UCRL-ID-129074

PCMDI Report No. 44

THE PCMDI SOFTWARE SYSTEM: STATUS AND FUTURE PLANS

Dean N. Williams

Program for Climate Model Diagnosis and Intercomparison
Lawrence Livermore National Laboratory, Livermore, CA, USA

December 1997

PROGRAM FOR CLIMATE MODEL DIAGNOSIS AND INTERCOMPARISON
UNIVERSITY OF CALIFORNIA, LAWRENCE LIVERMORE NATIONAL LABORATORY
LIVERMORE, CALIFORNIA 94550

The PCMDI Software System: Status and Future Plans

Dean N. Williams

Program for Climate Model Diagnosis and Intercomparison
Lawrence Livermore National Laboratory
Livermore, California USA

December 1997

Preface

This report on the status and future plans of the Program for Climate Model Diagnosis and Intercomparison (PCMDI) software focuses on software that PCMDI is currently distributing to the global climate community or has plans to distribute in the near future. Other secondary software necessary for the distributed software to work will not be included in this document. If anyone wishes to acquire PCMDI software source code, they may do so by filling out the collaboration agreement located on the web (http://www-pcmdi.llnl.gov/software), which identifies software ownership (i.e., U.S. Department of Energy) and prohibits any selling of this software.

The development of PCMDI software would not have been possible without input from the entire PCMDI staff, who have indicated to the author the type of software needed to do their jobs well. Among PCMDI's focal points are the intercomparison projects such as the Atmospheric Model Intercomparison Project (AMIP) and the Coupled Model Intercomparison Project (CMIP). Current and future software are designed to accommodate the needs of these intercomparison projects, and it is hoped that the software is flexible enough to assist in other areas of global climate modeling.

It is the desire of the author that anyone interested in PCMDI software provide input to their development so that PCMDI can better serve its constituents in the global climate research community. Readers are encouraged to e-mail or write to:

Program for Climate Model Diagnosis and Intercomparison (PCMDI)
Lawrence Livermore National Laboratory
P.O. Box 808, L-264
Livermore, California (USA) 94550
E-mail: williams@pcmdi.llnl.gov
Tel: (510) 422-7626
Fax: (510) 422-7675

Table of Contents

Abstract

This report describes the current status and future plans of PCMDI's software products. A complete description of each product is provided, including the product's problem statement, purpose, requirements, design diagram, current status, future development, developers, contributors, and off-site collaborators. While each software product can be executed as an independent process, it is important to note that all products work together in the complete PCMDI Software System: a suite of software tools facilitating the storage, diagnosis, and visualization of climate data.

Acknowledgments

Input from the entire PCMDI staff was greatly appreciated in preparing this document. Special thanks are due the PCMDI software design and development team which consists of Jerry Potter, Karl Taylor, Clyde Dease, Charles O'Connor, Susan Marlais and Bob Drach, without whom this document would not have been possible. In addition, special thanks to Tom Phillips and Larry Gates for their editorial assistance, and to Anna McCravy for her substantial graphics and layout contribution.

This work was performed under the auspices of the U.S. Department of Energy, Environmental Sciences Division, by the Lawrence Livermore National Laboratory under Contract W-7405-ENG-48.

Acronyms Glossary

ACL	Advanced Computing Laboratory
AMIP	Atmospheric Model Intercomparison Project
API	Application Programmer's Interface
CCM	Community Climate Module
CDAT	Climate Data Analysis Tool
CDMS	Climate Database Management System
cdunif	Common Data UNIform File Reader
CGI	Common Graphics Interface
COARDS	Cooperative Ocean/Atmosphere Research Data Service
CMIP	Coupled Model Intercomparison Project
CVS	Concurrent Version System
DDI	Data and Dimension Interface
DEC	Digital Equipment Corporation
DRS	Data Retrieval and Storage
GUI	Graphical User Interface
HDF	Hierarchical Data Format
HP	Hewlett Packard
IBM	International Business Machines
LANL	Los Alamos National Laboratory
LATS	Library of AMIP Data Transmission Standards
LLNL	Lawrence Livermore National Laboratory
NCAR	National Center for Atmospheric Research
NERSC	National Energy Research Supercomputing Center
netCDF	network Common Data Form
PC	Personal Computer
PCMDI	Program for Climate Model Diagnosis and Intercomparison
PMIP	Paleoclimate Modeling Intercomparison Project
PSQL	PCMDI Structured Query Language
PSS	PCMDI Software System
PVM	Parallel Virtual Machine
QC	Quality Control
QCS	Quality Control Software
SGI	Silicon Graphics, Inc.
SQL	Structured Query Language
VCS	Visualization and Computation System
VPOP	View POP

1. Introduction and Purpose

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 the diagnosis, validation and intercomparison of global climate models. While playing a leading role in the development of climate diagnostic tools, PCMDI has entered into collaborative agreements with the National Center for Atmospheric Research (NCAR) and the Los Alamos National Laboratory (LANL) for the provision of atmospheric and oceanic diagnostic functions that will be utilized in PCMDI's software products.

This document will focus primarily on the software tools necessary for PCMDI to sustain and meet its short-term and long-term missions. The job assignment(s) envisaged for each PCMDI software development staff member are also described. Here, we will only attempt to project PCMDI software development out for one to two years, with project time lines presented only where these now can be anticipated. As with all projected plans, it is anticipated that, with changes in technology and in the PCMDI staff, future modifications of this document may be needed.

2. Current and Future Distributed PCMDI Software

Processing intercomparison model data (i.e., Atmospheric Model Intercomparison Project (AMIP), the Coupled Model Intercomparison Project (CMIP), and the Paleoclimatic Model Intercomparison Project (PMIP) is an important part of PCMDI. All members of the PCMDI software team are directly or indirectly involved in this endeavor and give these projects their primary focus. To fully understand the data processing element of PCMDI see, "Processing and Distributing Intercomparison Data," by Peter Gleckler and Dean N. Williams (in preparation). Reading and writing of intercomparison model data are the software team's highest priority. Thus, it involves the further development of the Common Data UNIform File (cdunif) reader and the Library of AMIP Data Transmission Standards (LATS), both of which are at the core of PCMDI's software.

As an important component of PCMDI's work in the development, testing, validation, and intercomparison of global climate models, PCMDI has developed and distributed a suite of software tools for the storage, diagnosis, and visualization of data. Section 2.1 below presents a short list of the software products that are currently being used at PCMDI and distributed to the climate community at large, or that are under design and development for future release. Section 2.2 describes each PCMDI software product in more detail. Section 3 describes web development and software documentation in detail. Section 4 lists all of the platforms that PCMDI software supports. Section 5 describes the software collaboration agreements with outside organizations. Section 6 describes the needed code repository. Finally, section 7 concludes with an overview of PCMDI's primary software goal, the PCMDI Software System.

To further PCMDI's principal mission, development of the Climate Data Analysis Tool (CDAT), the Climate Database Management System (CDMS), and the Visualization and Computation System (VCS) are also high on the list of priorities, but development of other special-purpose software will also be needed. It is the aim of the PCMDI software development team that, with the future development of CDAT, CDMS, and VCS, most special-purpose software will be eliminated. Thus, the need to maintain so many software products will be reduced.

2.1 Short-List of Software Products

Name	Acronym	Description	Distributed	Version
Climate Data Analysis Tool	CDAT	Utilizes an interpreted language to manipulate data and provide climate scientists with diagnostic, statistical, and regridding routines.	Internally	Beta
Climate Database Management System	CDMS	Designed to automatically locate and extract metadata (i.e., variables, dimensions, grids, etc.) from PCMDI's collection of model runs and analysis files.	Internally	Beta
Common Data UNIform File	cdunif	A library of input functions for accessing netCDF, HDF, DRS, GrADS/GRIB and VPOP data files.	Externally	1.7
Data and Dimension Interface	DDI	A Graphical User Interface (GUI) that reads and writes available file formats, and sends data to a variety of visualization systems.	Externally	1.2
Data Retrieval and Storage	DRS	A library that supports direct access I/O, and multi-dimensional array variables.	Externally	1.5
EzGet		A FORTRAN application programmer's interface (API) for reading various data file formats via cdunif, and for performing grid transformations, data masking, and some calculations.	Externally	1.1
Library of AMIP Data Transmission Standards	LATS	A collection of software routines to output gridded data in either netCDF or GRIB formats for the Atmospheric Model Intercomparison Project (AMIP).	Externally	1.1
Quality Control Software	QCS	Software that checks the correctness of AMIP model data.	Internally	Beta
Visualization & Computation System	VCS	A graphics software package that allows the display, animation, and manipulation of scientific data.	Externally	2.7

The PCMDI software products can be represented in levels as shown in Figure 1. The figure shows the foundation-, intermediate-, and top-level design of the PCMDI software system. Although each of the top-level applications are standalone, they can also be used by other PCMDI applications directly or indirectly, thus making the system flexible to individual user preferences.

Levels of PCMDI Software System

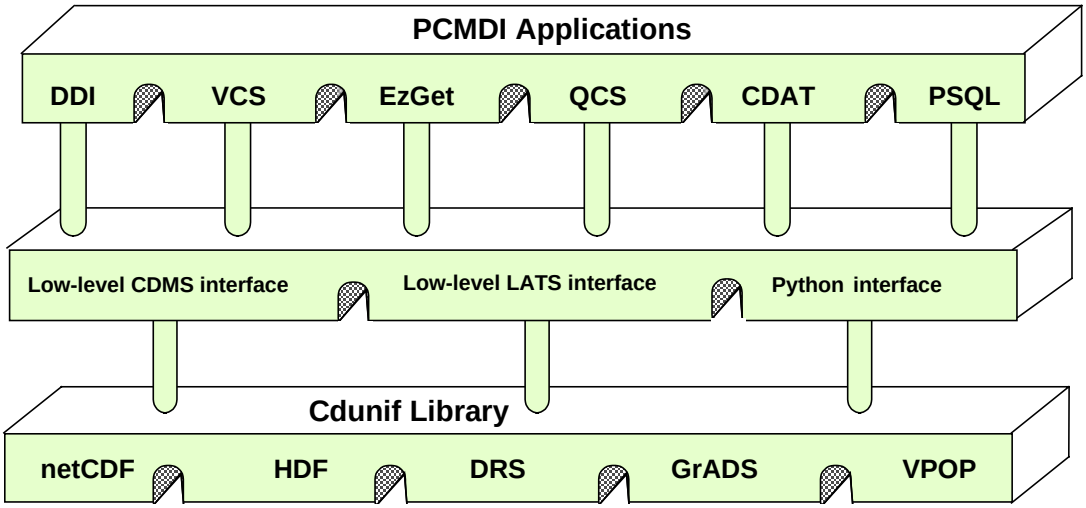

Figure 1. A building block look at the design structure of the PCMDI Software System.

2.2 Detailed Description of PCMDI Software Products

2.2.1 Climate Data Analysis Tool (CDAT)

CDAT Problem Statement

A basic problem facing climate scientists is not the absence of software to analyze data, but rather a shortage of interrelated diagnostic tools that are consistent, flexible, portable, adaptable, efficient, share data, and easy to use. Consequently, many scientists are writing their own programs to ingest, manipulate and display their data. Redundancy in such efforts diverts time for the debugging and enhancing of these programs that otherwise would be spent on research. The resulting software is often not user-friendly, reusable, or portable, and does not promote common standards within the climate community.

Another obstacle to sharing analysis software is the wide variety of data file formats that are in use, making it necessary to write programs to convert data to a user's preferred file format and conventions. This data conversion requires additional expenditure of efforts on testing and quality assurance. Modular and interrelated software to perform such tasks transparently would thus be a valuable asset, allowing climate scientists to spend their time analyzing rather than processing data.

CDAT Purpose

CDAT provides the climate scientist with an easy and fast method to read different file formats (via cdunif) and to analyze data. It includes a set of predefined functions to allow the user to manipulate the data and send the output to a file which can be viewed as an image, or as a collection of animated images. CDAT also has a gradual learning curve, allowing the novice user to quickly obtain useful results.

CDAT Requirements

CDAT uses software that has been developed mainly at PCMDI, NCAR, and LANL. Using Python--an interpreted, object-oriented scripting language--as its centralized control module, CDAT is designed to interact with other PCMDI software products. By using Python, CDAT will fulfill the necessary requirements: a standard mathematical library and logical operators, graphics capabilities, scripting language, API, application and user extensibility, and World-Wide Web access.

PCMDI's atmospheric scientists will provide CDAT with diagnostic, statistical, regridding, and units conversion routines. NCAR's CCM model diagnostic routines and LANL's ocean model routines will also be accessible from within CDAT.

CDAT will work as a standalone process and also will be accessible from within CDMS and VCS. All CDAT, CDMS and VCS scripts will be, for the most part, interchangeable.

Users outside PCMDI will be able to access stored data by means of the World Wide Web. Web browsers, such as Netscape and Internet Explorer, will allow outside collaborators to search PCMDI's database, access variables, manipulate data, and display the necessary information.

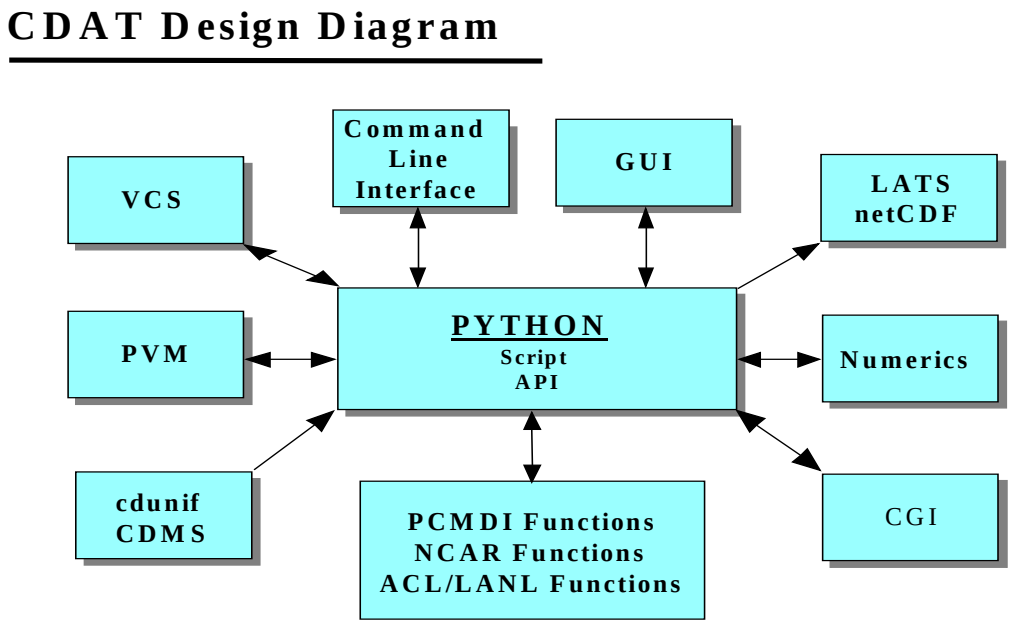

Figure 2. Conceptual view of CDAT's modular design

CDAT Design Diagram Modules

Module	Definition/Function
Python	An interpreted, object-oriented scripting language that allows the user to call CDAT scripts and use CDAT API commands.
GUI	Allows the user to access CDAT from a point-and-click environment. From this environment the user also has the ability to access the Command Line Interface.
Command Line Interface	Allows the user to interact with CDAT by entering commands or creating, modifying, and/or running scripts. This interface also allows the Colormap Editor and Animation Interface GUIs to appear on the screen.
LATS/ netCDF	LATS allows users to save manipulated data in the PCMDI convention standards. LATS format options for saving data are either netCDF or GrADS/GRIB. For those not wishing to save data in PCMDI's convention standards, the option of raw netCDF output is provided.
Numerics	Supplies CDAT with trigonometric functions, fast Fourier transforms, eigenvectors, etc.
CGI	This interface provides CDAT with the ability to browse PCMDI's database, access variables, manipulate data, and display the results via a Web browser.
PCMDI	The most commonly used PCMDI diagnostic and statistical functions included in CDAT.
NCAR	NCAR's CCM atmospheric diagnostic functions.
ACL/ LANL	ACL/LANL's ocean diagnostic functions.
cdunif	Allows CDAT access to many types of data file formats (i.e., netCDF, HDF, DRS, GrADS, and GRIB).
CDMS	Allows CDAT to automatically locate and extract physics quantities (i.e., variables, dimensions, grids, etc.).
PVM	PVM is a software system that allows you to combine a number of computers which are connected over a network into a parallel virtual machine. This machine can consist of computers with different architectures, running different flavors of the UNIX operating systems and can still be treated as if it were a single parallel machine.
VCS	Allows CDAT users the ability to display and animate scientific data.

CDAT Current Status

Module	Status
Python	Completed.
Command Line Interface	Completed, may need modifications.
Graphical User's Interface	Needs design and development.
LATS	Completed.
netCDF	Completed.
Numerics	Modifications are necessary.
Common Graphical Interface	Needs design and development.

PCMDI Functions	Under development, currently implementing regridder.
NCAR Functions	Needs design and development.
LANL Functions	Needs design and development.
cdunif	Connections to completed.
CDMS	Needs design and development.
VCS	Completed, may need modifications.
Documentation	Incomplete, further documentation is necessary.

CDAT is scheduled for beta release in August, 1997. Beta release will feature: Python Numeric Functions, Command Line Interface, LATS, netCDF, cdunif, and VCS modules. It also will contain a PCMDI function and limited documentation.

2.2.2 Climate Database Management System (CDMS)

CDMS Problem Statement

PCMDI is facing a data storage dilemma. Currently, climate scientists are storing data on PCMDI's UNIX server disk space (i.e., /scratch) and at the National Energy Research Supercomputing Center (NERSC) on PCMDI's storage archive. In addition, scientists also are storing data there on local disk space. Having data in various locations can be confusing and data is often misplaced and lost, costing scientists valuable time searching for or regenerating data.
Having access to someone else's data is also a problem, as it is often hard to understand another methodology for storing or naming data and their attributes. Also, once the data is located, it is often in a different file format, dimension arrangement, or naming convention.

All of the above are a hindrance to effectively using the data, limiting productivity and ability to communicate on all levels of the data.

CDMS Purpose

CDMS is a software tool designed to automatically locate and extract metadata (i.e., variables, dimensions, grids, parameters and attributes, etc.) from PCMDI's collection of model runs and analysis files. Its function will be to keep track of various data files, their contents, and the format (i.e., netCDF, HDF, DRS, GrADS, GRIB) of each data file. CDMS will allow the user to: conceptually reorganize stored data files; navigate and extract data by use of quantity names and/or attributes; conceptually rename file quantities and join quantities that are physically stored across multiple servers and/or files; and reorder and select arbitrary subsets as CDMS transfers data from disk to memory.

CDMS is also designed to include an interactive tool, as well as a callable library, for the maintenance, grouping, locating, and scanning of PCMDI's data analysis files. It also will be used to 'override' incorrect file coordinate values and 'insert' new attribute tags/comments. In addition, users will be able to 'combine' quantities spread across several files into an outwardly appearance single multi-dimensional array, all without physically having to move or modify the actual data analysis files.

In short, CDMS is designed to be the logical storage structure for PCMDI's data management system, providing a relational database to retrieve arbitrary databases, datasets, dimensions, grids, variables, parameters, and attributes. With the completion of CDMS, development of standards for data archiving, access and exchange will lead to the system integration and documentation of all model and observational climate data sets envisaged by PCMDI.

CDMS Requirements

CDMS is written entirely in the C programming language. It provides access to databases through a scripting language, command line user interface, an API, and a GUI. The Structured Query Language (SQL), the accepted universal database language, will be adapted for PCMDI's purposes (designated as PSQL) to allow users to describe a CDMS data set in an intelligible form.

CDMS will work as a standalone process and also will be seamlessly accessible from within CDAT, VCS and other PCMDI applications. All CDAT, CDMS and VCS scripts will be, for the most part, interchangeable.

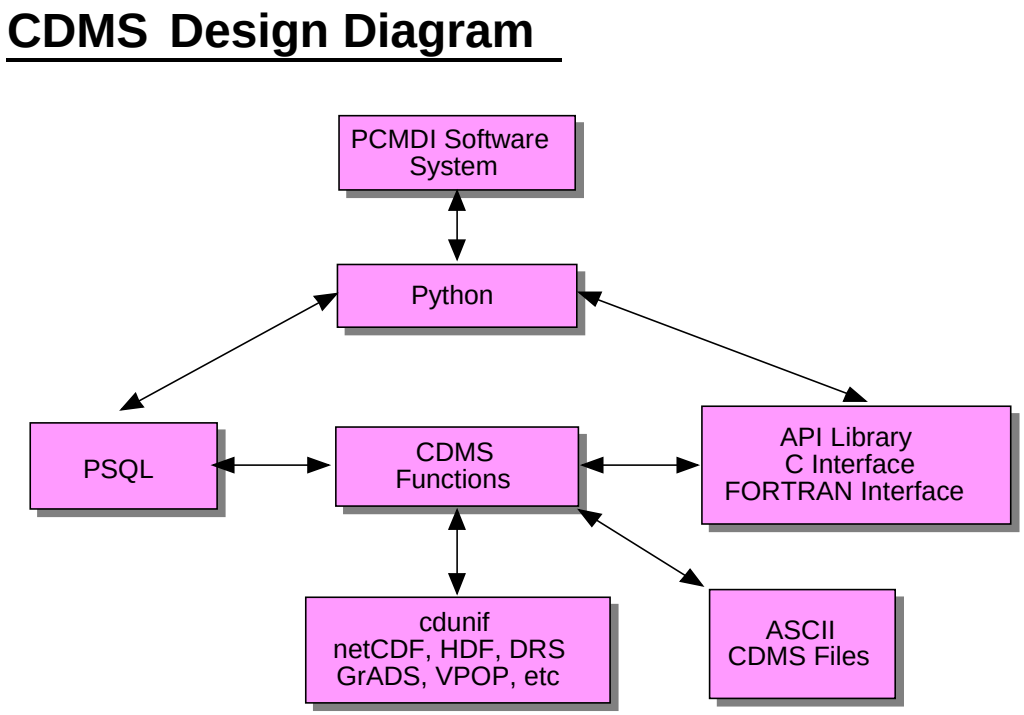

Figure 3. Conceptual view of CDMS's modular design.

CDMS Design Diagram Modules

Module	Definition/Function
PCMDI Software System	PCMDI's complete software system, consisting mainly of CDAT, PCMDI's data manipulator; CDMS, PCMDI's database; and VCS PCMDI's visualization system.
Python	An interpreted, object-oriented scripting language that allows the user to call CDAT scripts and use CDAT API commands.
CDMS Functions	Low-level routines that create and queries the database.
API Library Interface	C and FORTRAN libraries that interface into the low-level CDMS Functions.
ASCII CDMS Files	ASCII CDMS database files that user's are allowed to create and manipulate.
cdunif	Allows CDAT access to many types of data file formats (i.e., netCDF, HDF, DRS, GrADS, and GRIB).
PSQL	Command line interface that allows the user to run database commands to create, modify and query the database.

CDMS Current Status

Module	Status
PCMDI Software System	Under development, completing and integrating CDAT, CDMS, and VCS together.
Python	Connecting CDMS to Python needs to be developed.
CDMS Functions	Under development, 60 percent complete.
API Library Interface	C Interface to low-level CDMS functions are 60 percent complete. FORTRAN Interface needs to be designed and developed.
ASCII CDMS Meta Descriptive Data Files	Completed, may need modifications.
cdunif	Connections to CDMS Completed.
PSQL	PSQL is completed, may need additional functions added to the database.
Documentation	Incomplete, documentation is 10 percent complete.

2.2.3 Climate Data UNIform File (cdunif) Reader

Cdunif Problem Statement

As a participant in the international climate community, PCMDI receives and sends climate data in many different formats, requiring the ongoing transformation of others' data to fit one's own application. This often leads to data conversion errors, data duplication, and use of large

amounts of disk space. Data conversion also is expensive and time consuming; as a result, some scientific investigations in the past have been abandoned.

Figure 4 shows a hypothetical 4D data set that may have been received. Note, data can be received as a 1D, 2D, …, up to 128D data set. The figure below is just to give a perspective view of our concept of a data array.

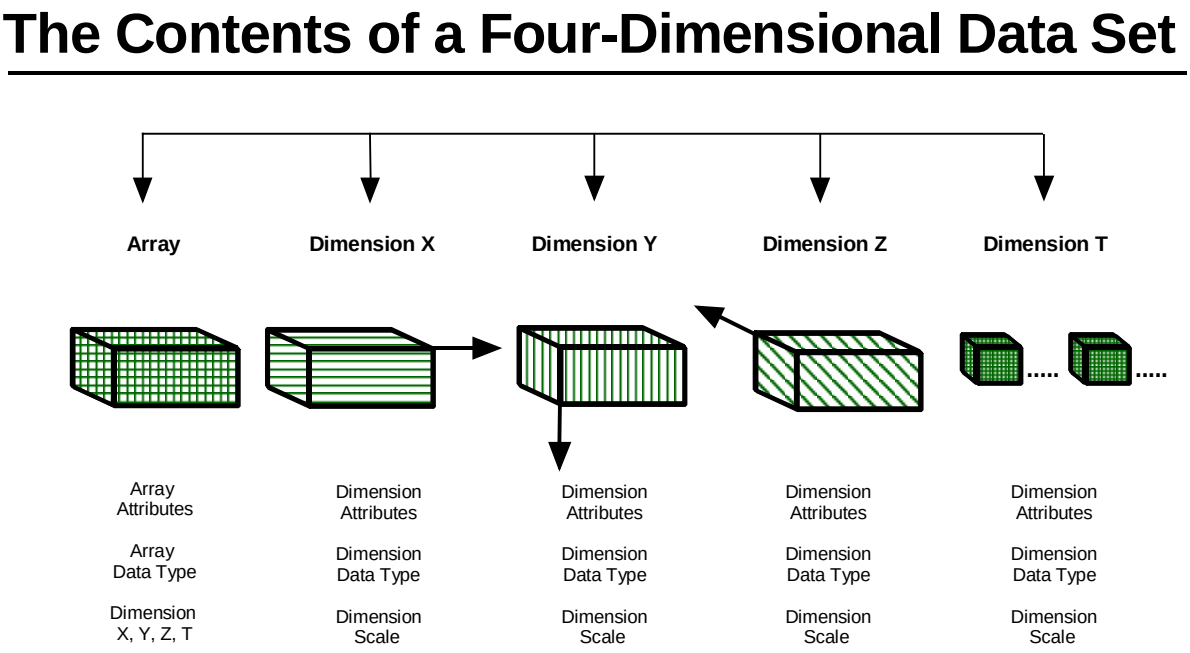

Figure 4. A conceptual look at a hypothetical array data.

Cdunif Purpose

Cdunif is an API library that allows programs to read many different structured data files without concern for their actual file format. Presently, cdunif includes a collection of uniform input/output (I/O) routines for accessing netCDF, HDF, DRS, GrADS, and VPOP file formats. Through the GrADS control file, GRIB files also can be read. Cdunif will be designed to accommodate other structured data file formats, as needed by the climate community.

All major PCMDI software products will use cdunif to ingest data directly or indirectly via CDMS.

Cdunif Requirement

The cdunif interface is modeled very closely after the UCAR netCDF interface. In most cases, cdunif functions are identical in everything except name to analogous netCDF functions. The logical model for metadata in cdunif also will be very similar to netCDF. A file contains a set of variables, dimensions, and attributes. Variables will be identified by integer ID or name;

variable IDs range from 0 to N-1, where N is the number of variables. Attributes also are identified by number or name, and can either be global or variable-specific.

Dimensions will be treated somewhat differently than in netCDF. Some of the file formats supported by cdunif, notably the Data Retrieval and Storage (DRS) format, support dimensions which are local to a given variable; they are not shared by different variables and do not have unique names relative to other dimensions. Consequently, a distinction is made between local and global dimensions, where the latter can be shared. (DRS vector dimensions are global.)

Every dimension also has an associated coordinate vector, which can be retrieved by the cudimget function. Like netCDF, dimensions are identified by name or numeric ID; both local and global dimensions have unique numeric IDs, which are sequential integers starting at 0.

The cdunif interface supports the data types corresponding to the C programming language: byte, character, short, integer, long, float, double, and long double. The cdunif functions are accessible from both C and FORTRAN programs via different API's, but contain identical function calls and parameter values. The APIs will allow the user to open and close files, to inquire about variables and dimensions and attributes, and to read data in a way that is user-specific (e.g., wrapped, transposed, reversed, etc.).

In compliance with the Cooperative Ocean/Atmosphere Research Data Service (COARDS) convention, cdunif can read data that has "missing_value", "add_offset", and "scale_factor" attributes set for the variable.

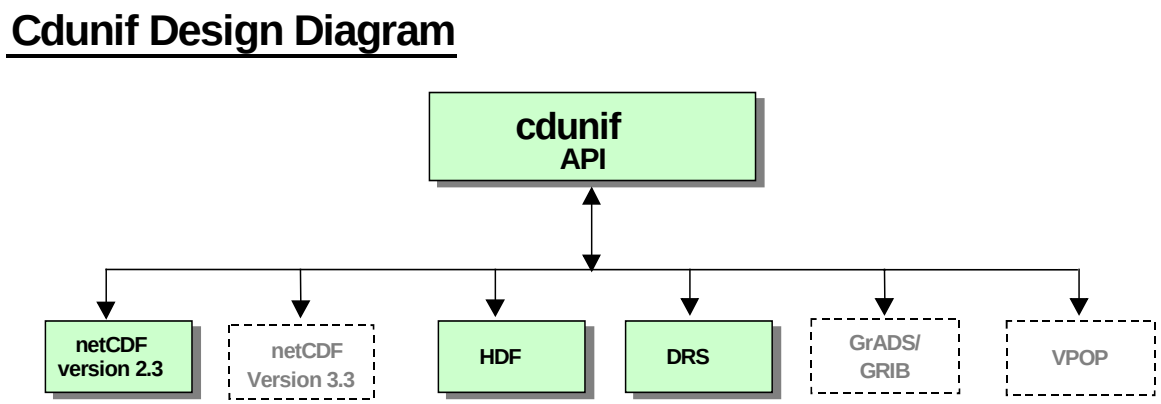

Figure 6. Conceptual view of cdunif's modular design.

Cdunif Design Diagram Modules

Module	Definition/Function
cdunif API	Provides a C and FORTRAN application programmer's interface to low-level cdunif functions (i.e., open and close a file, inquire file information, read data, etc.).
netCDF version 2.3	NetCDF library version 2.3, reads data bit-by-bit via XDR.
netCDF version 3.3	NetCDF library version 3.3, reads data differently via mmap. This may speed up the ingest process of netCDF files.
HDF	HDF library allows cdunif to ingest HDF files that were created with HDF version 3.0 or higher.
DRS	DRS library allows cdunif to ingest DRS files.
GrADS/GRIB	GrADS/GRIB is a series of C routines that read GrADS files into cdunif. Through a GrADS control file cdunif can read GRIB data.
VPOP	VPOP is a library that allows access to LANL's ocean data.

Cdunif Current Status

Module	Status
cdunif API	C interface is complete. FORTRAN interface complete, but may need modifications. cuhyperslab (allows the user to transpose, reverse, sub-set, and wrap data) - completed but may need modifications.
netCDF version 2.3	Completed.
netCDF version 3.3	Needs development.
HDF	Completed.
DRS	Completed.
GrADS/GRIB	Needs further development.
VPOP	Needs development.
Documentation	50 percent complete.

The next version of cdunif, scheduled for beta release in September 1997, will feature: HDF, VPOP, newer version of GrADS/GRIB, the cuhyperslab function, and attribute manipulation of the data (e.g., "add_offset", "missing_value", and "scale_factor").

2.2.4 Data and Dimension (DDI)

DDI Problem Statement

The Data and Dimensions Interface (DDI) addresses a significant problem in the visualization of large data sets: the need to extract only the relevant data and to readily provide these in the required form to a chosen graphics engine.

Many different research communities have developed formats for storing and retrieving scientific data and their associated metadata. These formats, while similar in their intent, are often incompatible, and their lack of support by commercial visualization systems has been an impediment to their widespread use. Accessing large data files has also proven problematical.

Sometimes generated on supercomputers, these data often are visualized on smaller platforms (i.e., workstations, PCs, etc.), with the data transfer being implemented either by copying to disk or by use of the Internet's File Transport Protocol (FTP). In both cases, the data has lacked solid quality-control.

DDI Purpose

DDI transfers data between files, formats and visualization systems.

DDI Requirements

DDI works within the specifications of each supported format to promote compatibility and provide the following capabilities:

- Browsing the contents of files written in a variety of formats;
- Randomly selecting data variables;
- Rearranging variables into the desired form;
- Modifying the values of certain variable attributes;
- Saving variables to new files;
- Feeding variables directly to a visualization system.

DDI operates in two modes:

- As a module in a data flow environment;
- As a stand-alone application capable of sending data over the network.

Presently, DDI is able to transfer variables to the following visualization systems:

- The PCMDI Visualization and Computation System (VCS);
- The Applications Visualization System (AVS), from Advanced Visual Systems Inc.;
- Collage and XImage, from the National Center for Supercomputer Applications (NCSA);
- The Interactive Data Language (IDL) from Research Systems, Inc.;
- IRIS Explorer, from Silicon Graphics Inc.;
- PV-WAVE, from Visual Numerics Inc.
- Khoros, from Khoral Research, Inc.

DDI Design Diagram

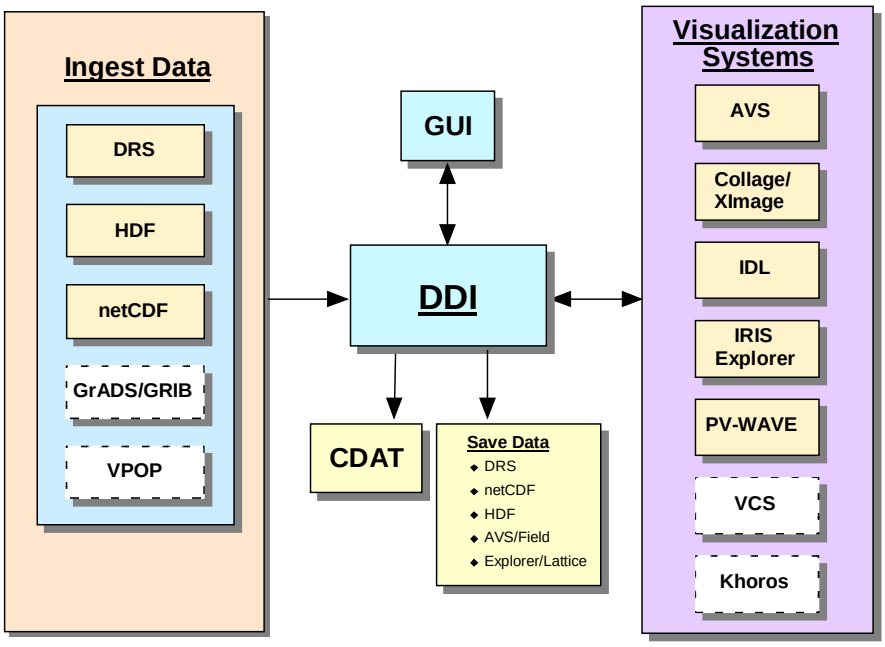

Figure 7. Conceptual view of DDI's modular design.

DDI Design Diagram Modules

Module	Definition/Function
DDI	Performs data operations, such as: transformation of the data; reversal and scaling of coordinate values; modification, creation, and removal of attributes; export of data to visualization engines.
GUI	Allows data access and access to visualization systems via a point-and-click interface.
AVS	Allows sending of data to AVS directly, where DDI is a module in the AVS data flow environment, or indirectly using UNIX sockets.
Collage/ XImage	Allows sending of data indirectly to the NCSA visualization tools, using UNIX sockets.
IDL	Allows the sending of data indirect to IDL, using UNIX sockets.
IRIS Explorer	Allows the sending of data to Explorer directly, where DDI is a module in the Explorer data flow environment, or indirectly using UNIX sockets.
Khoros	Allows the sending of data to Khoros directly, where DDI is a module in the Khoros data flow environment, or indirectly using UNIX sockets.
PV-WAVE	Allows the sending of data to PV-WAVE indirectly using UNIX sockets.
VCS	Allows the sending of data indirectly to VCS using UNIX sockets.
Save Data	Allows the saving of data in five file formats: DRS, netCDF, HDF, AVS/Field, and Explorer/Lattice.
Ingest Data	Allows the ingesting of data in five file formats: DRS, GrADS/GRIB, HDF, netCDF, and VPOP.

DDI Current Status

Module	Status
DDI	Completed.
GUI	Completed.
AVS	Completed.
College/XImage	Completed.
IDL	Completed.
IRIS Explorer	Completed.
PV-WAVE	Completed.
VCS	Needs design and development.
Khoros	Needs design and development.
Ingest Data	Incomplete, needs access to cdunif module. By accessing the cdunif module, DDI will be able to ingest GrADS/GRIB and VPOP data.

DDI version 1.2 is currently released to PCMDI users and to the public.

2.2.5 Data Retrieval and Storage (DRS)

DRS Problem Statement

Prior to the 1990's, climate scientists commonly stored data in a FORTRAN binary or ASCII file, with metadata also in ASCII. This method was problematical when porting data from a supercomputer to a workstation, but later the cross-platform issue was solved by the introduction of the IEEE standard file format. However, the exchange of scientific data remained difficult due to the difficulties of reading different file formats. This problem is alleviated by the Data Retrieval and Storage (DRS) library and utilities developed by PCMDI.

DRS Purpose

The DRS (Data Retrieval and Storage) library and utilities support the scientific data format used at PCMDI as well as the high-volume multi-dimensional array output from global climate models. Presently, DRS runs on the following platforms: Cray/UNICOS, Sun/SunOS 4.1.3, Sun/Solaris 2.4, SGI IRIX 5.3, IBM RS6000/AIX 3.2, DEC Alpha/OSF, HP/HP-UX 9.0, and Linux/PC.

DRS Requirements

Notable features of the DRS system include:

- support for multi-dimensional array variables
- sub-setting and data reordering operations
- direct access I/O, and access by coordinate ranges
- machine-independent format
- support for Cray native mode
- C and FORTRAN APIs
- interactive data browser utility

DRS Design Diagram

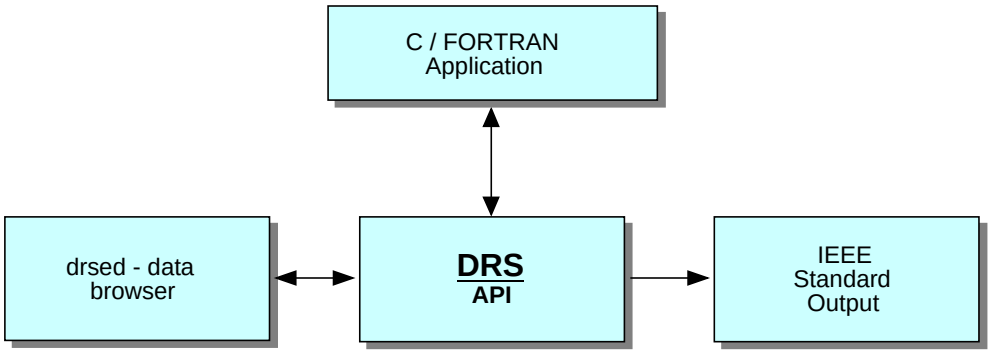

Figure 8. Conceptual view of DRS's modular design.

DRS Design Diagram Modules

Module	Definition/Function
DRS	Provides a C and FORTRAN application programmer's interface to low-level DRS functions (i.e., open and close a file, inquire file information, read data, write data, etc.).
C/FORTRAN Application	C or FORTRAN program that calls DRS functions via the API.
Drsed data browser	Utility program that browses DRS files.
DRS Output	Saves data in IEEE standard output. This allows the data to be read on different platforms.

DRS Current Status

Module	Status
DRS	Completed. May need to port to different platforms.
Drsed - data browser	Completed.

DRS version 1.6 is currently released to PCMDI users and to the public.

2.2.6 EzGet

EzGet Problem Statement

After the inception of DRS, the need for extensions of this data access software became apparent. These include the need for better error trapping capabilities and more detailed error messages; for more convenient ways to select data from specified regions (e.g., "Oceans", "North America", all land areas north of 45 degrees latitude, etc.); for the ability to map data to a new grid at the time it is retrieved; for automatic creation of "weights" used in subsequent averaging or masking of data; for automatic retrieval of all dimension information; and for increased control in specifying the domain, grid and structure of the retrieved data.

EzGet Purpose

EzGet (pronounced "Easy-Get") was developed to meet the needs noted above. It serves a different purpose from other software tools developed at PCMDI in that, unlike DDI and VCS, EzGet comprises a set of subroutines that can be linked to any FORTRAN program. The software accesses data stored in file formats that are readable by the cdunif FORTRAN API; however, use of EzGet does not require knowledge of cdunif.

EzGet Requirements

EzGet is written in FORTRAN, the programming language most familiar to climate scientists. However, to comply with the overall design of the PCMDI software system, many of EzGet's functions will be incorporated into CDAT.

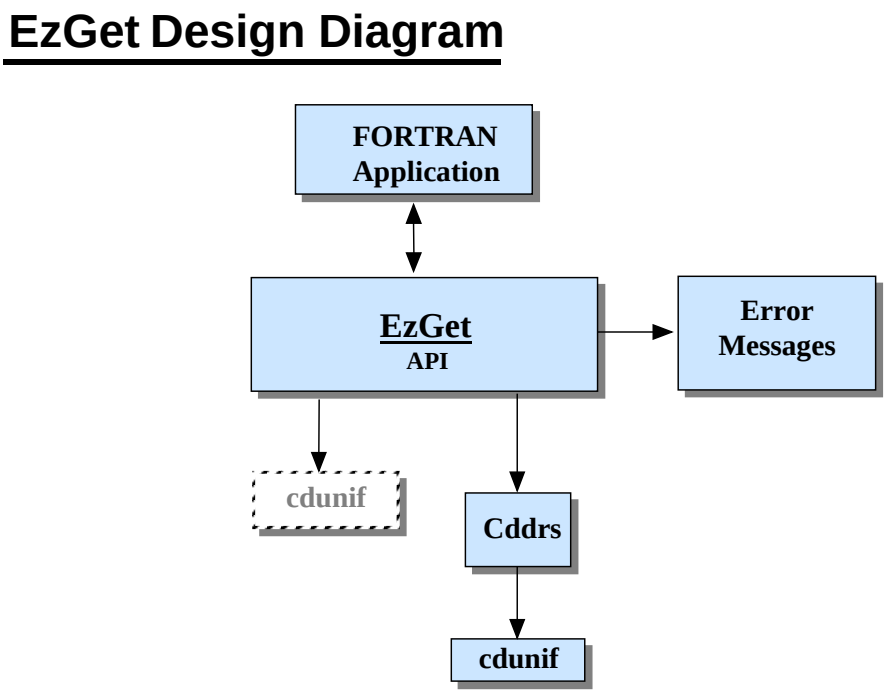

Figure 9. Conceptual view of EzGet's modular design.

EzGet Design Diagram Modules

Module	Definition/Function
EzGet	Provides the user with a FORTRAN API that has substantial error trapping capabilities, the capability of selecting data from specified regions, the ability to map data to a new grid, the creation of average weights, and the automatic retrieval of all dimension information.
Standard Output	Error Messages.
Cddrs	Indirect connection to cdunif. It uses the DRS API to feed information to cdunif.
FORTRAN Application	User specified FORTRAN program.
cdunif	Allows EzGet access to many types of data file formats (i.e., netCDF, HDF, DRS, GrADS, and GRIB).

EzGet Current Status

Module	Status
EzGet	Completed.
Standard Output	Completed.
Cddrs	Completed.
cdunif	Direct cdunif connection to EzGet is incomplete. Design and Development is necessary.

EzGet version 1.0 is currently released to PCMDI users and to the public.

2.2.7 Library of AMIP Data Transmission (LATS)

LATS Problem Statement

A library of software routines to output gridded data in the COARDS convention is needed to simplify the processing of voluminous data from the next phase of the AMIP (denoted as AMIP II) and other model intercomparison projects.

LATS Purpose

LATS is a library of software routines for output of gridded data that was developed by PCMDI in support of AMIP II and other intercomparison projects. The primary function of LATS will be to establish and implement a convention/standard for gridded data, and thus facilitate data handling and exchange. The LATS API, which is designed to be simple to understand and use, provides FORTRAN and C interfaces.

LATS applications can output data in either the WMO GRIdded Binary (GRIB) format or in the netCDF format under the COARDS convention.

LATS Requirements

LATS is not considered to be a general purpose gridded data output system; rather, it operates within the following design constraints:

- Simple to use
- Multi-platform
- Targeted for global climate model output (currently limited to use with rectilinear data)
- Implementing COARDS conventions for data and metadata
- Using two well-established and commonly used data formats
- Software to be made available in the public domain.

LATS outputs rectilinear, gridded, spatio-temporal data written in either GRID or netCDF formats that are machine-independent. The fundamental unit of data written with a single function is known as a horizontal longitude-latitude slice of a variable. However, derivative structures such as zonal mean-height cross sections (i.e., a sequence of single-longitude, multiple-latitude grids) are also supported.

LATS maintains an internal parameter table that prescribes variable names, description, units, datatype, basic structure (e.g., upper air or surface), and compression (GRIB options). These descriptors are inferred from the parameter name only. Thus, most of the metadata needed to write GRIB and/or netCDF data are located in the parameter table to simplify the API. An option also is provided to override the internal table with an external parameter file.

LATS parameter tables are designed to fit the specific needs of climate model intercomparison projects:

- More than one LATS file may be open simultaneously for output;
- For a given time point, multiple variables and variables at multiple levels may be written in any order;
- Data can only be written in increasing time sequence;
- All data are floating-point or integer. Only FORTRAN REALS (C floats) and INTEGERS (C ints) are written.

LATS Design Diagram

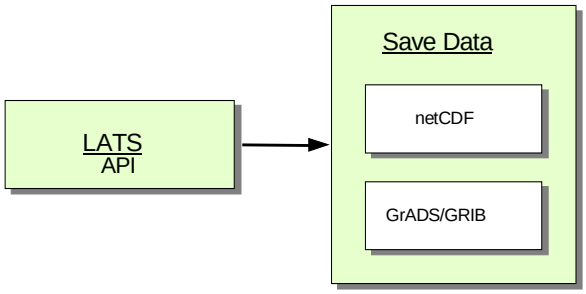

Figure 10. Conceptual view of LATS's modular design.

LATS Design Diagram Modules

Module	Definition/Function
LATS	Contains functions to save data in the COARDS conventions. It also contains a C and FORTRAN API.
Save Data	Saves data in either netCDF or GrADS/GRIB formats.

LATS Current Status

Module	Status
LATS	Completed.
Save Data	Completed, but will need work on the GrADS/GRIB API interface.

LATS version 1.1 is currently released to PCMDI users and to the public.

2.2.8 Quality Control Software (QCS)

QCS Problem Statement

In AMIP II, PCMDI anticipates the need to process and quality control large data sets from numerous climate models in a relatively short period of time. Past experience suggests that the quality of the data will vary considerably over time and from one modeling group to another. In addition, the large size and variety of the data, the short time frame for their evaluation, and the possibility of errors all demand the development of software to automate the quality control (QC) process.

QCS Purpose

QCS provides a semiautomatic procedure for quality control of these model output data. The data first are read and organized for evaluation of their consistency, especially with respect to integrity of time sequence. Then the data are rewritten in a standard time series to facilitate the

QC process, which consists of two phases. First, the data are subjected to a numerical test for statistical idiosyncrasies; then, VCS plots of specially constructed diagnostics fields are visually inspected to reveal problems not uncovered by the numerical test. As a final step, QCS archives the quality-controlled data in a standard format.

QCS Requirements

QCS is written in Python with emphasis on the use of other software currently under development, in particular the LATS and cdunif modules that are being coded for CDAT. In addition, QCS has its own math module to implement special calculations needed in the QC process.

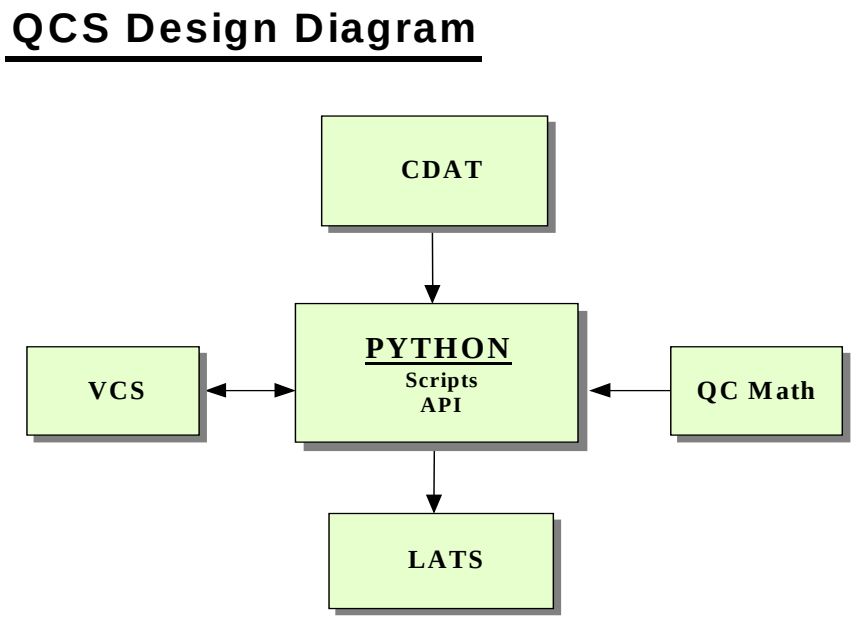

Figure 11. Conceptual view of QCS's modular design.

QCS Design Diagram Modules

Module	Definition/Function
Python	An interpreted, object-oriented scripting language that allows calls to CDAT, QCS, and VCS scripts and also allows use of API commands for CDAT, QCS, and VCS.
CDAT	Provides the diagnostic, statistical, and regridding routines.
LATS	Allows the saving of manipulated data in the PCMDI convention standards. LATS format options for saving data are either netCDF or GrADS/GRIB.
QC Math	Implements numerical testing of the data for statistical idiosyncrasies.
VCS	Allows the display and animation of scientific data.

QCS Current Status

Module	Status
QC Math	Under development. Once the QC math functions are complete, then the connections to Python must be implemented.
Database Inquiry	Completed. These database inquiries are not to be confused with CDMS.
VCS	Completed, may need modifications.
LATS	Completed, may need modifications.
CDAT	Started, but incomplete.

QCS is scheduled for beta test in September, 1997. Beta release of this product will follow that of CDAT.

2.2.9 Visualization and Computation System (VCS)

VCS Problem Statement

The massive amounts (~ gigabytes) of data produced by climate model simulations makes imperative the development of powerful visualization tools. Because no single display technique can elucidate all facets of climate model behavior, the visualization system must possess a wide range of graphics capabilities. In comparing simulations from many different models, it is also necessary to perform grid transformations and computations of additional diagnostic variables.

VCS Purpose

The Visualization and Computation System (VCS) is designed to meet the visualization and graphics needs of climate scientists. Because of the breadth of its capabilities, VCS has proven to be a useful tool for other scientific applications as well. VCS allows wide-ranging changes to be made to the data display, provides for hard-copy output and includes a means for recovery of a previous display. VCS also allows the user to browse large amounts of data and obviates the need to create and keep track of intermediate data files.

VCS Requirements

In the VCS model, the data display is defined by a trio of named attribute sets, designated the "primary elements". They include: the data, which define what to display; the graphics method, which specifies the display technique; and the picture template, which determines the appearance of each segment of the display.
In addition, detailed specification of the primary elements' attributes are provided by eight "secondary elements":

- colormap: specification of combinations of 256 available colors;
- fill area: style, style index, and color index;
- format: specifications for converting numbers to display strings;
- line: line type, width and color index;
- list: a sequence of pairs of numerical and character values;
- marker: marker type, size, and color index;
- text: text font type, character spacing, expansion and color index;
- text orientation: character height, angle, path, and horizontal/vertical alignment.

By combining primary and secondary elements in various ways (either interactively or in batch mode), the VCS user is able to comprehensively diagnose and intercompare climate model simulations.

VCS provides capabilities to:

- ingest data written in netCDF, HDF, DRS, GrADS, GRIB, or VPOP data file formats
- browse data directories and read these file formats
- quickly display variables via default settings
- view, select and modify attributes of data variables and of their dimensions
- create and modify existing template attributes and graphics methods
- save the state of the system as a script to be run interactively or in batch mode
- save a display as a Computer Graphics Metafile (CGM), raster, or Postscript file
- perform grid transformations and compute new data variables
- create and modify colormaps and zoom into a specified portion of a display or the VCS Canvas
- change the orientation (portrait vs. landscape) or size (partial vs. full-screen) of a display
- animate a single data variable or multiple data variables simultaneously
- save modified data variables in netCDF-, HDF-, or DRS-formatted files

VCS Design Diagram

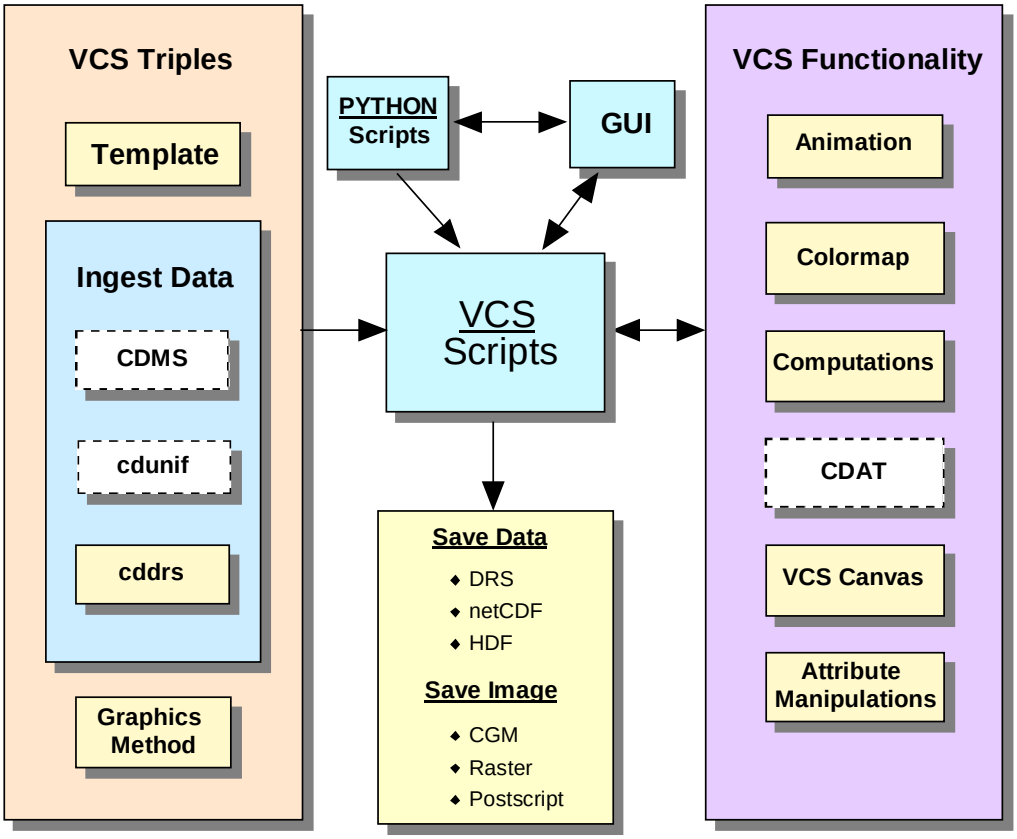

Figure 12. Conceptual view of VCS's modular design.

VCS Design Diagram Modules

Module	Definition/Function
VCS	Contains functions to: create templates, graphics methods, and modify data, interact with GUI and scripts, animate, compute, change colormaps, save data, and save images.
VCS Script	Allows the saving and running of VCS commands in batch mode or from the GUI.
VCS GUI	Allows the user to interface with VCS via a graphical user's interface.
VCS Functionality	VCS functions allow the user to animate data, change the colormaps, compute data, interface to CDAT, modify the VCS canvas, and change data attributes.
Save Data	Saves data in three different file formats: DRS, netCDF, and HDF.
Save Image	Saves images in three different formats: CGM, raster, and Postscript.
VCS Triples	Defines by a trio of named attributes: data, template, and graphics method.

VCS Current Status

Module	Status
VCS	Completed.
VCS Scripts	Completed, may need modifications.
VCS GUI	Completed, will need modifications.
VCS Functionality	Completed, may need modifications.
Save Data	Completed, may need modifications.
Save Image	Completed, may need modifications.
VCS Triples	Completed.

VCS version 2.7 is currently released to PCMDI users and to the public.

3. Web Development and Software Documentation

Documentation is an important means to inform users of PCMDI's software products. Documentation also explains the problem the software addresses, describes the purpose and requirements of the software, and shows how to use the software by providing relevant examples.

All of PCMDI's software products described in this report will be documented to high standards of quality and will be accessible via the World Wide Web. Currently, documentation for five of PCMDI's software products (DDI, DRS, EzGET, LATS, and VCS) may be viewed at Web address **http://www-pcmdi.llnl.gov/software/**

Also, on our Web site, PCMDI software users can submit bug and enhancement requests via the "PCMDI Software Bugs Form" and the "PCMDI Software Enhancements Request Form", respectively.

Software documentation will be provided in several formats, including compressed HTML, Adobe Acrobat PDF, and compressed book-form Postscript with pagination, table of contents, and index.

4. Multiple Platforms

All software products listed in the PCMDI Software System are able to run on the following platforms:

Platforms	Operating Systems
Cray C90 and J90	UNICOS 8.0
DEC(Digital Equipment Corporation)	OSF 3.2
HP (Hewlett Packard)	HP-UX 9.0.5 or higher
IBM(International Business Machines)	AIX 3.5 or higher
PC (Personal Computer)	Linux 3.0.3 or higher
SGI (Silicon Graphics, Inc.)	IRIX 5.2, 5.3, or 6.1
SUN Microsystems, Inc.	SunOS 4.1.3 or Solaris 2.4 or higher

Note: PCMDI will work towards porting our software products to Windows NT. This will be done either by rewriting the code to work in the Windows NT environment, or by porting the UNIX/X11R6/Motif/C/FORTRAN code using NutCracker NT or OpenNT.

5. Software Collaborations

In July of 1995, PCMDI software team members met with Dave Williamson and Jim Hack of the National Center for Atmospheric Research (NCAR), at which time the Visualization and Computation System (VCS) was demonstrated. As it was clear that PCMDI and NCAR would greatly benefit from collaboration, Dean Williams subsequently drafted a PCMDI-NCAR collaboration agreement. PCMDI's part of the agreement was to provide NCAR with VCS, and to design and develop a calculator for climate analysis (now designated as CDAT). NCAR's part of the agreement was to provide PCMDI with climate diagnostic routines.

In May of 1997, PCMDI software team members met with Dr. Bob Malone, the director of the Advanced Computing Laboratory (ACL) at Los Alamos National Laboratory. After demonstration and discussion VCS and CDAT, PCMDI and ACL also entered into a software collaboration agreement. PCMDI will provide ACL with software products (i.e., PCMDI's Software System) and ACL will provide PCMDI with oceanic numeric functions.

6. Code Repository

Since PCMDI has a large amount of developed code, it is imperative that some type of code history be implemented. To maintain current and future code, the PCMDI staff has elected to use the Concurrent Version System (CVS). CVS will provide network-transparent source code control for PCMDI's software system, and will assist the software development team by providing the following features.

- CVS will maintain a history of all PCMDI software changes made to each directory tree. Using this history, CVS can recreate past states of the tree, or show a developer when, why, and by whom a given change was made. By supporting branches, CVS will help manage long-term changes and bug-fix releases.

- CVS will provide reliable access to PCMDI's software directory trees from remote hosts, using internet protocols. The PCMDI software team member, working either from home or at a remote site, can perform all the same operations as are available locally. Access can be authenticated using the Kerberos network security system.

- Most importantly, CVS supports parallel development, allowing more than one developer to work on the same sources at the same time.

Cyclic Software provides commercial support and development expertise for CVS. They have been contacted, and approval to purchase and setup CVS for our entire software system is underway. Dean Williams, Bob Drach, and Susan Marlais will be in charge of setting up this system.

Each person on the PCMDI software team will be expected to follow the following CVS protocols when developing or maintaining the PCMDI Software System. (The version control is described below by Cyclic Software.)

- Each developer uses the command **cvs checkout** to create their own copy of the source tree from the CVS repository. The command can operate on a directory tree, on a single file, or on a module; a module groups several files or directories into one entity, which can be operated on as a unit. One defines modules by editing the 'modules' file.

- The developer modifies, compiles, and tests the code in their copy of the source tree (called a working directory) with whatever editors and tools they choose -- **Emacs**, **make**, **etags**, etc. Users will use commands **cvs add** and **cvs remove** to add and remove files.

- When the changes are complete, the developer uses the **command cvs commit** to merge changes back into the repository. This makes her changes available to other developers.

- At any point, the developer may use the command **cvs update** to merge changes committed by others into her working directory. If there are uncommitted changes to files in their working directory, CVS prints a message and attempts to merge the changes from the repository with their changes in the working directory. If the merge fails, CVS indicates a conflict, which the developer resolves manually with a text editor.

- The developer can show the differences between two revisions with the command **cvs diff**; show the log of changes to a particular file **with cvs log**; show the history of each line of a file with **cvs annotate**; and show who has used **cvs checkout**, **cvs tag**, and several other CVS commands, by using the **cvs history** command.

- CVS supports **watches**, allowing developers to request notification when someone else begins editing a file, or to obtain a list of developers currently working on a file.

- The user can record the state of the repository at a particular point with the **cvs tag** command, and can then use that tag as an argument to most CVS commands, for example to retrieve the files as of the tagged point.

- The developer can create a new development branch with the command **cvs tag -b**, and manipulate branches with **cvs update -r and cvs checkout -r**. Subsequent operations in that working directory apply to that branch. To return to the main branch, the developer can use **cvs update -A**. The **cvs update -j** command merges changes made on another branch into the working directory.

- The existence or nonexistence of a file is itself version controlled, so that files can exist on some, but not all, branches and users can reproduce the state of the files at any given point in time.

- The developer can mark a file as binary, which prohibits merging and line terminator conversions, using the command **cvs admin -kb**.

- **Wrappers** allow the developer to run files through a filter on their way in or out of CVS; for example, developers can instruct CVS to apply the **indent** command to source files before committing the changes.

The CVS client and server run on most UNIX platforms. (PCMDI will be developing and maintaining code on SUN and SGI machines.) The CVS client also runs on Windows NT, a desirable feature since we may wish to port the PCMDI Software System over to Windows NT in the future.

7. Summary of the PCMDI Software System

The principal mission of the Program for Climate Model Diagnosis and Intercomparison (PCMDI) is to develop improved and seamlessly interconnected methods for the diagnosis, validation and intercomparison of global climate models. The need for such standards has become increasingly apparent as more complex climate models are developed, while the disagreements among models, and between models and observations, remains significant and poorly documented. The nature and causes of these disagreements must be better understood before these models can be confidently used for climate sensitivity and predictability studies in support of global change research.

The PCMDI Software System was designed to aid climate scientists in the study of global climate models. Its concept is simple and flexible enough to interchange its parts and expand in the future. The primary software system comprises of three parts: CDAT manipulates data and provide climate scientists with diagnostic, statistical, and regridding routines; CDMS automatically locates and extracts metadata (i.e., variables, dimensions, grids, etc.) from PCMDI's collection of model runs and analysis files; and VCS, a graphics software package to display, animate, and manipulate scientific data.

Each one of these software products is independent and can run as a standalone process or can run together as part of a single process. The three systems share common scripting commands implemented by Python, an interpreted, object-oriented language, and are available for use in C or FORTRAN applications via an API. In the figure below, the central box "Script" represents Python, the mechanism that connects all PCMDI software.

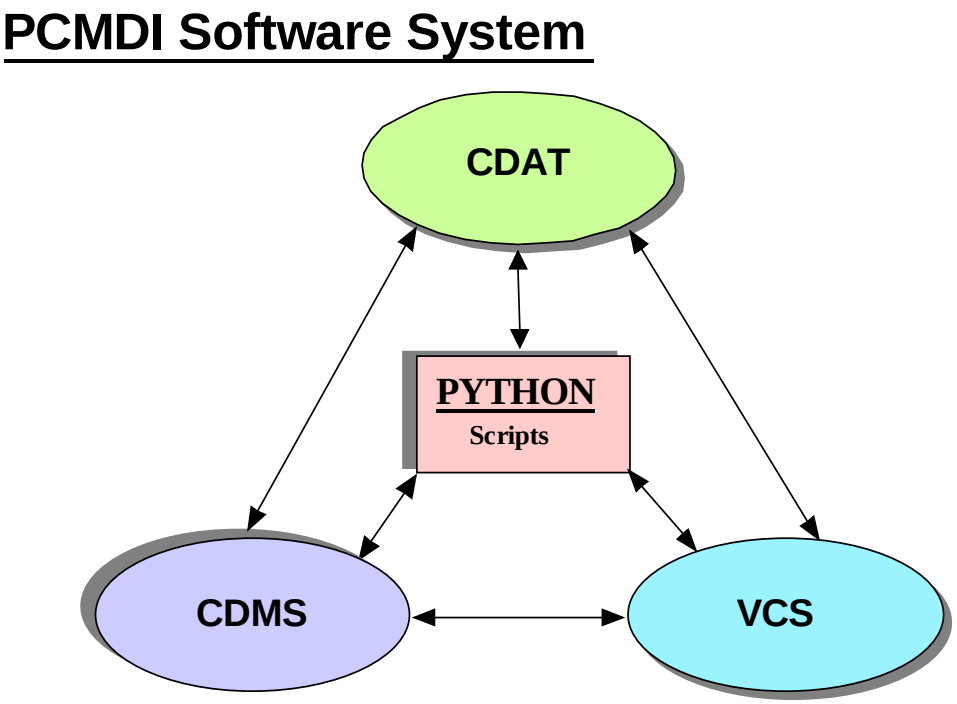

Figure 13. Conceptual view of PCMDI's Software System.

When running CDAT as a standalone process, a command line user interface will appear along with a CDAT output window. The "CDAT>" prompt will wait for the user to input a command CDAT command. Through this interface, data are manipulated, and the other two software systems can be accessed via a Python script command. Therefore, a likely scenario in a CDAT session would be to search the directory for data, manipulate the data, and then plot the data via VCS. To help users move quickly through the data, some of the VCS GUI widgets are included in (e.g., "Animation Panel", "Colormap Editor", etc.). CDAT also contains an on-line help menu for quick command lookup.

CDMS is similar to CDAT in its command line user interface, the only exception being the prompt "CDMS>" instead of "CDAT>". CDMS also can be initiated to run in an Xterm window. When operating in this mode, the user has no access to CDAT or CDMS, but still can run scripts that are designated to be done by CDMS. In that event, the prompt at the Xterm window then changes to "PSQL>" which stands for PCMDI Structured Query Language.

VCS has the ability to run in batch mode or from an elaborate graphical user interface (GUI). Because VCS was developed first, and in order to accommodate the many current VCS users, CDAT and CDMS are designed to be accessed from within the VCS GUI. Thus, the user will be able to choose whether to access data files and variables via the currently implemented directory, or though CDMS, using PSQL calls. Here, data manipulation will be done with CDAT via a point-and-click environment or through the command line user interface.
Taken together, PCMDI software tools are designed to provide climate scientists with the best possible means to analyze their data. These software products allow manipulation of data to

perform needed calculations, display of results, annotation of various data file formats and access of multi-dimensional arrays.

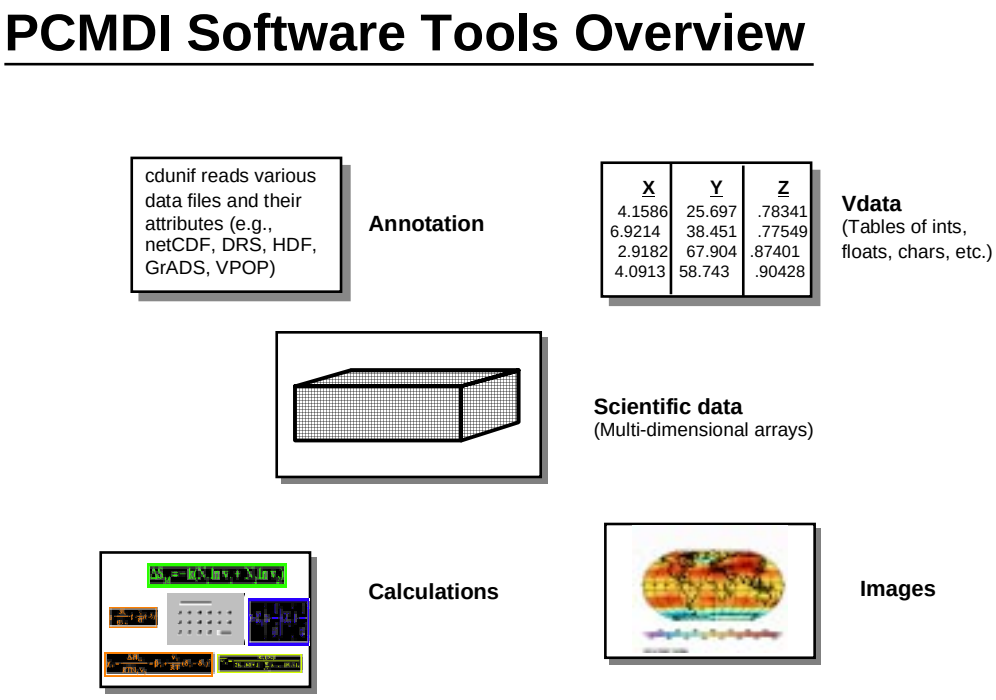

Figure 14. Conceptual view of PCMDI's Software System.

For further information on PCMDI and the PCMDI Software System visit our Web site at URL **http:/www-pcmdi.llnl.gov/**