The Legend of CDAT
A Link to the Past

Matthew B. Harris, Sam B. Fries, Dean N. Williams, Sterling A. Baldwin, James W. Crean, Bryce J. Sampson, Edward M. Brown, Anna Paula M. Pawlicka,

***Abstract*—Those who cannot learn history are doomed to repeat it. The Climate Data Analysis Tool (CDAT) has had many names since its creation in 1989, but its purpose has remained the same; to provide capabilities needed for validating, comparing and diagnosing climate model behavior [1]. CDAT has grown from a small visualization package to a complex piece of desktop software with many components, including VCS, VTK, VisIt and many others. CDAT's primary users are climate scientists, and as CDAT expands to run in the browser, the user experience needs to be just like the versions our users have loved and used over the last twenty-seven years.**

***Index Terms*—UV-CDAT, CDAT, vCDAT, CDATWeb**

I. Introduction

THE Program for Climate Model Diagnosis and Intercomparison (PCMDI) was established in 1989 at the Lawrence Livermore National Laboratory (LLNL) with the principal mission to develop improved methods and tools for diagnosis, validation and intercomparison of global climate models. [2]

The massive amounts of data produced by decadal simulations with climate models makes it imperative that a utility exists for easily visualizing the data with a number of different techniques. It is virtually impossible to gain insight into these models' behavior without graphic representations of the data, and no single display technique is best for recognizing or for explaining all features of their behavior. In comparing decadal simulations from tens of models, it is important to have a functionality to perform computations on the data, and, since most of these models use different grids, to also have the ability to perform grid transformations. These additional capabilities not only make it easier and quicker to browse through terabytes of data in the comparison of models, but it also removes the need for intermediate data files [1][2].

Visual displays are used not only for browsing data, but also for documenting a simulation and presenting results, so it is necessary that the visualization software allow as much user control over the display as is feasible, and provide a method for producing hardcopy visuals without exhaustive human interaction [1].

The design of this software was driven by the need to provide a platform upon which an ever-growing selection of methods for displaying one-, two- and three-dimensional data could be incrementally developed with extensive user control over all elements the process. This software provides the user with the capability either interactively or from a script. It provides scripting functionality for saving the instantaneous state-of-the-system for late recovery, and the ability to save a continuous script of an interactive session that can be edited (or not) and re-ran [1].

The CDAT software is a suite of tools for storing, diagnosing and visualizing climate data. It may also be useful for general scientific computations and graphics [3].

Manuscript received July 1, 2016; revised July XX, 2016. This work was funded by the UV-CDAT project at Lawrence Livermore National Laboratory as a process for merging the CDATGUI and CDATWeb teams.

M. B. Harris is a Computer Scientist, Mathematical Programmer in the AIMS (Analytics and Informatics Management Systems) team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-423-8978 email: harris112@llnl.gov).

S. B. Fries is a Computer Scientist, Mathematical Programmer in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-422-5859 email: fries2@llnl.gov).

D. N. Williams is a Principal Investigator and the Team Leader of the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-423-0145 email: williams13@llnl.gov).

S. A. Baldwin is a Computer Scientist, Mathematical Programmer in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (phone: 925-423-8954 email: baldwin32@llnl.gov).

J. W. Crean is a Computer Science Intern in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (email: creanjames@gmail.com).

B. J. Sampson is a Computer Science Intern in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (email: sampson.bryce@yahoo.com).

E. M. J. Brown is a Computer Science Intern in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (email: ebrown37@mail.csuchico.edu).

A. M. P. Pawlica is a Computer Science Intern in the AIMS team at Lawrence Livermore National Laboratory, Livermore, CA 94550 USA (email: anna.maulle@gmail.com).

II. Past

1989: PCMDI software developers started working on a diagnosis and visualization tool called MapView for use on their machines, which were equipped with Apollo 20MHz processors and 2MB of RAM [4]. This first-generation version of CDAT was a command line tool written in C [5], Fortran [6], and the Graphical Kernel System (GKS) [7]. It was able to produce one- and two- dimensional graphics.

1992: MapView was given a graphical user interface (GUI) [8] and renamed to Visualization Command Systems (VCS). VCS was upgraded from GKS to XGKS [9] for the windowing system support. The new GUI was written in C and SunView [10]. VCS helped expand the number of users by providing both command-line and GUI usage.

1994: Visualization Command Systems (VCS) was renamed to Visualization Control Systems (VCS) and given upgrades in programming languages. VCS started migrating to Python [11] and the windowing system migrated to Motif [12]. These changes were vital to insuring the software

would remain extensible and maintainable in a time of rapid changes in the programming community.

1996: VCS became the Climate Data Analysis Tools (CDAT). The major change was introducing the concept of several packages; all of the previous work became the VCS package, and the CDMS [13] and NumPy [14] packages were added to CDAT. The GUI was re-written in Tcl/TK [15]. Users at this time had started talking about other types of software and to ensure continued usage of CDAT, the concept of plug-and-play was developed, which allowed for other packages to be added to CDAT.

1998: The GUI became its own package, vCDAT, which was added to CDAT. This change brought the addition of the PMW [16] to Tck/TK for the GUI. This meant hardcore users would not have to install the GUI, making set up easier. It also gave users the freedom to install only the packages they wanted or needed, giving the system more flexibility.

2004: CDAT's windowing system was replaced by Cairo [17]. Two new packages were added: CDUtil [18] and GenUtil [19]. These additions not only kept current users happy, but again helped to increase the number of users.

2009: CDAT again made a core language addition of the Visualization Tool Kit (VTK) [19]. With the addition of VTK, CDAT was able to offer its users interactive three-dimensional plots of their data, another impressive feature for the user base.

2011: The beginning of a new era. CDAT up to this point had its six core packages (vCDAT, VCS, CDMS NumPy CDUtil, and GenUtil); with the addition of eight new packages (matplotlib [21], VISUS [22], ESG [23], R [24], VTK [19], VisTrails [25], VisIt [26], SciPy [27], and CMOR [28]), the Ultrascale Visualization Climate Data Analysis Tools (UV-CDAT) [20] was released. Our users now had a wide selection of tools at their disposal, available not only from the command line but also from the new GUI, which was completely rewritten in Qt [29].

2013: With large growth in climate data, ESG was forced to expand to the Earth System Grid Federation because storing all the data in one location was no longer feasible. UV-CDAT had a package update (ESG to ESGF). The development of UV-CDATLive was started. Unlike UV-CDAT (the stand-alone version), UV-CDAT Live did not require any software installation. By using UV-CDAT Live, scientists could begin their data analysis by simply opening a web browser [30].

2014: UV-CDAT added two more packages: UVCMetrics and PMP. These two packages gave UV-CDAT the ability to run and display model diagnostic from standard diagnostic libraries of that time.

2015: After a year of hibernation UV-CDATLive transitioned to CDATWeb, further hardening the client/server architecture and capturing the capabilities of the desktop software UV-CDAT.

III. PRESENT

UV-CDAT is an integrated framework that provides an end-to-end solution for management, analysis, and visualization of "ultrascale" datasets. Figure 1 shows version 2.4 of the desktop GUI. The project is distributed as a bundle of Python packages, which can be installed into any standard environment. This means that the usage of UV-CDAT is not limited to desktop computers, but extends to web servers, supercomputers, and more.

UV-CDAT has recently been integrated as part of the ESGF Compute Working Team backend [32]. UV-CDAT provides re-gridding, re-projection, and aggregation tools directly installed as a component of the ESGF data node. This eliminates or substantially decreases data movement by reducing the size of the data before transferring it to the scientist for further analysis. This allows for larger and more complex datasets to become more accessible, enabling in-depth and complicated analysis. This integration of UV-CDAT provides a turnkey application for building complex data analysis and visualization workflows. These workflows can use various predefined components for transformation and analysis of data, gathering data from external sources, and visualizing the results of computations. Provenance data is captured throughout the process, ensuring reproducibility.

UV-CDAT is preparing for a large 3.0 release. Two big features of this release were to be a new redesigned refreshed GUI with CDATGUI and a CDATWeb with minimum viable functionality.

CDATGUI was designed to address some of the limitations of the current VisTrails-based UV-CDAT GUI. Maintaining the current GUI is difficult, due to the extremely high level of abstraction that the VisTrails framework operates at. Numerous bugs occurred at the interoperation layer between VisTrails and the various CDAT libraries. The complexity of the interoperation also made it difficult to fully expose the flexibility of the CDAT libraries, and as new features were added to CDAT, they were not exposed within the GUI due to these challenges. CDATGUI sought to remedy these issues by providing a minimal amount of abstraction between the user and the CDAT libraries. It was built from the ground-up with a focus on user experience, and a testing infrastructure to ensure that new development wouldn't break existing features. It also included a new provenance tracking engine that allowed for project persistence in the form of python scripts. An example of the CDATGUI is shown in figure 2.

CDATWeb is a browser-based version of UV-CDAT, which leverages all of the existing libraries. It uses the "VTK-Web" infrastructure to directly display visualizations rendered on a server using UV-CDAT's VTK-based implementation of VCS. Visualizations are parameterized in the browser, and those parameters are sent to the server, via a remote procedure call over a WebSocket, which builds the visualization, renders it to a PNG file, and sends the image back over the socket. Data for the visualization can be obtained using either ESGF [23] (which is directly streamed via OpenDAP), or read off of a filesystem co-located with the visualization server. An example of CDATweb is shown in figure 3.

CDATWeb is currently a Django [33] application, using Autobahn [34] to make the connection to the visualization server where UV-CDAT is deployed. Figure 4 outlines the full architecture.

IV. Future

The volume of output being generated by each run of climate models is in the petabytes, and growing [35]. The infrastructure to store this data already exists, and UV-CDAT is already in use to provide client-consumable reductions of the data [32]. UV-CDAT needs to scale to provide access to that data in a simple fashion for climate scientists, and give all of the analysis tools that they need to actually do their jobs. To do so, we need to evolve UV-CDAT into the software we need tomorrow, rather than just handling what we need today. So, rather than developing multiple user interfaces to the same library, and splitting our development team as well as our user base, we've decided to combine CDATGUI and CDATWeb into a single project. This new project will be called "vCDAT 7.0", hearkening back to the original CDAT GUI (the version number is roughly accurate, if UV-CDAT is considered "CDAT 6.0").

The initial goals of vCDAT are fairly modest; we want to build the minimum viable user interface, with as little new work on the CDAT libraries as possible. It will leverage all of the existing tooling, for data management, visualization, and analysis. The only "new" feature will be that it opens inside a browser, rather than in a standard window. A small RESTful web server will provide access to the existing CDAT functionality (statistics functions, metadata reading, visualization parameters…), and the VTK-web infrastructure will be utilized to display the visualizations. We'll run a web server locally on the user's computer, and the entire experience will be seamless.

Once we have a working local experience, it'll be time to work on getting it running remotely. At this point, we'll be able to leverage the ESGF Compute Working Team's efforts and integrate with their services. We'll also be able to install the server in data centers adjacent to the ESGF data nodes, leveraging the high throughput internal network of those data centers to transfer high resolution model output to the vCDAT server for processing and visualization. Downscaled data will be able to be sent directly to the user's machine to be rendered client side, and we will also be able to do server-side rendering on high resolution data.

To make the transition from server-side rendering to client-side as painless as possible, the UV-CDAT team has architected a JavaScript framework that will allow visualization methods to be switched from server-side to client one at a time. As we do so, we can use whatever the most appropriate rendering technique for the client is (Canvas, WebGL, SVG, etc.), using whichever JavaScript framework makes the task easiest.

V. Conclusion

CDAT has a long history of providing the tools that climate scientists need to perform their jobs. Looking back at the last 27 years of climate science, we can easily see trends of data becoming bigger and more complex, and our software growing to meet the challenges of the day. Right now, our users have UV-CDAT installed on a wide variety of platforms, from desktop, to server, to supercomputer, and we support their usage in each of those locations. Looking forward, we can see that to remain on the cutting edge of climate research, we have to get ahead of the curve and start providing the tools that tomorrow's scientists will need, today. vCDAT is our plan to do just that.

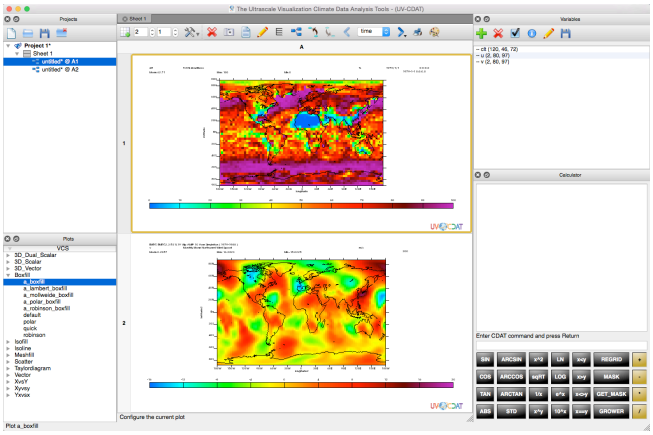

Fig. 1. Example of the UV-CDAT v2.4 released GUI.

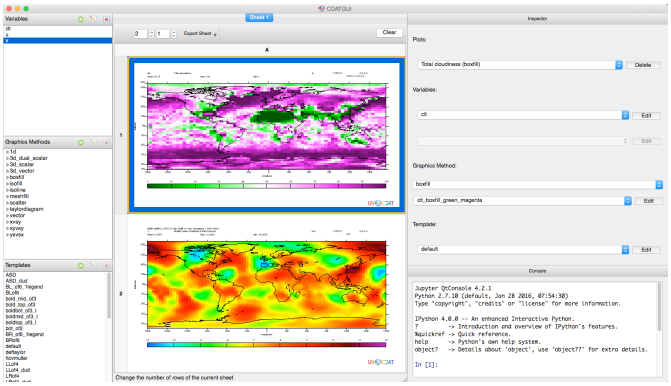

Fig. 2. Example of the unreleased CDATGUI.

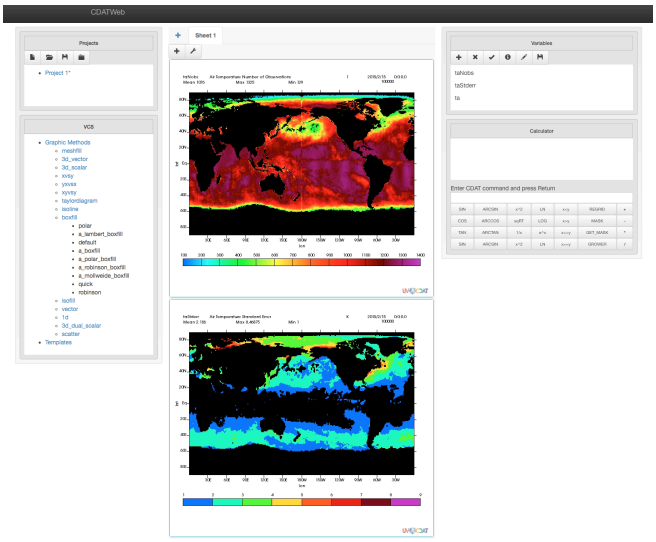

Fig. 3. Example of the CDATWeb's alpha released GUI.

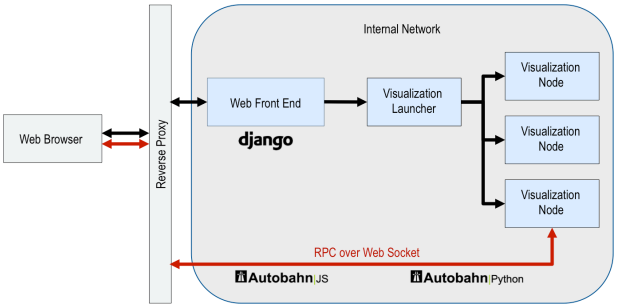

Figure 4

ACKNOWLEDGMENT

We would like to thank each member of our team for all their hard work and dedication to the project and its goals: Dean Williams, Charles Doutriaux, James McEnerney, Aashish Chaudhary, Jonathan Beezley, Dan Lipsa and every member of the CDAT project.

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DEAC52-07NA27344.

LLNL-CONF-696844

REFERENCES

[1] Williams, Dean N., and Robert L. Mobley. The PCMDI Visualization and Computation System (VCS): A Workbench for Climate Data Display and Analysis. Rep. no. UCRL-ID-116890. Springfield: National Technical Information Service, 1994. Print.

[2] Williams, Dean N. The PCMDI Software System: Status and Future Plans. Rep. 1997. Print.

[3] Visualization Control System: Python Command Line and Application Programming Interface. Livermore: Regents of the U of California, 2000. Print.

[4] Network World 3.34 (1986). Books.google.com. Web. 28 June 2016.

[5] "C." ISO/IEC. Web. 28 June 2016. <http://www.open-std.org/jtc1/sc22/wg14/>.

[6] "All Things Fortran." The FORTRAN Company. Web. 28 June 2016. <http://www.fortran.com/>.

[7] Duce, DA, and FRA Hop Good. "The Graphical Kernel System (GKS)." Computer-Aided Design 19.8 (1987): 364-409. Dl.acm.org. Web. 28 June 2016.

[8] Tuck, Mike. The Real History of the GUI. 2001. Web. 28 June 2016.

[9] "XGKS: Graphical Kernel System for the X Window System." XGKS. Web. 28 June 2016. <http://xgks.sourceforge.net/>.

[10] Salus, Peter H. A Quarter Century of UNIX. Reading, MA: Addison-Wesley Pub. 1994. Print.

[11] "Welcome to Python.org." Python.org. Web. 28 June 2016. <https://www.python.org/>.

[12] Brain, Marshall. Motif Programming: The Essentials-- and More. Burlington, MA: Digital, 1992. ISBN: 1555580890. Print.

[13] "CDMS Manual Table of Contents." CDMS Table of Contents. Web. 28 June 2016. <http://uv-cdat.llnl.gov/documentation/cdms/cdms.html>.

[14] "NumPy." NumPy — Numpy. Web. 28 June 2016. <http://www.numpy.org/>.

[15] "Tcl Developer Site." Tcl Developer Site. Web. 28 June 2016. <http://tcl.tk/>.

[16] "Pmw 1.3." Pmw Megawidgets 1.3. Web. 28 June 2016. <http://pmw.sourceforge.net/doc/index.html>.

[17] "CDAT Utilities." UV-CDAT Utilities Chapter 1. Web. 28 June 2016. <http://uvcdat.llnl.gov/documentation/utilities/utilities-1.html>.

[18] "General Utilities." UV-CDAT Utilities Chapter 2. Web. 28 June 2016. <http://uvcdat.llnl.gov/documentation/utilities/utilities-2.html>.

[19] "VTK-Enabled Applications." VTK. Web. 28 June 2016. <http://www.vtk.org/>.

[20] "Ultrascale Visualization." UV-CDAT. Web. 28 June 2016. <http://uvcdat.llnl.gov/index.html>.

[21] "Introduction." Matplotlib: Python Plotting — Matplotlib 1.5.1 Documentation. Web. 28 June 2016. <http://matplotlib.org/>.

[22] "ViSUS." ViSUS. Web. 28 June 2016. <http://atlantis.sci.utah.edu/>.

[23] "Earth System Grid Federation." ESGF Home Page. Web. 28 June 2016. <http://esgf.llnl.gov/>.

[24] "The R Project for Statistical Computing." R. Web. 28 June 2016. <https://www.r-project.org>.

[25] "VisTrails." VisTrails. Web. 28 June 2016. <http://www.vistrails.org/index.php/Main_Page>.

[26] "VisIt." VisIt. Web. 28 June 2016. <https://wci.llnl.gov/simulation/computer-codes/visit>.

[27] "SciPy." SciPy.org. Web. 28 June 2016. <http://www.scipy.org/>.

[28] "Climate Model Output Rewriter." CMOR. Web. 28 June 2016. <http://pcmdi.github.io/cmor-site/>.

[29] "Qt." Qt. Web. 28 June 2016. <http://www.qt.io/>.

[30] Williams, Dean N. Department of Energy's Biological and Environmental Research Ultra-scale Visualization Climate Data Analysis Tools (UV-CDAT) Three-year Comprehensive Report. Rep. 2013. Print.

[31] "American Meteorological Society." Metrics for the Diurnal Cycle of Precipitation: Toward Routine Benchmarks for Climate Models: Journal of Climate: Vol 29, No 12. Web. 28 June 2016. <http://journals.ametsoc.org/doi/10.1175/JCLI-D-15-0664.1>.

[32] Harris, Matthew B. CDATWeb. Dec. 2015. Poster. ESGF Face-to-Face Conference, Monterey CA.

[33] "Django." The Web Framework for Perfectionists with Deadlines. Web. 28 June 2016. <https://www.djangoproject.com/>.

[34] "Autobahn JS." Autobahn JS Documentation. Web. 28 June 2016. <http://autobahn.ws/js/>.

[35] Bernholdt, David, S. Bharathi, D. Brown, K. Chanchio, M. Chen, A. Chervenak, L. Cinquini, B. Drach, I. Foster, P. Fox, J. Garcia, C. Kesselman, R. Markel, D. Middleton, V. Nefedova, L. Pouchard, A. Shoshani, A. Sim, G. Strand, and D. N. Williams. "The Earth System Grid: Supporting the Next Generation of Climate Modeling Research". Proceedings of the IEEE, 93 (3), p 485-495, 2005.

[36] Matthew Harris, “Webengine”, Proceedings of the World Congress on Engineering and Computer Science 2014, Vol. I, WCESC 2014, 22-24 October, 2014, San Francisco, USA, pp. 131-135, http://www.iaeng.org/publication/WCECS2014/ ISBN: 978-988-19253-6-7

[37] Matthew B. Harris, Samuel B. Fries, Sterling A. Baldwin, and Dakotah S. M. Webb, "Nerd Herding: Practical Project Management in the Field" Proceedings of The World Congress on Engineering and Computer Science 2015, Vol. I, WCECS 2015, 21-23 October, 2015, San Francisco, USA, pp123-126, http://www.iaeng.org/publication/WCECS2015/ ISBN: 978-988-19253-6-7