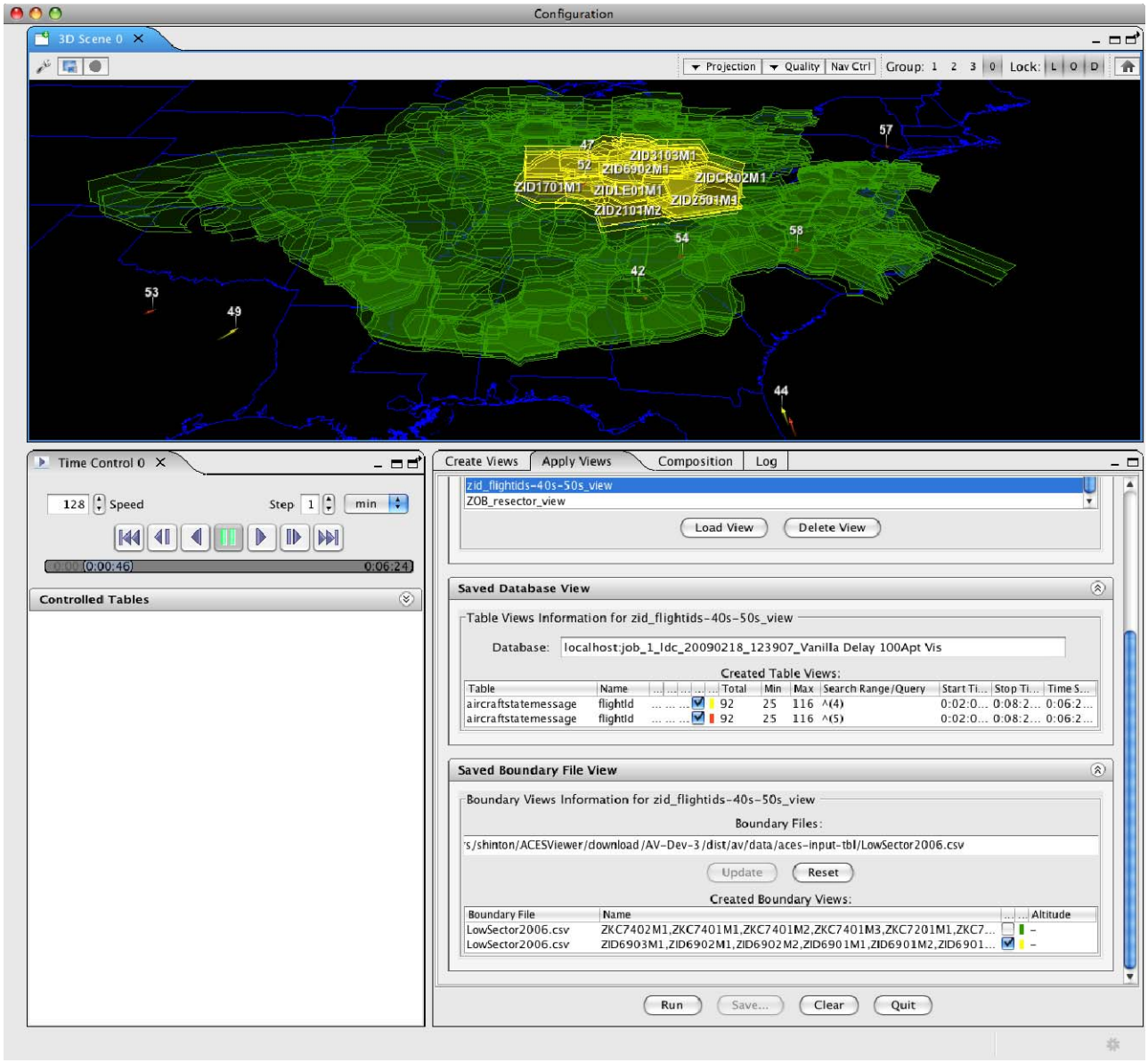


Figure 17: The flight Ids view is combined with an earlier airspace volumes view

3.4 DISPLAY HEURISTICS

The JView graphics engine allows nearly unlimited visual configurations, while the Viewer allows users to set hundreds of display options.

To help users of IV4D obtain fast results, IV4D allows users perhaps a dozen ways to filter objects to visualize (e.g. by name, using regular expressions, using SQL queries, by time interval, etc.), but deliberately offers next to no display options: Users can turn labels on or off, and users can pick colors for visualized objects. This is done for two reasons:

1) Aeronautics researchers are not graphic artists, and most have no desire to become graphic artists. They want to focus on their research and their research results.

2) Aeronautics researchers want to see something fast. If it takes too long to produce a result or otherwise takes a researcher away from research, he or she is not likely to keep using the tool.

This is not to say that researchers do not want to see a visual display or that they do not want graphically good pictures. The opposite is true: They often make statements saying they want clear, accurate, and easy ways to view displays. But they do not want to spend time making them.

Of course, because a “good” or a pretty visual display often has a number of refinements necessary for the display to be considered pleasing or easy to view, this presents a challenge for the software developer. IV4D attempts to meet this challenge by making assumptions about what the user wants to see, given the data that the user has selected. In other words, IV4D employs a set of heuristics to try to match a reasonable visual display to the data.

Essentially IV4D employs two types of display heuristics. One is global, and is applied as a default for every display launched by IV4D: Because IV4D assumes that researchers are interested in NAS-wide flight behavior, i.e., airspace volumes and flights over the continental U.S., IV4D always includes a map showing continental U.S. state boundaries. IV4D also exaggerates altitude by a scale of 10:1, displays animated flight objects using a model that resembles a paper airplane, adds a 5 minute history trail behind each animated flight object, specifies label height and font, and uses a graduated pin to connect the label to a flight or to a waypoint. Complete trajectories are drawn as solid lines that connect waypoints along a trajectory’s path. The default playback time-step is 1 minute. Airspace volumes are lit, have translucent fill, solid lines, and completely transparent shelves. There is more, but mainly the idea is to assume that the best view is of the forest rather than the trees.

The second set of heuristics attempts to decode rows of MySQL data: As mentioned earlier, animated objects must have unique Ids. For animated objects there should be a single row at each appropriately incremented time interval. Otherwise, when an object Id appears in multiple rows, all with an identical time, it’s assumed that the collected set should be represented as a trajectory. Sometimes there are odd cases, with object Ids sometimes appearing multiple times and at other times appearing in single instances. In these cases, IV4D attempts to resolve the differences, by drawing each instance as a sphere, as though each point was a waypoint. Finally, IV4D recognizes one ACES specific database table that pinpoints a single latitude and longitude point for an airport. These are drawn as airport towers.

3.5 DISPLAY REFINEMENT

The current set of display heuristics (see above section ***Display Heuristics***) works pretty well, in that users see something quickly and, if they want to try something different, tinkering with Create View and displaying the next result is as quick or quicker. Also, the Viewer and JView provide a wealth of functionality for playing an animation, pausing, running in reverse, zooming in and out, spinning the scene around, etc.

But sometimes a user can spin and zoom the display to his or her heart's content and not obtain the view he or she would like. In Figure 18 is an example of a display which, though it adequately shows airspace volumes with respect to each other, could be made better: The U.S. map is unnecessary and the orange ARTCC volumes make it harder to see the sectors, though one would like to see how the sectors fit within the overall scheme.

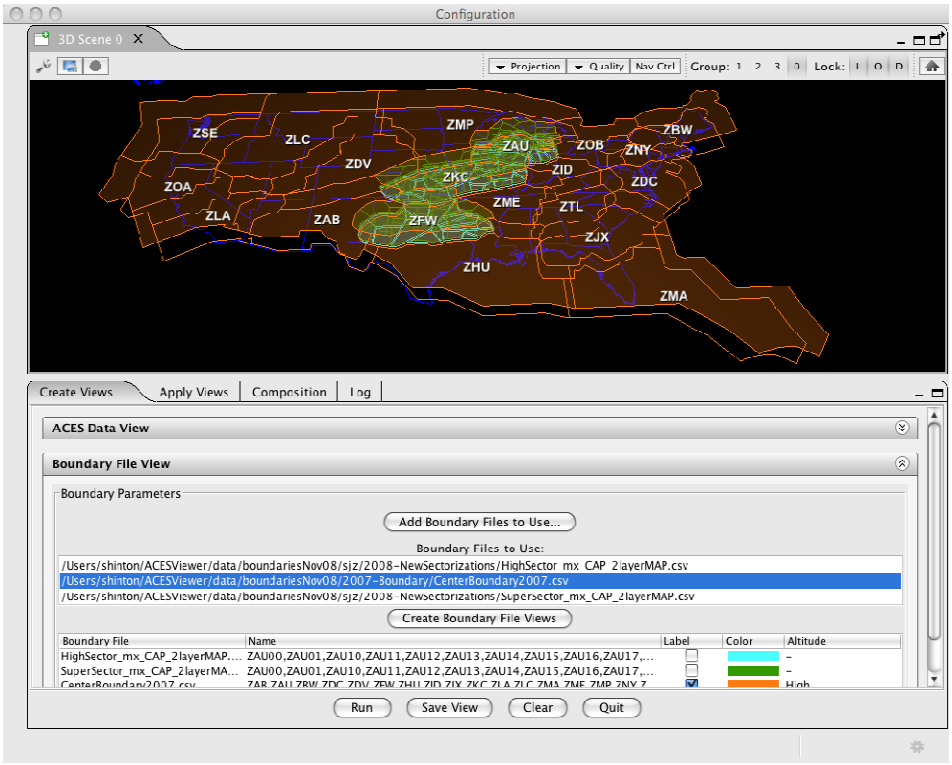


Figure 18: Experimental set of airspace sectorization with ARTCC boundaries

Fortunately, in this case, a user has options – in fact perhaps too many. The Viewer comes with a powerful UI of its own which, though less intuitive for most people, allows users to alter and tweak hundreds of settings.

The Viewer represents the internal workings of the graphics software with a “scene graph.” Though using the Viewer UI to build a scene graph from scratch can be challenging, even for software experts, student interns have been able to alter an existing scene graph with a little practice. IV4D causes the scene graph to be built and displayed in the Composition panel when it tells the Viewer to create a display.

The picture in Figure 19 shows the Composition panel scene graph that was created when the user clicked the Run button for the display above. On the lower left is a graph that depicts how data was filtered, and then rendered for the display shown above the graph. The Polygons renderer, circled at the bottom of the scene graph, is currently highlighted (i.e., a user clicked on the Polygons icon causing the icon color to change to a more saturated blue). The right side of the Composition panel contains a list of buttons that expand or contract when clicked. Currently the Property Editor section is expanded and shows settings for the highlighted Polygons renderer icon. Users can use these settings to modify the current display.

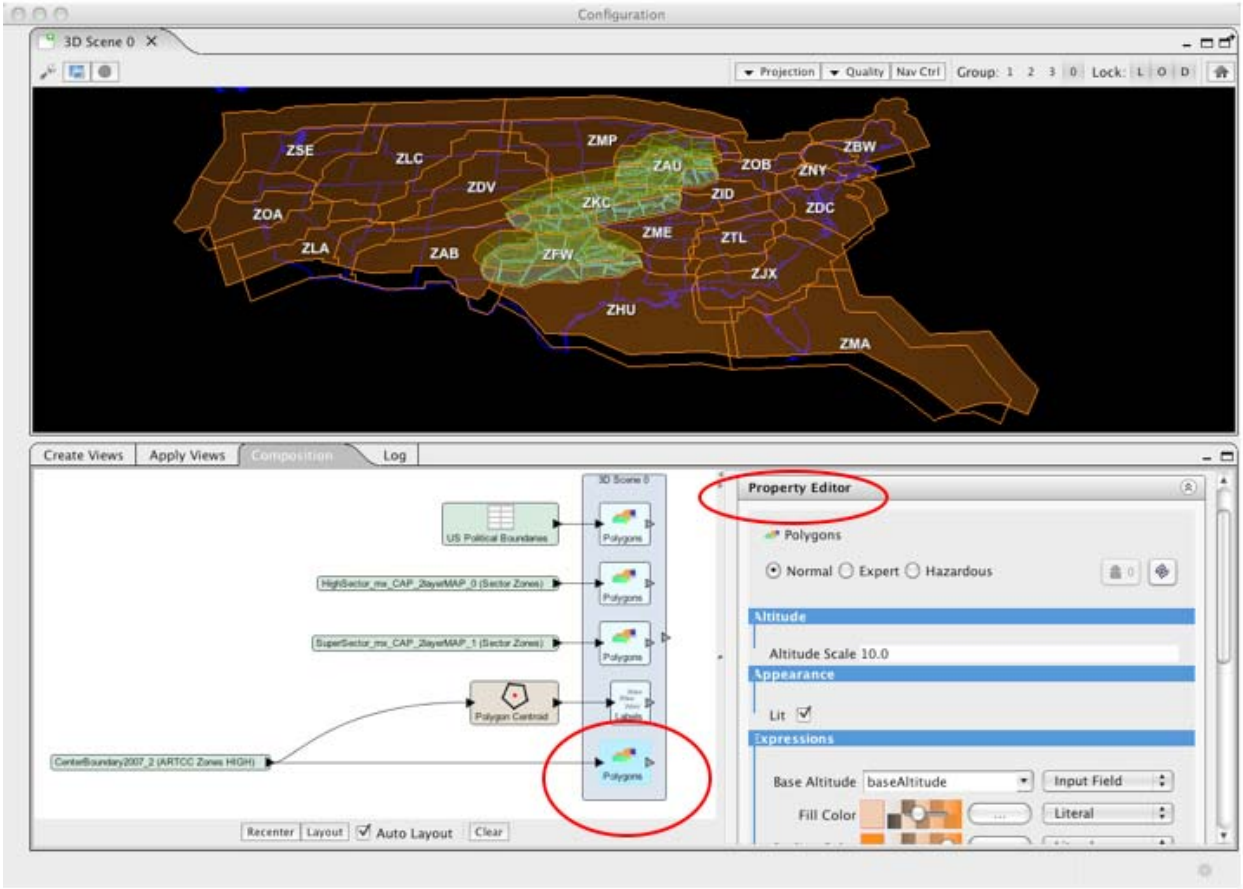


Figure 19: Viewer UI shows scene graph, on left, and property settings, on right

Figure 20 shows the improved display, made by editing as follows:

1) The Polygon renderer for the U.S. map was deleted, removing the U.S. map from the scene

2) The Center Boundary altitude settings were set to zero (0), so the ARTCCs appear as flat, and on the ground

3) The orange fill color for the ARTCCs was made transparent, so only the orange outlines remain

4) The Super Sector fill color was made less transparent, so the top plane is easier to spot

5) The High Sector shelves (vertical walls for airspace volumes) color was made less transparent, so sector height, in turquoise, is easier to see

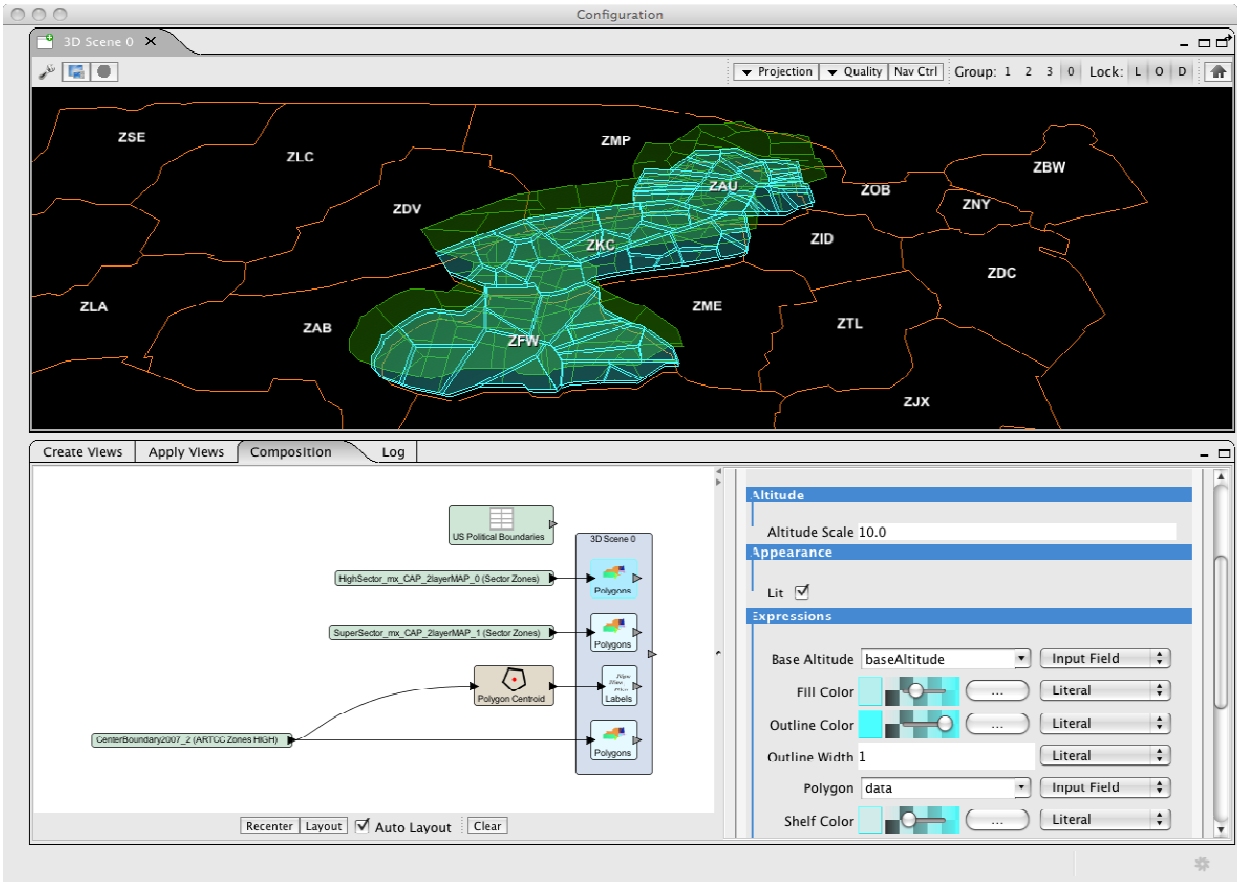


Figure 20: Experimental sectors are easier to see after changes to display properties

3.6 PERSISTENCE (SAVED VIEWS)

As mentioned earlier, user filtering and specifications set up in the Create Views panel can be saved as a kind of meta-data. The meta-data can later be loaded from the Applied Views panel to

1) Replay a display created in Create Views

2) Apply meta-data to new datasets, in order to quickly assemble a new display of previously un-visualized data

3) Replay multiple displays, created in Create Views, together

4) Apply multiple sets of meta-data to multiple new datasets, in order to quickly assemble a new display of previously un-visualized sets of data

A current limitation is that, because Create Views meta-data does not capture changes made by the user from the Composition panel (see the previous section ***Display Refinement***), the new displays may need to be updated from the Viewer Composition panel, once they appear. From the Composition panel, users can save a visualization that contains all display refinements. However, this form of saved visualization cannot be combined with other displays. (It can be applied to other data, but the process for doing so is prolonged, and is neither easy nor straightforward.)

To learn more about how Applied Views works, see the earlier section ***Applied Views***.

3.7 PERFORMANCE

Displaying a growing body of data continues to challenge display software applications, and IV4D is no exception. While a large portion of the planned functionality for IV4D was in place at the end of the first year of development, trying to visualize larger amounts of data resulted in a sluggish response and, eventually, out of memory errors.

Ultimately, in order to satisfactorily display a multitude of airspace polygons along with thousands of flights at a time, fixes and re-engineering needed to take place at all levels of the combined JView/Viewer/IV4D software application. JView developers were working on a long-term set of changes for tracking and drawing polygons. In the Viewer and in IV4D, memory errors were fixed and memory handling was improved. New open source packages were put into place, by the Viewer, and employed for handling comma delimited table data by both the Viewer and IV4D. An older persistence mechanism, Hibernate, was replaced with JDBC specific code and, over time, quite a bit of JDBC support code was added to the Viewer and to IV4D. The types and number of JDBC queries made by IV4D were expanded and reworked. In many cases broad queries made by IV4D were split into a series of more narrowly targeted queries with, in a number of instances, resulting data being put back together afterward. Finally, many pieces of code, generally, were reworked to provide faster processing with more efficient memory usage.

4 ORGANIZATIONAL METHODS, ASSUMPTIONS, AND PROCEDURES

IV4D was developed on the West Coast, at NASA Ames Research Center where key users live and work, while the Viewer and JView were developed on the East Coast, at AFRL in Rome, NY. Also, because IV4D modules Create Views and Apply Views plug into and utilize Viewer and JView functionalities, development of IV4D depended directly upon development being done at AFRL.

Normally developers at AFRL utilize an agile software development process, known to work best when developers work in close proximity, i.e., sitting nearby, while developers for this project were situated on opposite coasts. It therefore seemed like a good idea to introduce an alternate process, with a more methodical set of expectations than is typical for agile development.

This part of the report describes the process that was proposed and accepted, how the process changed over time, and lessons learned from this effort.

The goal is not to exhaustively describe pieces of the process, but rather to describe how adopting the process worked, or didn't, how it changed over time, and to suggest possible improvements.

4.1 ORIGINAL CMMI-DEV PROCESS

When developers belong to different organizations and are not sitting near each other, agile methodologies become difficult to follow. Since IV4D developers had experience working on ACES, a large, successful software project with as many as 20 people distributed across organizations and across the country, adapting and downsizing the ACES development process seemed like a good place to start.

The goal was to set up a process in line with CMMI (Carnegie Mellon's Capability Maturity Model Integration) process, targeted loosely between level 2, characterized as "Repeatable" – i.e., some processes are repeatable, with hopefully consistent results – and level 3, characterized as "Defined" – i.e., defined, documented standard processes have been established, and are subject to improvement over time.

The resulting process involved writing an Engineering Design Document (EDD), a Software Design Document (SDD), and a Software Test Document (STD), coupled with a defined set of test data. These documents were based on template documents produced for the ACES development project. The templates are abbreviated, annotated documents that can be filled in and completed by following the annotated directions. All documents are connected in that requirements entered into and described in the EDD form the basis for both software design in the SDD, and for testing in the STD. (A copy of the templates was passed to AFRL at the outset of the project. Interested parties may contact ACI for a copy of the template documents.) The IV4D development team was responsible for writing these documents.

Additionally, developers kept in touch through weekly teleconferences, email, and planned and impromptu code and document exchanges. A bug database was set up, and bugs and enhancement requests were entered and tracked. Also, during project development, researchers and aeronautical research developers had access to and made use of in-development, as well as released versions of IV4D. Their on-going feedback formed an essential step leading to a successful final product. In addition to reporting bugs, their usability and feature requests regularly changed the course of development. (Changes included, but were not limited to: Allowing users to pick the same data source as many times as desired while applying different filters and display specifications; Keeping the last visualization display up while the user creates and starts the next one; And, of course, insisting on improvements in performance.)

The Principal Investigator (PI) for IV4D acted as Project Manager for the project. The PI was responsible for setting up teleconferences and meetings, producing agendas and minutes for teleconferences, arranging document and code exchanges, arranging and encouraging user testing, formulating and incorporating responses to user requests and suggestions, entering and tracking bugs, editing and finalizing document production, and generally coordinating and shepherding progress on the project. The PI acted, therefore, as keeper of the process.

Regarding documents for project progress, the most useful turned out to be the Engineering Design Document and subsequent addendum, along with the Software Test Document, though the Software Design Document was useful for initial planning, actual design for the Viewer and IV4D changed quite a bit between planning and final execution. Given that the project did not require strict adherence to plan documents this was not, in itself, an issue. Also, keeping track of bugs and requests with a bug database was critical.

During the first year of three, IV4D developers fully embraced the software development process, though a Viewer API that evolved semi-independently lessened effectiveness of the overall process. The rapidly changing API meant that in addition to lagging Viewer development – IV4D modules plug into the Viewer software and therefore, depends directly upon the Viewer API – some stretches of IV4D code were repeatedly rewritten to match the API of the moment. (For this reason another developer, working on a separate effort to connect ACES simulation software during run-time to the Viewer, chose to work with an older, stable, version of the Viewer rather than try to keep up with the most current version.)

4.2 “PARALLEL PLAY”

During the second year the Viewer API stabilized somewhat and the process became looser, while the development emphasis shifted from initial design and implementation of planned features to improving performance.

Unlike the first year, it was possible for quite a bit of Viewer and IV4D implementation to progress independently, and API changes were more limited, more planned in advance, and there was more overall communication on how to properly use new APIs. During the second year, as a result, software development was consistent with what teachers of younger children call “parallel play” where youngsters play sociably, but independently, next to each other, with occasional interaction.

At the beginning of this period the EDD was updated through an addendum, and the testing document, rather than being expanded, was used for regression testing. Throughout the performance upgrade period existing tests were run with successively larger sets of data.

4.3 FINAL ORGANIZATIONAL STATE

Improved performance encouraged new users to use IV4D, which, in turn, led to useful feedback concerning feature enhancement and direction for the final, and third, IV4D software release. The final seven to eight months of development focused on the user experience, streamlining the user selection interface and adding more flexibility in filtering objects for display.

However the kind of independent development, between IV4D and the Viewer, that worked well for performance improvements did not work as well once development resumed in standard mode. Though the Viewer API did not change as much as during the first year of development, underlying Viewer design, including support for the Viewer and IV4D GUI, was being altered drastically and the Viewer, therefore, was not stable for a prolonged time. This made it difficult to properly upgrade and test IV4D since, as mentioned previously, IV4D depends directly upon the Viewer and does not run independently.

Also, returning fully to the original agreed upon CMMI process (i.e., stepping back to plan and upgrade EDD, SDD, and STD documents before taking up shared implementation) would have slowed both ongoing, independently planned (by AFRL) Viewer changes and final IV4D development. In an ideal setting, the development period would have been stretched to make room for shared planning but, in reality, IV4D development had a final deadline that could not be moved.

Through extensive end-of-project testing and bug fixing, Viewer functionality stabilized and version 3.0 was released. There remain, however, specific operations, when performed repetitively, that are known to cause a memory leak. It is hoped there may eventually be a 3.01 release of the Viewer to address this situation. Fortunately most users of IV4D and the Viewer are not likely to run into this issue, and release 3.0 is being used successfully at NASA Ames and NASA Langley Research Centers.

5 RESULTS AND DISCUSSION

In Table 1 is the original list of requirements from the Engineering Design Document. Of the 53 entries, 7 are “stretch goals,” which are goals that, once others are completed, may become a regular goal. Stretch goals are generally acknowledged as being more difficult. Nonetheless, two of the stretch goals were implemented. Additionally, as mentioned earlier, a number of user requests, not listed below, were implemented, and, though not added to the requirements list, some were added to the EDD Addendum. The addendum described a set of changes, implemented during the 2nd year, to improve performance for large data sets.

Of the regular 46 entries (non stretch goals), 37 have been fully implemented, 4 have been partially implemented, and 5 have not been implemented.

Notes on the partially implemented features can be found in the list below. Three of the five not implemented features belong to the category “Query Views.” These were intended to comprise a third plug-in module to the Viewer, along with Create Views and Apply Views. Query Views were meant to be a group of already assembled sets of Applied Views – In other words, a group of Applied Views that a user would not need to create, but that would be ready to be applied to most ACES data sets. Though it would have been desirable to complete this module, users are able to create sets of Applied Views, and it is therefore possible say these are partially – indirectly – present.

The other two features not implemented were requirements for the Viewer: One is directly related to a partially implemented feature having to do with jumps in time during visualization, while the other involves an ‘Undo/redo’ feature. Though it was originally thought that an undo feature would be highly important, editing a display has turned out to be so flexible that no user has thought to ask for an ‘Undo/redo’ button. Additionally, users can save their display in several ways, and at any time, again, mitigating the need for ‘Undo/redo.’

To summarize, IV4D is a user-centered piece of software created to help users study aeronautical research data. IV4D plugs into an AFRL API layer, the Viewer with JView, for visualizing the NAS. Responding to user needs, preferences, and requests, and incorporating Viewer and JView changes took precedence. The AFRL technical representative concurred with the work emphasis, and it always made sense to follow up on as many user requests as could be accommodated.

Key:

- White – Completed goal, or a stretch goal
- Yellow – Completed stretch goal
- Light Grey – Partially completed
- Purple – Not completed. Primarily “Query Views”

Table 1: IV4D Requirements

ID	IV4D View Requirements
1	It shall be possible to name, load, edit, use, and save 'views' of selected data and groups of views. Views are filters for pulling out specific viewable data.
2	There shall be named views and time views.
3	Named views shall reference position elements, such as latitude, longitude, and altitude.
4	It shall be possible to apply a named view to any ACES database or ACES boundary data file that contains data appropriate for the view. (Other data types may be added later.)
5	Time views shall be secondary to, and applied in combination with, named views.
6	A time view shall capture time slice and time segment data. A time slice is defined as having both a start time, which is => simulation start time, and an end time, which is <= simulation end time. (A moment in time is a time slice with the same start and end time.) A time segment is defined as a repeating unit of time, such as 5 minutes or 2 hours, and shall be <= an associated time slice. (A time segment for a time slice of a moment in time must also be a moment in time.)
7	Time shall be displayed as Days: Hours: Minutes: Seconds.Fraction from simulation starts time. For example 0:00:00:10.0 is 10 seconds after start time and 00:22:30:00.0 is 22 1/2 hours after start time.
8	The software shall track time in relation to simulation start time. The format will keep Hours, Minutes, Seconds, and Fraction of a second (e.g. Milliseconds) from simulation start time.
9	It shall be possible to apply a time view to any ACES database or supported ACES data file that contains timed data appropriate for the view.* *(*Timed CSV polygon data can be saved and applied in a view, though users cannot specify time slices less than the full start to stop time.)*
10	Any part of a time view may remain unspecified. When the user does this • A start time in a time view shall be applied as follows: One or more named

	views are applied to data and the earliest simulation time from the data becomes the start time for the time view. • A stop time in a time view shall be applied as follows: One or more named views are applied to data and the latest simulation time from the data becomes the stop time for the time view. • A time segment in a time view shall default to the length of the time slice.
11	It shall be possible to combine named and time views into a nameable group of views.
12	When time views are grouped a single new time view is created with rules similar to, but not the same as, requirement 10 above: • The time slice start time shall be the earliest start time from the views. • The time slice end time shall be the latest stop time from the views. • The time segment shall be the smallest time segment from the views.
13	When applying a named view it shall be possible to select specific table-column_names, such as KSYR or flightIds 2121, 320, 854 or specific boundary names, such as ZFW or ZSE3201M1 through ZSE3201M5.
14	Stretch Goal: IV4D shall read ACES text input files other than boundary files that contain either latitude, longitude, and altitude or altitude plus a series of latitude and longitude points (such as in a Flight Data Set file).

ID	IV4D Query View Requirements
15	Users shall be able to name, load, edit, use, and save a specific type of combined name and time view, called a 'Query View'. The purpose of a Query View (QV) is to display answers to questions frequently posed by users.
16	This QV shall be supported: Given a boundary object A, show those selected objects (table:name:lat/lon/alt) which were in or at A during time slice X.
17	This QV shall be supported: Given boundary object A, time slice X, and selected objects L which were in A during X, show where and when selected objects L originated.
18	This QV shall be supported: Given boundary object A, time slice X, and selected objects L which were in A during X, show where selected objects L were at time T, before or after X.
19	Stretch Goal: There shall be one or more QV for displaying flight delay Visualization. NOTE: FlightTimeDataMessage.

ID	IV4D View Auto fill Requirements
20	Users shall select an ACES database or ACES boundary file to start the view creation process.
21	When an ACES MySQL database is selected for view creation, the application shall automatically look for tables and sets of similarly named tables (i.e., the first part of each table in a set of tables is identically named) that contain the tokens “latitude”, “longitude”, and “altitude”. The appropriate table names will be stored for later presentation to the user.
22	The user shall be able to select from a list of table names that have been found to contain “latitude”, “longitude”, and “altitude”.
23	For each table name selected by the user, the user shall pick a column name to associate with “latitude”, “longitude”, and “altitude”.
24	The application shall create time views that are attached to and saved with each named views or group of named views.
25	For time views, the user shall have the following options: leaving the time fields in an unspecified state, having the application automatically find time slice start, stop, and time segment values, or the user can enter values into the fields.
26	When the application automatically looks for time slice and time segment values, the application shall find the earliest time in the tables or set of tables for the time slice start value, the latest time in the tables or set of tables for the time slice stop value, and shall set the time segment length to the length of the time slice.
27	The application shall prevent the user from entering time slice start values before 0:00:00:00.0 or time slice stop values after the simulation stop time or time segments larger than the difference between the time slice stop and start times.* (**One can enter a stop time that exceeds the data stop time. Display playback, however, will not continue past the actual data stop time.*)

ID	ACES Viewer Scene Configuration Requirements
28	View instances passed to the scene configuration shall consist of tables.
29	One or more scene configuration Data Source classes shall be available for tables produced by IV4D
30	One or more Renderer classes shall be available for tables produced by IV4D

31	One or more Control classes shall be available for tables produced by IV4D
32	The Property Editor shall be available for renderers which use tables produced by IV4D.
33	It shall be possible to see appropriate scene configuration visualization for view instances passed to appropriate IV4D scene configuration renderers, data sources, controls, transforms, and property editor.
34	Tables produced by IV4D for visualization shall include selected name, position or positions (in the case of a trajectory, line, or polygon), and time slice. Other elements may also be included.
35	It shall be possible to name, load, edit, use, and save scene configurations.

ID	ACES Viewer Visualization Requirements
36	When IV4D tables are properly passed to the appropriate scene configuration renderers, data sources, controls, transforms, and property editor, the user shall see an appropriate Visualization.
37	Visualization shall include a 'frozen' scene, where table elements in a time slice are seen.
38	Visualization shall include a moving scene, where objects appear, disappear, and move as appropriate during the time slice (i.e., like a movie).
39	Visualization shall include sequential time segments, where each segment is displayed in a sequence of contiguous, equal size Frozen instances, such as twelve 5 minute segments across an hour, i.e., like a slide show.* (**The time control Step-forward and Step-backward buttons allow users to step through a series of regular time jumps, but the time control Play or Reverse-play buttons will not automatically play back a series of regular time jumps that are not in the data.*)
40	Visualization shall include animated time segments, where each segment is animated in a fashion similar to a time slice (i.e., like a series of short movies).
41	It shall be possible to associate a time range with every scene element, including boundary points, lines, and polygons. (This will lead to animated boundary elements at a later time.)
42	Stretch Goal: It shall be possible to display animated (moving) boundary data.
43	Stretch Goal: It shall be possible to display popup visualizations, such as a graph

	or object list.
44	Stretch Goal: It shall be possible to display sequential time segments in a slide tray style Visualization

ID	ACES Viewer Visualization User Interaction Requirements
45	Users shall be able to save comma delimited plain text output from the tables associated with the Visualization. (Other text formats may be supported later.)
46	Visualization of the scene configuration shall be kept in synchronization with the scene configuration
47	Users shall be able to show/hide scene elements on-screen, and saved comma delimited output from tables associated with the visualization will reflect what the user sees.
48	Users shall be able to edit specified scene elements on-screen: Supported editing shall include adding, removing, and changing boundary points, lines, and polygons. (Other elements may become editable later.)
49	Users shall be able to undo, redo, or reset their edits. Reset will revert the visualization to the original state.
50	Users shall be able to select and pivot around an anchor point on-screen.
51	Users shall be able to sequentially play time segments either manually (forward one at a time), or automatically (the 'slide' changes every few seconds). The segments themselves may be in a 'frozen' or 'animated' mode. NOTE: combo of Transform and Control.* (**See the comment for Requirement 39.*)
52	Stretch Goal (follow on to initial effort): keep track of user boundary changes in order to produce timed/boundary data files
53	Stretch Goal: When user saves content from tables associated with Visualization, the UI shall allow user to rearrange column order and show/hide columns for output. (Not for internal representation, just a filter for table output.)

6 CONCLUSIONS

IV4D and the Viewer form a successful set of software tools that are being used by aeronautics researchers at both the NASA Ames and the NASA Langley Research Centers. Additionally, IV4D was considered interesting enough to cause NASA Ames Research Center to fund a separate plug-in module for the Viewer, the ACES simulation run-time plug-in.

6.1 TECHNICAL RESULTS AND USER SATISFACTION

The goal was to produce a tool that is easy to understand and use, performs well, and provides researchers a way to study and understand national airspace (NAS) using various forms of input and output data, such as data from NASA's ACES (Airspace Concept Evaluation System) simulation software.

To date, IV4D has helped visualize portions of the NAS using data from ACES and CTAS (software applications from NASA Ames), SAHITL ASTOR (human-in-the-loop simulation software from NASA Langley), ASDI (Aircraft Situation Display to Industry) data from the FAA, and weather data modified from various NOAA formats. Users regard the application as a flexible, easy to understand, and easy to use tool, that performs well. It provides researchers with a way to observe and understand both still and animated NAS data.

IV4D functionality has plenty of room for growth. For example, a considerable improvement would involve standardizing IV4D and Viewer data so users are able to move more seamlessly between visualizations saved from the Viewer Configuration panel and visualizations saved from IV4D. Other possible improvements include allowing users to perform simple edits – such as changing colors – in the Apply Views panel, making Create Views and Apply Views modules more malleable with respect to each other, improving heuristics for default display results, and adding new default methods to visually compare and understand one set of data with respect to another.

6.2 ORGANIZATIONAL METHODS

Development of IV4D, which plugs into the AFRL Viewer, is dependent upon the Viewer API. Unfortunately for IV4D, the Viewer API was stable only intermittently. It was hoped, at the outset of the project, that deciding upon and following a CMMI process, would enforce shared planning and synchronize development during the project. But this was not the case because, while IV4D developers followed the process closely, at least during the first year, Viewer developers continued their existing practice (i.e., agile development, known to work best when developers are in close proximity).

The main benefit of the adopted process, which was a downsized and streamlined version of NASA's ACES development process, was that IV4D and Viewer developers did have a written, shared understanding of IV4D goals. This helped foster development of Viewer functionality needed by IV4D. The process also helped IV4D developers to rationally design (i.e., think about) the initial effort.

Had the Viewer kept a stable API there is no doubt that IV4D would have progressed more rapidly. Instead, especially during the first year, IV4D developers found themselves needing to repeatedly rewrite some stretches of code to meet a changing API. In spite of this, users were enthusiastic about the first release and about follow-on releases.

The best model for developing modules for the Viewer made an appearance when, as a result of user enthusiasm for IV4D, NASA funded a new, separate effort to connect the ACES application to the Viewer during ACES run-time. (IV4D works with non run-time data.) For the new effort, NASA stated it wanted a product that, once working initially, would continue to run while new features were being added. This necessitated development with a stable version of the Viewer and, in fact, development was done with a version of the Viewer that did not contain later performance-enhancements. This meant that the developer for the new plug-in was forced to add performance-enhancing features in order to work around Viewer/JView performance issues. This also meant that the best developer for this work was a senior graphics developer rather than an aeronautics developer, as a senior graphics developer was able to read through Viewer/JView code while devising workarounds for known issues.

While the IV4D development team continues to believe that a shared CMMI process could work for a shared East-West Coast development cycle, especially since other projects have been successful in this fashion, such a model does not appear to be best for the current project. In the event of a follow-on effort, we recommend that CMMI development be abandoned – at least as far as trying to use CMMI with developers hewing to a non-CMMI model – in favor of having IV4D developers work with an older, known to be stable, version of the Viewer. Upgrades to newer versions of the Viewer would occur once the Viewer was thoroughly tested and declared stable.

LIST OF ACRONYMS

ACES – NASA software application, Airspace Concept Evaluation System, for simulating activity, such as flights and control of flights, in the NAS

ACI – Aerospace Computing, Inc.

AFRL – Air Force Research Laboratory

AFRL Viewer – AFRL JView application for visualizing NAS data (originally) from ACES

API – Application programming Interface

ASDI – Aircraft Situational Display for Industry, FAA flight data

ARTCC – Air Route Traffic Control Center, sometimes referred to simply as a Center

ATM – Air Traffic Management

CMMI – Carnegie Mellon's Capability Maturity Model Integration, a process improvement approach for software and systems engineering projects

CPU – Central Processing Unit, an essential part of a digital computer

CSV – Comma Separated Values, a data format delimited by commas

FAA – Federal Aviation Administration

GUI – Graphical User Interface

IV4D – GUI plug-in to the AFRL Viewer, created for NAS researchers

JView – Extensive visualization library created by AFRL

MySQL – A semi-open source (see licensing for MySQL) implementation of SQL

NAS – National Airspace

NASA – National Aeronautics and Space Administration

NOAA – National Oceanic and Atmospheric Administration

PI – Principal Investigator

SAHITL ASTOR – NASA simulation software, Separation Assurance Human-in-the-Loop Aircraft Simulation for Traffic Operations Research

SQL – Structured Query Language, used to access data from relational databases

UI – User Interface

Waypoint – An intermediate point, marked by latitude and longitude, along a route