The Cube Analysis and Rendering Tool for Astronomy

Final Project Report

2017-01-06

Prepared By	Organizational Role	Signature
Erik Rosolowsky	Project PI	

Revision Number	Date	Affected Sections	Authors	Change Description
v0.0	2017-01-06	All	Erik Rosolowsky	Initial Release
v1.0				

1. Executive Summary

This document presents the final report on the ALMA Development Project (Cycle 2) entitled *The Next-Generation ALMA Image Viewer*. This project funded the development of the Cube Analysis and Rendering Tool for Astronomy (CARTA), which has been developed and delivered to NRAO for ongoing development in the CASA group. The tool will be a drop-in replacement for the current viewer software presenting the Common Astronomy Software Applications (CASA) package.

CARTA provides image visualization and analysis, mimicking much the functionality present in the current CASA viewer. At present, CARTA has a robust architecture that serves as a foundation for ongoing development. At the time of delivery, some pieces of functionality had not been replicated in CARTA, but there is a clearly defined path for establishing that functionality. However, CARTA surpasses the current tool in the flexibility of its user interface and multitude of use cases.

In addition to the standard visualization application mode that astronomers typically use, CARTA also allows for server-client visualization using a remote server hosted in a data archive and a user's web browser. CARTA also allows for scripted deployment inside a pipeline environment. The remote server use-case is enabled by the PureWeb software, a proprietary package developed by Calgary Scientific, Inc., and licensed to NRAO without cost.

CARTA is authored in C++ and JavaScript using a modular architecture that will enable easier maintenance, rapid expansion of functionality, and flexible deployment on a wide range of platforms. The application has a well developed system of plugins, which provide the basic functionality for the visualization. CARTA is developed on the github platform as an open source project.

CARTA has been reviewed by an expert committee of users, the CASA Users Committee, and a panel of expert users at NRAO Charlottesville. These reviews have raised ongoing concerns, but they have broadly endorsed forward progress for the project. The feedback from these reviews as well as individual users is integrated through the github site and addressed by developers.

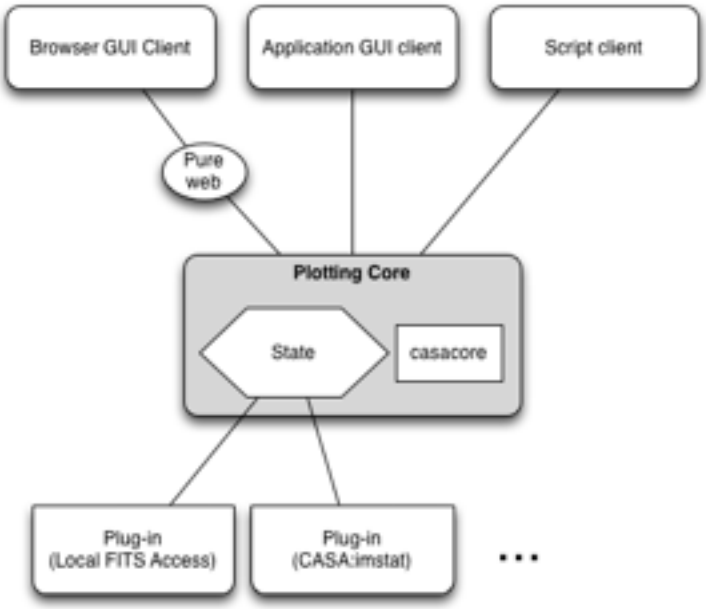
At the time of writing, CARTA has been transitioned to the CASA group at NRAO, where development is going. CARTA will be released into the CASA package in the near future as an alternative to the current viewer tool.

2. The CARTA Software Project

The Cube Analysis and Rendering Tool for Astronomy (CARTA) was developed as part of the Cycle 2 North American ALMA Development Projects. This document represents the final report for the project entitled *The Next Generation ALMA Image Viewer*.

2.1 CARTA Software Overview

CARTA is a visualization tool for astronomical data. In particular, it is intended for use with submillimetre-wave telescope data products produced by the Atacama Large Millimetre/submillimetre Array (ALMA). It provides scientific visualization for the most common astronomical image data formats (the Flexible Image Transport System, FITS, and the Common Astronomical Software Applications, CASA, image format). The tool is intended as a drop-in replacement for the viewer application that is part of the CASA software system, providing the ability to meet the future visualization needs of the astronomical community. CARTA, in particular, provides an enhancement to the ALMA Archive by allowing for a novel astronomical use case, where astronomical visualization is performed by a remote server and the user connects to that server through their web-browser. By enabling visualization in the ALMA Archive without a user having to download data, the CARTA tool reduces the bandwidth requirements and enhances the use of the archive. CARTA also includes a use case where it can be run as a stand-alone piece of software on the user's machine as well as through a series of scripted commands written in the Python language.



CARTA also uses a modern software development structure and system. This allows for agile development, ease of maintenance, and contributions from the user community. These features are enabled by the CARTA architecture, which is summarized in the Figure at left. The architecture is tiered into three separated layers. At the heart of CARTA, is the Plotting Core, which tracks the *State* of the application. The State is an XML data structure that describes the entire state of CARTA application. The Plotting Core also includes the casacore software library, which provides essential context for software to interpret astronomical data types The Plotting Core allows for user interaction through a server-client relationship with the Plotting Core and this software is abstracted into a *client*. These clients take three forms: an Application GUI client, which provides functionality on a user's machine, the Browser GUI Client

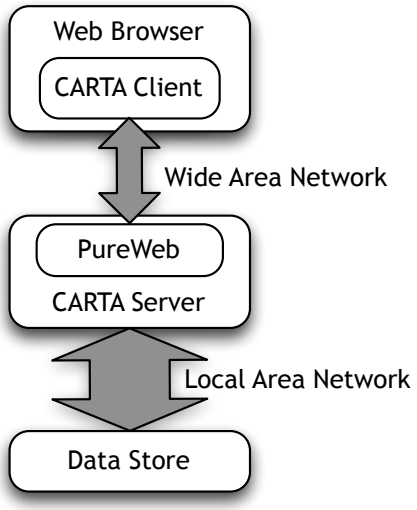
which provides the same functionality as the Application Client but allows for the client to be the user's web browser and the server to be hosted at a remote location. Finally, a scripted client allows the full application control to be managed by a Python script, allowing the CARTA software to be used in a pipeline environment.

Most of CARTA functionality is provided through a well developed plugin ecosystem, with a design goal of exporting as much functionality as possible into plugins and not in the plotting core. Plugins have a hierarchy for functionality, allowing multiple plugins that provide the same service to coexist. For example, a statistics plugin could use standard methods on a user's computer while another may connect to a statistical package running on a high performance computing resource. The plugin system can be configured to use on or the other plugin to provide statistical analysis.

CARTA Software displays astronomical data as an image viewer with full awareness of image metadata (units, coordinate systems, etc.). It allows for flexible visualizations that highlight different features of the data including small locations within an image and changing the brightness and color-scale to highlight certain ranges within an image. It allows for different images to be displayed in conjunction with each other. It also allows for the annotation of the images and selection of regions of interest within the image.

2.2 PureWeb Software

The new remote server application is enabled by the PureWeb software, a proprietary product created by Calgary Scientific, Inc. PureWeb allows for the efficient synchronization of an XML object over the Wide Area network between a server and a remote client though that client's web browser. Since the Plotting core stores the State as an XML object, this allows for changes in the XML state to be coordinated across the internet. A schematic of the behavior of PureWeb is shown in the Figure at the Right.



The primary application for PureWeb is for medical imaging and Calgary Scientific provides the PureWeb software to the CARTA project and NRAO without cost. The expectation is that the CARTA use case will highlight areas of improvement for the medical imaging applications.

2.3 CARTA Language Choices

CARTA is written in C++ and JavaScript. The C++ (v. 11) language manages the plotting core and many of the plugins where high performance is required for the visualization applications. The Browser and Application clients are written in JavaScript, which can be executed by web browsers. CARTA uses Qt 5.x as an application development framework with the qooxdoo framework to link the JavaScript to C++ of the plotting core.

The choices for these languages was shaped by the need for prospective development and platform agnostic. This requires focused development on those parts of the

functionality that are part of W3C web standards. Some functionality is restricted by there not being a well developed standard for, e.g., dashed lines in the JavaScript drawing.

2.4 CARTA Supported Platforms

CARTA is intended the be supported on the same platforms as the CASA applications. Specifically, this means that CARTA is supported on Linux and Mac platforms. The base development case for Linux is Ubuntu 14.04 LTS and Red Hat Enterprise Linux (RHEL 6.x). The Ubuntu platform is not formally supported by CASA, but recent version operate without difficulty on Ubuntu. Ubuntu is a "developer-friendly" platform which has more up-to-date standard libraries than the RHEL standard software. On Mac, CARTA is currently supported on Mac OS 10.10 and 10.11. The software should operate on the recently released 10.12 version.

Since CARTA can be run as an in-browser application, there is some dependence on the choice of browser. For this case, the Firefox browser is selected as the primary test environment, but good functionality is seen on all the standard browsers including mobile browsers such as Mobile Safari. Thus, CARTA can be used on iOS and Android devices as well as standard laptop and desktop computers.

CARTA servers can be deployed on generic hardware owing to a reliance on the Docker containerization scheme. The Docker software environment provides for basic library dependencies on a wide range of system architectures. This also facilitates migration of CARTA servers to new hardware as it becomes necessary based on hardware lifetime or changing user needs.

3. Project Summary

This project was completed as a partnership between the University of Alberta, the CASA Software Group at NRAO, and the University of Calgary. The project was managed by the University of Alberta, which then provided subcontracts back to NRAO and to U. Calgary.

3.1 Project Personel

Project Management Personnel included Erik Rosolowsky (PI, U. Alberta), Jeff Kern (NRAO) with additional science advising from Gregory Sivakoff (U. Alberta) and Russ Taylor[1] (U. Calgary).

Project Design was completed by lead developer Pavol Federl (U. Calgary), Jim Jacobs (NRAO), and Susan Loveland (NRAO) in consultation with the Project Management.

Scientific Development was completed by lead developer Pavol Federl (U. Calgary), Susan Loveland (NRAO), Jeff Taylor (U. Alberta), and Alex Strilets (U. Alberta).

Project reviews were conducted by three groups. First, an external advisory board consisting of Alyssa Goodman (Harvard), Crystal Brogan (NRAO, CASA Project Scientist), and Adam Leroy (Ohio State University) reviewed early versions of the software advised on project direction and architecture. Second, the project was presented to the CASA Users Committee twice in the project tenure. Finally, a review by scientific users was conducted at NRAO Charlottesville in April 2016.

3.2 Organizational Structure

The distributed nature of the software development required assigning responsibility for different components of the project to different sites to minimize the need for frequent team communication. Teams were coordinated with a weekly telecon. The project was divided into individual stages. In each stage, the lead developer (Federl) created prototype code consistent with the Architecture principles. The code was then extended and moved into a production mode by the rest of the development team (Loveland, Taylor, Strilets). Once functionality was in place, the software was tested by the scientific team (Rosolowsky, Kern, Sivakoff, Taylor), providing feedback to the development team.

3.3 The CARTA Software Project Operational Goals

CARTA software is designed to be a drop-in replacement to the CASA viewer with several improvements. First, the software is designed have a minimum of the functionality present in the current CASA viewer. It should be more stable than the CASA viewer, being easier to maintain. CARTA should show better performance, in particular on large data sets. Finally, the project must integrate into the current CASA software set, becoming part of the development stream as well as part of the user software experience.

---

[1] In 2014, Russ Taylor relocated from U. Calgary to the University of Cape Town in South Africa. Since U. Calgary was signatory to the contract, activity and funding remained at U. Calgary. The U. Cape Town group has contributed some wo

3.4 CARTA Development Phases

CARTA was developed in several stages, successively building on the functionality of previous build stages. The following summarizes the development phases that took place in the main portion of the funded activities.

1. Architecture — This stage developed the primary architecture for the system, consisting of a "thin" core software which tracks the State of the visualization tool, a series of plugins which act as components of the viewer, and three visualization clients (a desktop client, a browser client, and a scripted client). The Architecture also defined the state in the eXtended Markup Language (XML),
2. Core — This component tracks the State of the visualization tool. The visualization clients render components of the State and plugins alter the State given user input. The core also provides the basic framework for interpreting astronomical data via the `casacore` library.
3. Plugin Management Framework — This system manages different plugins to the CARTA system and their interaction with each other. By default, plugins modify the State and these changes are managed by the core. However, functionality is also provided for plugins to modify other plugins. Nearly all functionality is managed by the plugin system. The system also includes a polling system where a user request is tested by the full chain of plugins to determine which plugin and with what priority user interaction is responded to. Documentation for authoring plugins is included in the delivery.
4. File Loading — This functionality reads images in the Flexible Image Transport System (FITS) and CASA image formats. The file loading system is sufficiently generic that it can also show images in the JPEG and GIF formats as a proof of concept for other file formats.
5. Basic Graphical User Interface (GUI) — The basic GUI is set up as a gridded panel display with a menu bar. The GUI hosts the remaining tools and renders the state of those tools. The GUI is fully customizable and can be saved into a file format that preserves either the configuration of the tool or the entire visualization session.
6. Server-Client Visualization — This functionality provides the ability for a user to use a web browser to control a visualization client running on a remote server over the internet using their web browser. This allows for the user to navigate large data files on dedicated visualization servers within an archive.
7. Image Viewer — This provides a two dimensional image viewer with all functional required for scientific interpretation of the data in a 2D or 3D ALMA data set. This includes the ability to navigate around the image, change the color scale to highlight different data ranges, display coordinate axes around the data, overlay a coordinate grid on top of the image, and change all the properties of the image.

8. Scripted Client Interface — This interface provides the ability to manage the visualization client using scripts written by the user and executed. These scripts provide handles to update the state of the viewer and drive plugin operation without any user interaction. The primary use case for scripts is in pipeline visualization where the quality assurance plots can be made using the CARTA scripting interface.
9. Color Mapping — This functionality converts raw data values in the image into color rendering of those image data. This image rendering is provided in a high efficiency way to have good performance on large images. This functionality necessarily also provides efficient mapping between data value and rank of the data (e.g., the top 1% of image brightness).
10. Histogram — This tool displays the probability density function of data values in an image and relates this to the color mapping. The tool displays the histogram and allows the manipulation of the displayed image ranges through the histogram interface. The histogram also supports many modes for relating image values back to the probability density function including shading the histogram.
11. Regions of Interest — This functionality allows for a user to define Regions of Interest (RoIs) within the image and perform operations on these regions (extract spectra, measure statistics, extract subimages). The RoIs also serve as a channel for annotations.
12. Profile Tool — This tool shows 1D profiles through image data sets of higher dimension as well as statistical summaries of the individual planes of 2D data from a 3D data set. The tool interprets all the spectral conventions supported by the astronomical standards and can convert between them.

3.5 Development System

CARTA was developed using the git version control system and the team is managed through [2] which provides software project management functionality. The system worked well with the distributed team. The system uses a branched commit structure, where each developer has their own main branch. The developers fork development branches off their individual development branch, edit codes and then pull the development branch back into their personal branch. Once a stage of code development is complete, the developers file a pull request into the Develop branch. The Develop branch is the main testing branch. All pull requests into Develop undergo a code review before being merged. Once the Develop branch has accumulated a stable, significant set of changes, and all tests are passing, the Develop branch changes are pulled into the Stable branch. The flow of a set of changes into the Stable branch is shown in the Figure below.

Developer branches and the Develop branch are part of the Continuous Integration (CI) testing system. For each commit or pull request in the Developer and Develop branch, the program is automatically compiled and subject to a series of predefined

[2] http://github.com/AstroUA/cartavis

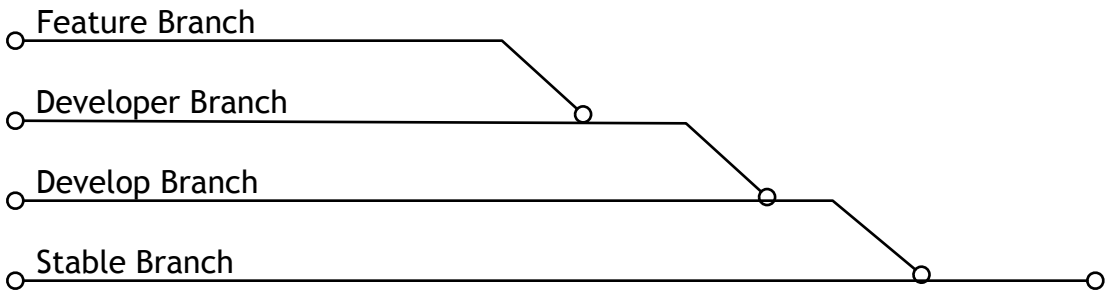
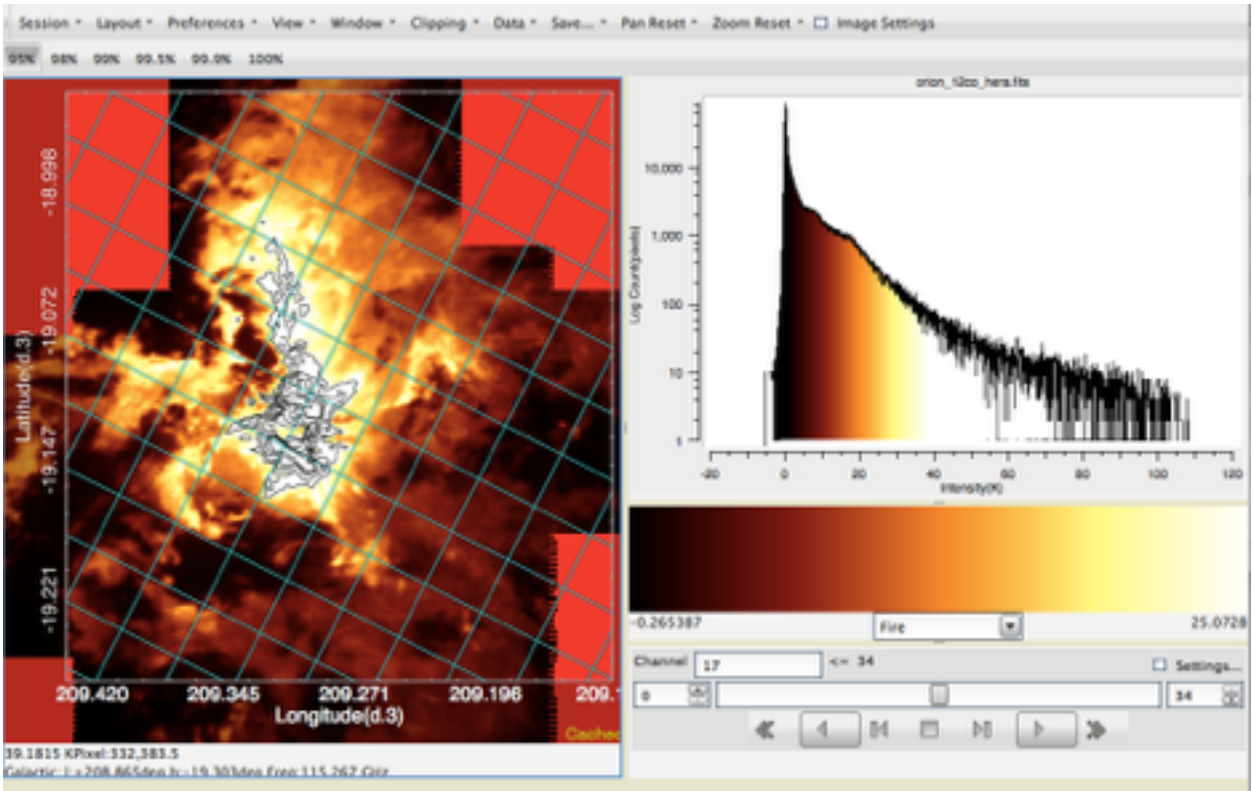


Diagram of the development streams and how they merge into the Stable branch for CARTA.

tests. Each test executes an action with the CARTA software and compares the response to a pre-defined behavior. If the behavior is inconsistent with the test or if the program crashes, the test fails and this is flagged. Currently, we are using the Circle CI system, which integrates into the github.com management system.

3.6 Testing and Requirements Capture

Once the CARTA project had developed sufficiently that a visualization tool was usable, it underwent performance review by several different stakeholders. Software testing was undertaken by the internal scientific team on beta versions. Early project steering and software review was completed twice by an Expert Review Panel. The team provided input to the project which shaped development and established the priorities for the project. The project was summarized to the CASA Users Committee at two separate meetings. Finally, a user review took place at NRAO Charlottesville in April 2016. The major concern found in these reviews was the slow pace of development, in particular since CASA viewer development has been slowed to allow for CARTA development. Several other pieces of user feedback were directly

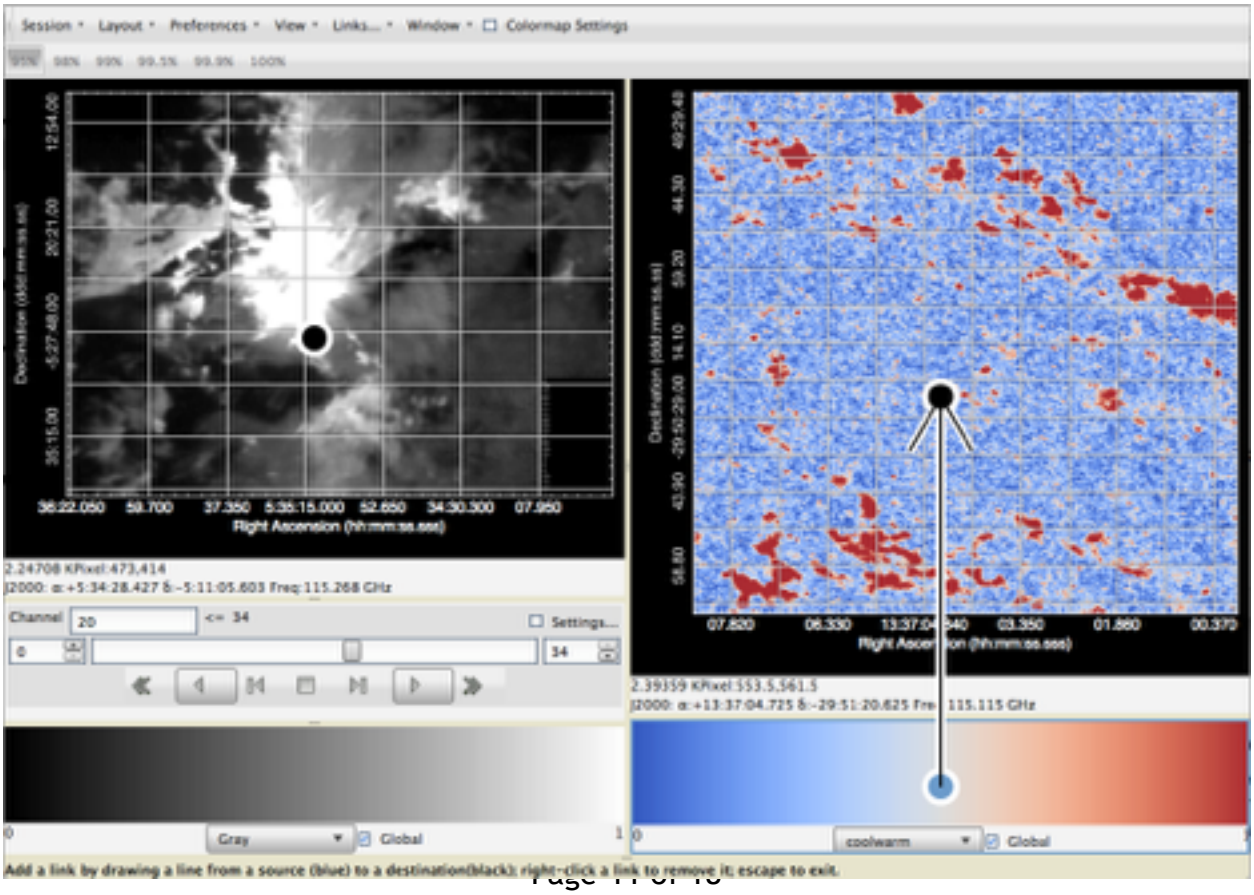


incorporated into the project and the feedback was used to prioritize ongoing development.

3.7 Software Example

The figure above shows an example of CARTA, displaying a 3D millimetre-wave spectral-line data set. The left hand panel shows one 2D plane from the image. The figure has a coordinate grid and brightness overlaid on top of the image. The bottom information panel is live with cursor movement, displaying information regarding the current pixel. The histogram on the right displays the image brightness distribution and is color-coded corresponding to the color map in the image. The color bar below the histogram shows the range of values in the image and allows for control of the color mapping and intensity ranges. Finally the "tape deck" animator controls at the bottom of the image selects different planes from the 3D image set.

The user interface uses the notion of *Views* that represent specific components of the UI. Each View is implemented and updated by one or more plugins. For example, the image viewer is a single View, but so too is the histogram and animator. The GUI canvas can be reorganized to meet user preferences. An example of this is shown in the Figure below where the viewer is reconfigured to show two separate images. To allow for users to specify which UI components control which parts of the CARTA viewer, a system of *Links* has been developed where a user can link a tool like an color controller to a specific image. Below the color viewer is linked (arrows) to the image viewer on the right.



4. Performance Review

This section reviews the performance of the CARTA project with respect to original project specifications, project timeline, and project budget. Because this is a software project and is governed by labor costs, these issues are inextricably linked. Overall, the CARTA project has had a linked shortfall with respect to its overall performance goals. To ameliorate this shortfall, the project has compromised on specifications and timeline with no change to budget. This decision was undertaken because the long term plan for CARTA is to deliver it to the cASA

4.1 Performance to Specifications

The CARTA application was developed as a sponsored research contract through the North American ALMA Development Stream. At the time of proposal, a set of functionality specifications was outlined in the application document. This list of functionality requirements is reproduced in Appendix A, which includes the current status for each specification and priorities for addressing these objectives.

The application also included a specification for a hardware server component to be purchased and deployed in ALMA archive locations globally, allowing the server-client use case. While the ALMA archive group remains interested in the possibility, the integration component of this server is not currently in their budget. In addition, our development team proposed that the server should not be a hardware entity but instead to use the developing virtual server technologies. This allows for far easier maintenance of the CARTA virtual server and migration/expansion to different hardware resources as they become available. Given these evolving changes, we requested and were approved for a Change Order to our deliverables.

As the project progressed, it became clear that the architecture and design phase of the application was not appropriately planned for at the application stage. The design and implementation of the architecture described above required more labor than budgeted. This has led to accrual of technical debt with respect to the original specifications, but also more advanced progress in other areas.

4.2 Performance to Timeline

The timeline was adversely affected by the architecture phase lasting longer than originally anticipated. Several phases of the development also took longer than expected (e.g., RoIs and color mapping), but several were facilitated by the novel architecture that allowed for rapid re-use of the CASA viewer code (e.g., image statistics and profiles). These delays have been addressed through our transition plan, which allows for ongoing work on this project.

4.3 Performance to Budget

CARTA was completed without budget variances, though a different set of deliverables was achieved than originally proposed. Many of these deliverables were addressed in the Change Order, which removed the obligation to provide hardware servers and instead developed virtualization software. Since the CARTA project is ongoing, the software development team remains engaged with bringing a set of functionality.

The primary difficulties in Budget Management came from currency fluctuations. In particular, the Canadian dollar fell 20% with respect to the US Dollar. The project had US dollar denominated obligations in the form of contracting back to NRAO for a portion of the CASA development team (i.e., Loveland, Kern, and Jacobs). Since the funding was paid to U. Alberta and converted to Canadian Dollars on a different schedule than payments were returned to NRAO, the value of the funding dropped significantly. This led to a shortfall in funding that was addressed by reduced scope of the project, namely eliminating hardware servers and not finishing low-priority deliverables.

5. Project Closure and Transition to NRAO

The CARTA project has transitioned to the CASA software group at NRAO for ongoing development. The transition has been facilitated by the ongoing work with the CASA group throughout the entire project. The delivered products include:

- Access and ownership of the open source CARTA software on github.
- Documentation including
    - Project reporting (this report and financials reporting)
    - Software documentation in the Plone system for CASA
    - Development framework documentation (Delivered to the ALMA CASA Development Center in Taiwan)
    - Development guidelines and tutorials
- Software licensing for the PureWeb software.

The software version at the time of transfer is v0.7, where v1.0 represents a fully-featured replacement for the CASA viewer. The CASA team is continuing work on developing CARTA, which has become an CASA software project. The PI for CARTA development, Erik Rosolowsky, has moved into the role of Project Scientist and will continue to help guide development.

CARTA will be packaged with the remainder of the CASA project in upcoming releases, allowing the entire CASA user base to experiment with the software before the CASA viewer is retired.

6. Lessons Learned

The CARTA project highlighted several lessons regarding outside software development with respect to the ALMA project.

6.1 Architecture Time was Underestimated

While the CARTA architecture that emerged has proven to be well matched to the use case and easy to integrate into the CASA system, the time taken to create and then implement the architecture was underestimated by six months. This underestimate stemmed from a lack of foresight by the PI, a lack of clarity regarding the internal operations of CASA, and the time it took to implement a full solution to the broad set of requirements.

6.2 Mitigating the Effects of Currency Fluctuations

The original contract was exposed to currency fluctuations due to the need to transfer funding from NRAO to U. Alberta and then back to NRAO, leading to over 20% loss of those funds from currency fluctuations. Had NRAO been able to transfer those funds internally or if those funds were retained in the original US Dollar denominated value by U. Alberta, those project would have made more progress.

6.3 Effective Collaboration with CASA Software Group

The project has enjoyed a relatively easy integration into the CASA software group because the partnership throughout the project. By working with developers inside the group and coordinating with the head of the CASA group throughout the project, the software had a clear path toward integration. The major outstanding issues include (1) different dependencies on software than the main CASA project and (2) different development and contribution pathway.

The different dependencies primary arise because of the Qt library, where the CASA project uses the older Qt 4 libraries whereas CARTA uses the Qt 5 libraries. Since some key components of the Qt 5 libraries are not back compatible, the CARTA software cannot be used in CASA without installing a separate Qt distribution. Critically, the system version of Qt on some of the RHEL systems supported in CASA is Qt 4. The team investigated downgrading Qt within the CARTA project and determined it would be a substantial effort (3 months of developer time). Since CASA must upgrade to Qt 5 over the next year, the decision has been made that the two packages will operate with separate dependencies over the next year with convergence toward a common set of dependencies.

The CARTA project uses the git version control system with a specific development path as outlined above. This allowed for agile development, which was important for a rapid buildout of the functionality in the software. CASA uses a more rigidly defined set of requirements and the subversion version control system. This is appropriate for a larger “production-scale” software project. Looking toward the future, these two systems will need to be merged, but it is unclear how best to do this. However, the best time for this will likely be concurrent with the alignment of the underlying software dependencies.

Appendix A: Original Project Functionality Deliverables

This appendix summarizes the status and planned future development of functionality for the CARTA tool. This includes a review of the originally proposed functionality and plans for ongoing development.

Component	*Deliverable Feature*	*Status*
D	Provide efficient access to 1, 2, 3D FITS and CASA images and subsets of the same	Complete
D	Provide for access to VO resources via Simple Image Access Protocol over http	Partially Complete (Ongoing)
D	Extract metadata pertaining to images and subimages	Complete
C	Save subsets of data including 1, 2, and 3D subsets, notably the displayed region in a plot	Partially Complete (Ongoing)
C	Regrid data along position and/or spectral directions	Partially Complete (Ongoing)
C	Spectral axis can convert to a variety of units (velocity, frequency, and wavelength)	Complete
C	Users can fit multiple Gaussians, Lorentzians, Poisson distributions and/or polynomials to any 1D data and export results	Partially Complete (Ongoing)
C	Users can fit 2D Gaussians and polynomial surfaces to any 2D data and export results	Partially Complete (Ongoing)
C	Moment maps will be generated for display along any axis (or subinterval along an axis) including the 0th, 1st and 2nd moment using the clipping and masking algorithms present CASA	Partially Complete (Ongoing)
C	Calculate an image histogram showing the intensity range of a 2D or 3D image or ROI	Complete
C	Calculate statistics and uncertainties of 3D data over a ROI including, max., min., sum, mean, median, standard deviation, and median absolute deviation	Complete
C	Calculate statistics and uncertainties of 2D planes of 3D data over a ROI including, max., min., sum, mean, median, standard deviation, and median absolute deviation; Display results as 1D plots	Complete
C	Position axes can be displayed in a variety of coordinate systems [Galactic, Celestial (B1950, J2000, ICRS), Ecliptic]	Complete
C	Slices will be generated through the data set on specified planes, most commonly position velocity slices; Data can be integrated in direction perpendicular to slice	Partially Complete (Ongoing)
V	Adjust margins, background color, and the number of plots	Complete
V	Display multiple images as rasters or contours	Complete
V	Display coordinate grids over 2D position images for different coordinate systems	Complete
V	Provide coordinate grids over 1D plots	Complete

V	Display both a top and bottom axis when multiple spectral plots are displayed	Complete
V	Axis labels and legends can be customized	Partially Complete (Ongoing)
V	1D and 2D plots can be annotated with text and symbols	Complete
V	Selected lines from splatalogue queries can be plotted and labeled on the graph	Partially Complete (Ongoing)
V	ds9 style annotation files can be loaded and displayed on the images	Complete
V	Animate a data cube (or aligned data cubes) by showing successive planes through the image as a movie	Complete
V	Align and simultaneously display multiple 2D and 3D data sets, in particular, aligning multiple data cubes along a common velocity axis	Complete
V	Combine multiple 2D and 3D images using RGB or using master hue/saturation images	Complete
R,L	Ability to save and restore the panel state	Complete
R,L	Raster and contour data can be panned/zoomed	Complete
R,L	Zoom the image histogram based on a percentage or graphical range	Complete
R,L	Draw the image histogram as either a line, outline, or filled	Complete
R,L	Adjust the number of bins in the image histogram	Complete
R,L	Use the image histogram to adjust color map settings for an image.	Complete
R,L	Manipulate color map of viewer including color table, thresholds and stretch	Complete
R,L	Position Tracking: Seeing the RA/DEC, Intensity, Velocity, pixel coordinates etc. at the position under the cursor	Complete
R,L	Position Tracking: When displaying multiple viewer windows, a marker tracks the world coordinate position of the cursor across all images	Complete
R,L	Support for ROI selection in multiple shapes including points, rectangles, ellipses, and polygons	Complete
S	Ability to print in a variety of formats, resolutions, and other options.	Partially Complete (Ongoing)
L,S	Clients will respect a set of stored user preferences, for example, default image scaling, color map, coordinate systems, velocity reference frames, line plot preferences, grid lines	Complete
P	Three-dimensional volume rendering provided through the yt package as an external service	Partially Complete (Ongoing)

Summary of Partially Complete Items and Timeline for Plans

This list summarizes the current status for functionality delivery including the version where the software is planned for delivery. The current version of CARTA is v0.7.

- *Provide for access to VO resources via Simple Image Access Protocol over http* — A template data loading plugin has been established that can provide this functionality. Planned deployment: Post v1.0.
- *Save subsets of data including 1, 2, and 3D subsets, notably the displayed region in a plot* — This has depended on the Region of Interest code and can now be deployed. A rough plan has been established but no implementation has taken place. Planned deployment: v0.9.
- *Regrid data along position and/or spectral directions* — This has required developing a regridding engine, implemented as a plugin. Development is in progress and functionality will be refined. Planned deployment: v0.8.
- *Users can fit multiple Gaussians, Lorentzians, Poisson distributions and/or polynomials to any 1D data and export results* — Gaussian fitting is implemented. The other functions and implementations are in development. Planned deployment: v0.8.
- *Moment maps will be generated for display along any axis (or subinterval along an axis) including the 0th, 1st and 2nd moment using the clipping and masking algorithms present CASA* — The connection to use CASA tools as plugins has been created for the image statistics functionality and can be adapted for this work. Planned deployment: v0.9. Clipping and Masking are being developed over the next year at U. Alberta. Planned deployment: v1.0
- *Slices will be generated through the data set on specified planes, most commonly position velocity slices; Data can be integrated in direction perpendicular to slice* — Displaying slices along different coordinate axes has been developed, but it remains to connect in the CASA position-velocity slice functionality to these tools. Planned deployment: v0.9.
- *Axis labels and legends can be customized* — Some functionality has been implemented for font, color, and style, but text entry has relied on recently implemented functionality. Planned deployment v0.8.
- *Selected lines from splatalogue queries can be plotted and labeled on the graph* — Like VO queries, this information retrieval has a ready connection though the plugin system but has not been a priority. Planned deployment: Post v1.0.
- *Ability to print in a variety of formats, resolutions, and other options* — Simple export has been developed but the results are not suitable for journal quality publication. After consultation with our user base, the solution as been chosen too make this export to a Python script and associated data bundle, which can then be edited by expert users or rendered in place. Planned deployment: v1.0.
- *Three-dimensional volume rendering provided through the yt package as an external service* — Originally intended as a test of the plugin system, this functionality can be readily implemented. However, it has been deemed less of priority by our user reviews and deferred for future development. Planned deployment: Post v1.0.