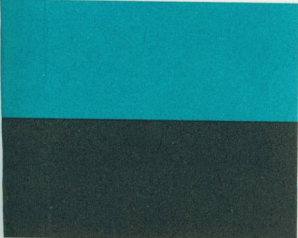


Keywords:
Air quality
Air quality management
Air quality models
CAMRAQ
Computer graphics
Computer simulation
Environmental models
Meteorological models
Atmosphere

EPRI TR-106852
WO4311-02
Final Report
September 1996



Design of a Framework for the Development of a Comprehensive Modeling System for Air Pollution

Published by
Electric Power Research Institute
Palo Alto, California

Design of a Framework for the Development of a Comprehensive Modeling System for Air Pollution

Distributed computer systems hold the potential to revolutionize the current approach to environmental modeling. This report presents the design of the framework for developing a comprehensive modeling system (CMS) for air pollution. This system would provide a platform for modeling pollutant emissions, atmospheric physics and chemistry, and the impact of pollution. The system would also provide a readily accessible interface, a powerful set of analysis and decision support tools, and a method to make maximum use of available computational resources.

INTEREST CATEGORY

Air quality

KEYWORDS

Air quality
Air quality management
Air quality models
CAMRAQ
Computer graphics
Computer simulation
Environmental models
Meteorological models
Atmosphere

BACKGROUND Air quality models currently address a variety of environmental issues such as emission permitting, ozone control, acid deposition, particulate matter, air toxics, emergency planning and response, human exposure, and risk assessment. Each issue is treated with different sets of modeling tools of varying levels of scientific detail and complexity. The benefits of the CMS would be multi-fold: the latest scientific models would be available, the user-friendly system interface would greatly increase the number of potential users and their access to more sophisticated modeling tools, policy decisions would be based on the latest science, and the costs of model applications would be greatly reduced. This project, sponsored by the Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ), presents the framework for CMS development.

OBJECTIVE

- To create a tool for system-assisted, user-friendly use of advanced air quality models in order to facilitate application of the best science in regulatory decisions.
 - To provide a software platform for the application and incorporation of the most advanced air pollution models.
-

APPROACH In designing a framework for the CMS, the project team took into account users' needs, both those explicitly mentioned in a user survey conducted at the beginning of this project and ones dictated by experience and expectations. The ultimate CMS would be developed through a series of increasingly comprehensive prototypes. Particular attention would be paid to the mechanisms the framework provides for rapid adaptation to changes in user requirements and methods, transitions in regulatory mandates, and the explosive growth in computational technologies.

RESULTS This report provides design, implementation, and management plans for the CMS. By design, users would access the CMS system through a standard graphical user interface (GUI), which would include visualization software for displaying two- and three-dimensional representations of input and output data. The interface would also include geographic information system (GIS) capabilities for accessing and displaying data according to geographical location. Powerful statistical packages would be available for summarizing, synthesizing, and analyzing data as well as extensive on-line help files to minimize or avoid reliance on operator's

manuals. The fully developed CMS would be accessible from the user's workplace through a Unix workstation or PC-based computer interface.

The GUI would allow the user to point and click with a mouse to select a model configuration (emissions, meteorological, dispersion, or air quality) consistent with the intended application. If desired, the user could also select individual modules (preprocessors, chemical mechanism, boundary layer treatment, advection scheme) other than the default set. Having configured the model, the user could then select the geographic domain and time period. Finally, the user could choose the corresponding emissions, meteorology, air quality, topography, land use, and demographic data, often from a geographically distributed archive via the Internet.

EPRI PERSPECTIVE The existing infrastructure and technology for developing, evaluating, and applying regional air quality models present a multitude of difficulties and obstacles to their convenient use. The realism of their simulations is generally ill-defined. Their inner workings are opaque. The breadth of the issues they address is narrow. These and related problems could be significantly alleviated if a new paradigm for air quality modeling were established: Use a comprehensive modeling system that explicitly takes into account existing impediments to reliable and efficient modeling. The CMS envisioned would provide emissions modeling, meteorological modeling, air quality modeling and characterization, decision support, and report preparation assistance. With software tools and hardware products that would place it at the cutting edge of computer and atmospheric sciences, the CMS would become the regulatory and research platform of choice for air quality modeling.

PROJECT

WO4311-02

EPRI Project Manager: D. A. Hansen

Environment Group

Contractor: Electric Power Research Institute

**For ordering information about this report, call the
EPRI Distribution Center (510) 934-4212.**

For membership information, call (415) 855-2514.

Design of a Framework for the Development of a Comprehensive Modeling System for Air Pollution

TR-106852
WO4311-02

Prepared by

P. Zannetti, FAILURE ANALYSIS ASSOCIATES, INC.

B. Bruegge, A. G. Russell, CARNEGIE MELLON UNIVERSITY

D. A. Hansen, ELECTRIC POWER RESEARCH INSTITUTE

N. Lincoln, D. A. Moon, SESCO

W. A. Lyons, MISSION RESEARCH CORPORATION

R. E. Morris, ENVIRON INTERNATIONAL CORPORATION

Funded by

American Petroleum Institute
Washington, DC

Atmospheric Environment Service, Environment Canada
Ontario, Canada

Electric Power Research Institute
Palo Alto, California

Ontario Ministry of the Environment & Energy
Ontario, Canada

Pacific Gas & Electric Company
San Ramon, California

Southern California Edison
Rosemead, California

U.S. Department of Energy
Germantown, Maryland

Prepared for
Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ)

EPRI Project Manager
D. A. Hansen

Air Quality, Health and Risk Assessment Target
Environment Group

DISCLAIMER OF WARRANTIES AND LIMITATION OF LIABILITIES

THIS REPORT WAS PREPARED BY THE ORGANIZATION(S) NAMED BELOW AS AN ACCOUNT OF WORK SPONSORED OR COSPONSORED BY THE ELECTRIC POWER RESEARCH INSTITUTE, INC. (EPRI). NEITHER EPRI, ANY MEMBER OF EPRI, ANY COSPONSOR, THE ORGANIZATION(S) NAMED BELOW, NOR ANY PERSON ACTING ON BEHALF OF ANY OF THEM:

(A) MAKES ANY WARRANTY OR REPRESENTATION WHATSOEVER, EXPRESS OR IMPLIED, (I) WITH RESPECT TO THE USE OF ANY INFORMATION, APPARATUS, METHOD, PROCESS, OR SIMILAR ITEM DISCLOSED IN THIS REPORT, INCLUDING MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, OR (II) THAT SUCH USE DOES NOT INFRINGE ON OR INTERFERE WITH PRIVATELY OWNED RIGHTS, INCLUDING ANY PARTY'S INTELLECTUAL PROPERTY, OR (III) THAT THIS REPORT IS SUITABLE TO ANY PARTICULAR USER'S CIRCUMSTANCE; OR

(B) ASSUMES RESPONSIBILITY FOR ANY DAMAGES OR OTHER LIABILITY WHATSOEVER (INCLUDING ANY CONSEQUENTIAL DAMAGES, EVEN IF EPRI OR ANY EPRI REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES) RESULTING FROM YOUR SELECTION OR USE OF THIS REPORT OR ANY INFORMATION, APPARATUS, METHOD, PROCESS, OR SIMILAR ITEM DISCLOSED IN THIS REPORT.

ORGANIZATION(S) THAT PREPARED THIS REPORT:

FAILURE ANALYSIS ASSOCIATES, INC.
149 Commonwealth Drive
Menlo Park, California 94025

CARNEGIE MELLON UNIVERSITY
Pittsburgh, Pennsylvania 15213

ELECTRIC POWER RESEARCH INSTITUTE
3412 Hillview Avenue
Palo Alto, California 94304

SSESCO
511 Eleventh Avenue South
Box 212
Minneapolis, Minnesota 55415-1536

MISSION RESEARCH CORPORATION
ASTER Division
P.O. Box 466
Ft. Collins, Colorado 80522

ENVIRON INTERNATIONAL CORPORATION
Golden Gate Plaza
101 Rowland Way
Novato, California 94945-5010

ORDERING INFORMATION

Requests for copies of this report should be directed to the EPRI Distribution Center, 207 Coggins Drive, P.O. Box 23205, Pleasant Hill, CA 94523, (510) 934-4212.

Electric Power Research Institute and EPRI are registered service marks of the Electric Power Research Institute, Inc. EPRI. POWERING PROGRESS is a service mark of the Electric Power Research Institute, Inc. Copyright © 1996 Electric Power Research Institute, Inc. All rights reserved.

ABSTRACT

This report presents a design of a framework for a Comprehensive Modeling System (CMS) for air pollution – a project sponsored by the Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ).

In our design, the CMS user sees the system through a series of interactions with a standard graphical user interface which promotes the effective use of the system by even uninitiated users. The fully-developed CMS would be accessible from the user's workplace through a computer interface, either workstation (Unix) or PC-based.

In our implementation plan, the CMS will be developed as a series of increasingly comprehensive prototypes and products. Each version will present new air quality modeling functionality to the user.

In our management plan, we define a clear management structure and precise allocation of responsibilities. We also propose a certain degree of planned redundancy and the use of task monitoring teams, to assure a cost-effective performance.

The design includes a series of six development efforts during the period 1995-1999, followed by a final effort for periodic maintenance and upgrade, from 2000 on.

PREFACE

MESSAGE TO THE READER

The Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ) is sponsoring an initial effort toward the development of a Comprehensive Modeling System (CMS) for air pollution. This framework design report was prepared as the final product of this initial effort.¹

A previous concept paper report (Zannetti et al., 1995; hereafter referred to as CP) was prepared as part of this project. The CP was a pilot document to set the stage for the full design of the CMS. In this preface, we will present a summary of concepts and information that have been already presented in the CP. Therefore, the reader already familiar with this project may choose to move directly to Chapter 1. We enclose the Table of Contents of the CP as Appendix A. A copy of the CP report can be obtained by contacting the first author at the address provided on the cover page.

WHAT IS A CMS?

The CMS must provide an infrastructure that helps its users to do their jobs better and faster, whether those jobs be regulatory and policy analysis, source impact assessment, understanding atmospheric chemistry and physics, or performing atmospheric research studies. As such, the CMS has been designed to provide the following:

¹ CAMRAQ has been created to provide an organizational structure for the development of a CMS. CAMRAQ's basic intent is to coordinate and pool individual model-development efforts of numerous organizations in a manner that leads to maximized progress, mutual benefits, and minimum duplication. CAMRAQ is dedicated to the development and application of regional air-quality models for practical planning and policy analysis. It is composed of cooperating representatives from government and industry and has this primary objective:

"To develop, evaluate the performance of, and apply comprehensive modeling systems (CMS's) for the analysis of air-quality issues on regional and smaller scales."

Additional information on CAMRAQ's approach and goals can be found in Hanna (1992), Hansen (1992), and Hansen et al. (1994).

1. A platform – for modeling pollutant emissions, atmospheric physics and chemistry, and the impact of pollution – in as scientifically sound a fashion as is desired or possible.
2. A readily accessible interface, so that its use is a benefit, not a distraction.
3. A powerful set of analysis and decision-support tools, be they graphical, visual, economic, or scientific, including report preparation.
4. A method to make maximum use of the available computational resources, including CPU power, disk storage, and communication systems.

The CMS has been designed in a way that facilitates its continuous evolution with science, computer capabilities, and user needs.

WHAT SHOULD A CMS DO?

Palo Alto, 7 April 1999. The user sits in front of an Apple-IBM *Penta III* computer screen. The screen shows a stylish *CMS* logo and several other buttons. By issuing a voice command to the system, or clicking the button *Beginners click here*, a series of windows are displayed. These windows contain detailed information sections and an animated user's guide to describe the entire system. By clicking the button *Education* a new series of "windows" and "chapters" are available. These sections are connected to CD's, laser disks and multimedia devices and provide, on the *Penta III* screen, interactive education tools on the subjects of atmospheric sciences, air pollution, laws/regulations, simulation modeling, and databases. A special *Communication* button allows the user to communicate, via user-friendly interfaces, with library databases, meteorological/air quality databases, and other users.

By clicking the *CMS Regulatory* button, the user accesses a subset of the *CMS* system in which only models and techniques that have received some regulatory approval are available. The use of these models is "locked," in the sense that they can only be used with computational options that are acceptable to the regulatory agencies. Regulations of different countries (USA, Canada, ECC, Japan, etc.) can be selected, therefore locking the execution of the simulations under different regulatory constraints. By clicking the *CMS full set* button, the user accesses the entire simulation system. Through a password and a voice recognition check, a user-developer is allowed to access the master version of *CMS* (in remote computer storage) and modify/add/update modules and functions.

A typical CMS session consists of a CMS-guided computer simulation and "report" preparation. The user defines the computational domain, the simulation period and other user-specified options. CMS assists the user in performing a sequence of simulations and choices to calculate emission data, meteorological fields, transport and diffusion scenarios, chemical reactions, dry and wet deposition, and some adverse effects of air pollution, such as visibility impairment. Any step can be fully visualized by superimposing input/output data on geographical information using a Geographical Information system (GIS) and full three-dimensional (3D) views (in a fly-through fashion). A special *Real-time* button allows real-time simulation for emergency response of accidental releases, if proper connections are made to access meteorological and other data on-line. At any time the user can select input/output data and ask CMS to perform special calculations and analyses in different computational environments (such as new versions of *Mathcad*, *Mathematica*, *Spyglass*, *Systat*, etc.) on the *Penta III* screen.

What is described above illustrates the first and, by far, the most important goal of the CMS – to let the computer do the work. In the crudest sense, this means more complex and more sophisticated computations, but this is only the tip of the iceberg. Far more important is to use the computer to compile, cross-reference, and comprehend intentions. It means using the computer/CMS to eliminate the tedium of job decks (or scripts) with their rigidly-defined sequences and syntax, to rid ourselves once and for all of the tyranny of incomprehensible file names, directory paths, file structures, and the other heirlooms of the age when computer time was more valuable than people's time and computers and computer models were run by a specially trained crew of acolytes. The goal of the CMS is as much a manifesto as a plan; it is meant to be revolutionary, not incremental.

Distributed computer systems offer the potential to revolutionize the current approach to environmental modeling. The proposed enabling technologies include high-performance heterogeneous computing systems, advanced software environments, parallel architectures, and networks with data exchanges on the order of gigabytes per second (GB/s). These developments offer the possibility not only of increasing the resolution of current models but also of providing the power needed to perform systematic sensitivity and uncertainty analyses, assimilate observational data, and incorporate more realistic treatment of the underlying physical and chemical processes. Research needs to be carried out to improve the current models, provide the mapping onto advanced architectures, and create a flexible software environment that will enable widespread use of the tools.

In summary, the CMS must include the best science and be *scientifically credible*; facilitate control strategy planning; be readily extensible; be easily used; take advantage of evolving computational environments; be robust; and be able to interact with and benefit from other modeling systems, such as EPA's Models-3 (Novak et al., 1994; Dennis et al., 1995).

INTENDED USERS OF THE CMS AND USER REQUIREMENTS

The CMS has been designed to be used both by regulatory users and researchers. By regulatory users, we mean the regulated entities, the environmental consulting firms they employ, and the regulating agencies, both at a local and national level. By researchers, we mean those who work to develop new modeling and analysis techniques and apply the latest techniques to gain new scientific insights.

Regulatory Users

For regulatory users, it is intended that the CMS will become their “standard” software environment for air quality modeling. Our initial thrust will be toward those regulatory situations that can most benefit from the use of sophisticated 3D models. This class of models has been under-used in the past, because of real or perceived onerous requirements for either data sources, computing hardware, or user expertise. We see three main areas that could be impacted by greater use of 3D models: 1) photochemical smog calculations in support of State Implementation Plan (SIP); 2) the analysis of the impact of accidental or planned release of hazardous materials; and, 3) site permitting studies.

For photochemical smog simulations, the use of 3D models has become an accepted, established fact. Also, there appears to be a lot of interest in going beyond the Gaussian plume approach for emergency preparedness, as the overly-conservative estimates (obtained when using highly simplified models) have the potential to cost industries huge amounts of money over the next few years. The other area that could be impacted is that of site permitting. The study methodologies and data requirements for these three cases are quite different, yet there are many shared requirements.

The regulatory user needs an integrated 3D graphical environment for the analysis and synthesis of all relevant spatially-referenced data, including input or set-up data and model outputs. This could include meteorological, emissions, air quality, topographical, land use, engineering, and architectural data. The ability to explore multiple datasets simultaneously in a highly interactive environment is crucial in identifying their mutual relationships and in weeding out inconsistent or implausible data. The users, for example, must be able to set a flag identifying such data as bad from within this graphical environment.

As far as photochemical modeling is concerned, regulatory modeling is making a transition from the use of simple box models, such as EKMA, to 3D advection and diffusion models with chemical reaction algorithms of varying completeness. The most frequently used model is the Urban Airshed Model (UAM-IV). The use of more state-of-the-art models outside of research applications is limited to a small number of high-profile cases.

Virtually all of the dispersion modeling related to permitting is currently done using straight-line Gaussian models, such as the Industrial Source Complex (ISC) model. Representation of chemical processes is either nonexistent or simple (first-order) decay at best.

Regulatory users would greatly benefit from the use of 3D particle and puff models (and the datasets required to drive them). The ability to simulate plume dynamics in the presence of thermal circulations (or shear, or fumigation, or spatially-variable stability, for example) is critical to many scenarios. There is also the need for better representation of chemical processes in the plumes. Also, the use of wind data interpolated from models for running the ISC model would be useful in cases of questionable representativeness of the nearest available meteorological data.

Regulators require decision support and information to develop and evaluate air quality improvement programs. This information includes emission control levels, technologies, costs, and their associated impacts. In this mode, the air quality, emission preparation, and meteorological models act as black boxes providing the link between control technology application and the expected impacts. The types of impacts that should be included in such analysis (and thus, in the CMS) include economics, air quality (including compliance demonstration), health effects, and welfare.

One attribute of the CMS will be the ability to formulate optimal control strategies for regulatory users. In this case, "optimal" will be user-specified (e.g., least cost, least cost with specific constraints, meeting a standard and minimizing exposure, etc.). One could and should imagine asking the CMS to formulate a strategy for, say, the Northeastern US subject to certain constraints and data (e.g., a least-cost strategy to meet the air quality standard using known technologies), and the CMS would specify the location and costs of the applied technologies, and the expected benefits in terms of air quality and welfare.

The ability to generate reports is particularly relevant to the design of the CMS for regulatory users. Users must be capable of easily exporting CMS-generated information into other software packages, such as word processors and spreadsheets. More importantly, the CMS will be capable of piloting the execution of other software packages (e.g., Microsoft Word and Excel) to produce quality reports.

Within the regulatory community, PCs are the dominant computer platform, with a relatively small number of practitioners working on Unix workstations. Some state agencies may have a group that deals with ozone issues that has acquired a Unix system, and of course, some consulting firms specialize in ozone issues and have therefore become Unix literate, as, in the past, only Unix systems had the operating system and processor support to tackle photochemical modeling. But the number of Unix users is very small compared to those who use PCs and work primarily with the EPA's UNAMAP set of models such as the ISC model.

Research Users

Researchers will use the CMS to make their time spent doing modeling studies and developing models more productive. By taking care of many of the repetitive aspects of a modeling project and providing standards for development of computational modules, the CMS will allow them to concentrate their attention on those aspects of the problem relevant to their research.

The primary focus of this group is research into scientific phenomena, modeling approaches, numerical methods, and analysis techniques. This often requires more complete parameterizations of physical processes like turbulent mixing or chemical reactions. This often leads to very large memory and processing speed requirements. In many cases, the need is for finer model grid spacing to resolve important flow features like sea/lake breezes and topographically induced or influenced flows. Higher-resolution simulations, over ever-larger domains, place tremendous demands on system resources.

For this group of users, access to fast computers with lots of RAM is often the overriding need. This is not to say that the benefits of a productivity-enhancing tool like the CMS are not also important. Inevitably, significant percentages of a project's time get swallowed up dealing with issues, such as model set-up and data exchange between models. It can be frustrating to have scientists with highly-specialized knowledge spending large amounts of time on these more or less mundane issues.

Research users share many of the characteristics of regulatory users, with some significant differences listed below.

- Since they are developing and not just using the codes, they need access to highly-productive development environments with fast compile and link times, sophisticated debugging tools, and code revision control systems.
- They frequently connect to remote computer servers, either high-end workstations or workstation clusters in their own group, or clusters, traditional supercomputers, or MPP machines at a central site such as an NSF computing center.
- They typically do much of their work on Unix workstations. They may go to a PC to get some tasks done (i.e., report generation) but a workstation is their "home base."

AIR QUALITY MODELING TODAY AND THE BENEFITS OF A CMS

Currently, air quality models are used to address a variety of environmental issues.

- *Permitting* of new sources or modifications of existing sources, including (in the US) new source review (NSR), prevention of significant deterioration (PSD) determination, and Title V permits.
- *Ozone control*: estimation of the effects of alternative VOC/NO_x emission control strategies on ozone concentrations and ozone attainment demonstration modeling.
- *Acid deposition* issues: estimation of the sources of acid deposition and identification of potential remedial control strategies.
- *Particulate matter*: air quality, deposition, and health effect impacts; regional-scale and local particulate matter (PM) concentration impacts at coarser (PM₁₀) and finer (PM_{2.5}) size distributions; visibility impairment.
- *Toxics*: impacts of toxic compounds (e.g., formaldehyde, benzene, 1-3-butadiene, POMs, dioxin, mercury, cadmium, etc.) on health, both through direct exposure to ambient air and through indirect sources (e.g., bioaccumulation of toxics in fish) and on ecosystems (e.g., deposition of toxics into the Great Waters).
- *Source Apportionment*, i.e., the assessment of the fractional contribution of a source (or a group of sources) to the air pollution concentrations measured in a certain region.
- *Emergency planning and response* through estimation of the range and level of impact of accidental releases (real or hypothetical) of toxic compounds.
- *Exposure* of humans to harmful air pollutants through inhalation (e.g., ozone, PM₁₀, toxics) and other pathways.
- *Risk assessment* of different mitigation measures including cost-benefits analysis of alternative emission control strategies.

Currently, each of these issues is treated with different sets of modeling tools with varying levels of scientific detail and complexity. The benefits of the CMS are multifold: the latest scientific models will be available, resulting in decisions being made using our most current scientific understanding; the user-friendly graphical user interface (GUI) will greatly increase the number of potential users and their access to more sophisticated modeling tools; policy decisions will be made using the latest science,

decreasing the risk associated with making the wrong decision; and the costs of model applications will be greatly reduced because of the ease of use of the CMS.

THE ATTRIBUTES OF THE CMS

The attributes of the CMS are grouped here into four parts.

1. Global Requirements
2. Applications of the CMS
3. Components
4. Design and Development Issues

These parts are discussed below.

Global Requirements

We have identified four overall attributes that we consider global requirements for the CMS: 1) the ability to incorporate the best science and advances in science; 2) the ability to present results to different audiences; 3) a distributed access to data; and, 4) a consistent software infrastructure for the modules to communicate with each other.

Applications of the CMS

The CMS will support the users by providing a family of applications including: 1) research applications; 2) regulatory applications; 3) decision support; 4) source apportionment; and, 5) emergency preparedness and response.

Components

We divide the components of the CMS into two sections.

- User Components
- Science Components

The user components present issues that are relevant to the different ways in which the CMS will be used. In other words, the user components cover those aspects that are of direct relevance to the CMS user. They are:

- User interface
- Visualization
- Geographical information systems (GIS)
- On-line CMS training
- Policy and regulatory aspects

- Data acquisition/reduction
- Support for report preparation

The science components are those which relate directly to the science of air pollution and atmospheric modeling topics. They are listed below.

- Emission Modeling System (EMS)
- Meteorological Modeling System (MMS)
- Air Quality Modeling System (AQMS)
 - Transport and diffusion
 - Chemistry
 - Aerosol
 - Deposition
- Statistical Methods
 - Data analysis
 - Receptor models
 - Performance evaluation
- Air Pollution Effects
 - Visibility
 - Exposure simulation
- Numerical Methods and Optimization

The CMS will include modules and routines to provide functionality to all these components. A summary of each component is presented below.

Emission Modeling System (EMS)

A key component of a CMS will be the emissions modeling system (EMS). The EMS will characterize current and future-year emission rates, evaluate the effectiveness of alternative emissions reduction scenarios and assumptions, track the progress of emissions reductions, and prepare an emissions inventory that is suitable for use in an air quality model.

In terms of CMS goals, emissions modeling should be based on the bottom-up calculation from its inception (AP42-type modeling) to the gridded model-ready emission inputs files. When new emission factors or activity levels become available, they can easily be included in the emissions modeling system.

When bottom-up emissions information is unavailable (e.g., for consumer products, paints, etc.), more representative and accurate surrogate distributions, based on GIS, should be used.

The CMS will use all the information obtained in the emissions reconciliation studies and use it to improve emissions inventory development techniques, possibly through the use of artificial intelligence. Both regulatory and scientific (i.e., best estimates) inventories will be available through the CMS.

Emissions estimation for modeling accidental releases is a completely different issue from the development of emission inputs for 3D photochemical or acid deposition modeling. An accurate real-time estimate of the emissions of a toxic pollutant is needed in emergency response release modeling. These estimates will be obtained by using techniques such as those presented by Hanna and Drivas (1987) and the US EPA (1989).

Algorithms for the estimation of emergency releases will be available in the CMS for quick use by the facility operators to estimate the amounts of pollutant discharges.

Meteorological Modeling System (MMS)

The meteorological modeling system (MMS) will be a key component of the CMS. The MMS will simulate all the required meteorological variables at any scale required by the CMS user. The MMS will include both diagnostic and prognostic modules. Diagnostic models are based on objective analysis of available meteorological measurements in the study region and simply provide a best estimate of steady-state meteorological fields by appropriate data interpolation and extrapolation techniques. Prognostic models simulate the evolution in time of the atmospheric system through the space-time integration of conservation equations.

Air Quality Modeling System (AQMS)

Another key aspect of a CMS will be the provision of an air quality modeling system (AQMS), which consists of four components: transport and diffusion, chemistry, aerosol, and deposition.

Statistical Methods and Performance Evaluation

The CMS will contain a set of modules to perform statistical data analysis of all available data, including those generated by the simulations. In addition to traditional statistical data analysis routines, such as frequency distribution analysis and spectral/time series analysis, the CMS will include receptor modeling modules, such as the Chemical Mass Balance (CMB) receptor model (Watson et al., 1990).

The CMS will also include a set of statistical routines and a detailed protocol to evaluate the performance of meteorological and air quality models. These performance evaluation modules will compare model predictions with field measurements and provide sophisticated analysis and interpretation of the differences. The comparison will include proper consideration of measurement errors and spatial/temporal representativeness of measurements. The performance evaluation modules are expected to be based in part on the work by Alcamo and Bartinicki (1987), Hanna (1988), and others, as further discussed in Section 2.1.3.2.

As a decision support tool, a CMS must facilitate decision analysis in the face of uncertainty. To do so, it initially must provide a mechanism for estimating the uncertainty and variability in the inputs to its modeling components (e.g., emissions or boundary conditions) and the uncertainty inherent in any parameters, approximations and parameterizations used in the component formulations (e.g., the advection scheme or the chemical reaction rate expressions). Then it must allow the overall uncertainty (say, as a probability of a modeled outcome) in specific outputs to be estimated based on the input and formulation uncertainties and variabilities. Although, the importance of a capability for modeling uncertainty estimation to assessing risk in a CMS is recognized, this particular aspect was considered beyond the scope of this study and is not discussed in any detail.

Air Pollution Effects

The CMS is expected to include the simulation of some adverse effects caused by air pollutants. Particularly important is the treatment of atmospheric visibility issues. Two major scenarios have been addressed by visibility modeling techniques: 1) plume visibility, i.e., the simulation of the visual effects of a single plume; and, 2) regional haze, i.e., the simulation of the visibility impairment caused by large air masses loaded with fine particles. (Regional haze, depending upon atmospheric conditions, can be layered or relatively uniform.) CMS will include modules for the numerical simulation of both scenarios.

One of the major goals of many air quality studies is the assessment of the health risk caused by chemical emissions from anthropogenic sources. This assessment requires the mathematical modeling of a variety of processes, including population exposure to these chemicals, their associated doses, and health effects. While, at least initially, we do not expect the CMS to include dose-response calculations and mathematical modeling of the chemistry of human body and health effects, exposure simulation will be a component of CMS.

Population exposure models have been recently reviewed by Seigneur (1994) as part of a series of articles on mathematical models for health risk assessment. Population exposure comprises of two components: 1) the movement of different individuals or groups of individuals through different micro-environments; and, 2) the exposure to chemicals in these different micro-environments. These models can be either empirical (i.e., based on statistical analysis of exposure data) or theoretical. Theoretical models may assume a static or dynamic population.

Numerical Methods

Two of the core components of the CMS are the meteorological and air quality models, which in turn can be viewed as a collection of numerical methods used to integrate their respective descriptions of the physics and chemistry of atmospheric systems. Because of their central importance, these numerical methods are evolving, improving both accuracy and computational efficiency. In current air quality model systems that are

grid-based, two of the more important routines are the advection routine solver and the chemical integrator. In future systems, aerosol dynamics solution methods will become more standard and will be supported by the CMS.

Design and Development Issues

Design and development issues have been introduced in the CP. A comprehensive and current discussion on these topics is presented in Chapters 2 through 5.

CMS DEVELOPMENT PLAN

The development plan for the CMS consists of two phases:

- Phase I - Framework design (current effort: June 1994-February 1996)
- Phase II - CMS development (future effort)

In addition to the two phases above, a parallel effort (PE Task) is under development. Under Task PE, an application prototype (Prototype #1) of the CMS is being developed.

Phase I has produced the following deliverables:

1. Concept paper. The CP was a pilot document that set the stage for the full design of the CMS. It discusses the goals of the CMS and its future use as a continuously evolving system. The Table of Contents of the CP is enclosed as Appendix A.
2. Videotape¹. The CMS Concept Video serves as a video "executive summary" of the CP. It also summarizes the current state of modeling (and its numerous frustrations and excessive costs) as well as portrays the range of potential applications and benefit of the proposed CMS system. The videotape is expected to be finalized in April 1996.
3. User survey.

¹ The CMS Concept Video is setting the stage for the next advancement in technical communications – the video technical or management report. Since the results from the CMS will be so inherently visual, in addition to the always needed text documents, the companion video (here construed as either conventional tape and/or newer media such as CD-ROM) will graphically portray the model's results in ways simply not achievable with text and static illustrations. The CMS will have as an important goal the ability to generate visuals which will readily summarize the model's calculations and illustrate them not only to the user, but management, regulators, and ultimately, the public. Desktop video is a technology that over the next several years will become as revolutionary as desktop publishing. Its impact is likely to be even greater. The CMS system has been designed to take advantage of the impending desktop video revolution. The concept video is a foretaste of what is to come not only in modeling but technical communications.

4. CMS framework design report (i.e., this report). This is the main result of Phase I.

The scope of work of Phase II is discussed in detail in the following chapters.

EXECUTIVE SUMMARY

CAMRAQ AND THE CMS

The Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ) is sponsoring an initial effort toward the development of a Comprehensive Modeling System (CMS) for air pollution. This framework design report was prepared as the final product of this initial effort.

A previous concept paper report (Zannetti et al., 1995; hereafter referred to as CP) was prepared as part of this project. The CP was a pilot document to set the stage for the full design of the CMS. (We enclose the Table of Contents of the CP as Appendix A.)

WHAT IS A CMS?

The CMS must provide an infrastructure that helps its users to do their jobs better and faster, whether those jobs be regulatory and policy analysis, source impact assessment, understanding atmospheric chemistry and physics, or performing atmospheric research studies. As such, the CMS has been designed to provide the following:

1. A platform – for modeling pollutant emissions, atmospheric physics and chemistry, and the impact of pollution – in as scientifically sound a fashion as is desired or possible.
2. A readily accessible interface, so that its use is a benefit, not a distraction.
3. A powerful set of analysis and decision-support tools, be they graphical, visual, economic, or scientific, including report preparation.
4. A method to make maximum use of the available computational resources, including CPU power, disk storage, and communication systems.

The CMS has been designed in a way that facilitates its continuous evolution with science, computer capabilities, and user needs. In particular, the CMS must include the best science and be *scientifically credible*; facilitate control strategy planning; be readily extensible; be easily used; take advantage of evolving computational environments; be robust; and be able to interact with and benefit from other modeling systems, such as EPA's Models-3.

The CMS has been designed to be used both by regulatory users and researchers. By regulatory users, we mean the regulated entities, the environmental consulting firms

they employ, and the regulating agencies, both at a local and national level. By researchers, we mean those who work to develop new modeling and analysis techniques and apply the latest techniques to gain new scientific insights.

Currently, air quality models are used to address a variety of environmental issues, such as emission permitting, ozone control, acid deposition, particulate matter, air toxics, emergency planning and response, human exposure, and risk assessment. Each of these issues is treated with different sets of modeling tools with varying levels of scientific detail and complexity. The benefits of the CMS are multifold: the latest scientific models will be available, resulting in decisions being made using our most current scientific understanding; the user-friendly graphical user interface (GUI) will greatly increase the number of potential users and their access to more sophisticated modeling tools; policy decisions will be made using the latest science, decreasing the risk associated with making the wrong decision; and the costs of model applications will be greatly reduced because of the ease of use of the CMS.

CMS DEVELOPMENT PLAN

The development plan for the CMS consists of two phases:

- Phase I - Framework design (current effort: June 1994-February 1996)
- Phase II - CMS development (future effort: 1996-1999)

In addition to the two phases above, a parallel effort (PE Task) is under development. Under Task PE, an application prototype (Prototype #1) of the CMS is being developed and is expected to be completed in February 1996.

Phase I has produced the following deliverables:

1. Concept paper. The CP was a pilot document that set the stage for the full design of the CMS. It discusses the goals of the CMS and its future use as a continuously evolving system. The Table of Contents of the CP is enclosed as Appendix A.
2. Videotape. The CMS Concept Video serves as a video "executive summary" of the CP. It also summarizes the current state of modeling as well as portrays the range of potential applications and benefit of the proposed CMS system. The videotape is expected to be finalized in April 1996.

3. User survey.
4. CMS framework design report (i.e., this report). This is the main result of Phase I.

THE CMS AS A CONCEPT

The CMS concept was developed with two main objectives in mind.

- 1) To create a tool for a system-assisted, user-friendly use of advanced air quality models, in order to facilitate the use of the best science in regulatory decisions.
- 2) To provide a software platform for the application and the incorporation of the most advanced air pollution models, in order to facilitate scientific interactions, exchanges, and new developments.

THE CMS AS A SYSTEM

The fully-developed CMS will be accessible from the user's workplace through a computer interface, either workstation- (Unix) or PC-based. The user's interface envisioned for this purpose, includes visualization software for displaying two- and three-dimensional representations of input and output data. It also includes geographical information system (GIS) capabilities for accessing and displaying data according to geographical location. Powerful statistical packages will be available for summarizing, synthesizing, and analyzing data, as well as extensive on-line help files to minimize (or avoid) reliance on operator's manuals. Not necessarily on-site, but accessible through this user's interface, are utilities and facilities, e.g., for producing video copies of animated visualizations of the data.

By pointing and clicking with a mouse, this envisioned interface will allow the user to select a model configuration (e.g., type of emissions model, type of meteorological model, type of air quality model) consistent with the intended application. If desired, the user can also select individual modules (e.g., preprocessors, chemical mechanism, boundary layer treatment, advection scheme) other than the default set. Having configured the model, the user then can select the geographic domain and time period. Then, the corresponding emissions, meteorology, air quality, topography, land use, and demographic data can be selected as needed, often from a geographically distributed archive via the Internet.

The CMS component models will be executed through the user's interface, as well as evaluated using on-line comparative and diagnostic tools. The user can generate tabular, graphical, and textual reports based on the data also via the interface. All of this can be done interactively with other collaborators through the interface's digital TV/ networking capability. Of course, such a capability also will support E-mail and easy file transfers.

Numerical models available from the CMS fall roughly into the categories of emissions, meteorological, dispersion, and air quality, and will be selected based on the application and data availability. They could be applied to assessments on local, urban, regional, or global scales of emergency response, population or ecosystem exposure, or response of visibility, ozone, deposition, and/or particulate matter to emissions control. In particular, special modules in the CMS will allow the system to assist during accidental releases of air pollutants and emergency situations. These modules will provide, among other things, real-time access to on-line meteorological data and satellite images.

THE USER PERSPECTIVE

The CMS user sees the system through a series of interactions with the facilities provided by what should essentially appear to be a “black box” containing all the required functions. The primary interface will be through a set of standard graphical user interfaces (GUI) which promote the effective use of the system by even uninitiated users.

One user’s view of the “CMS black box” is displayed in Figure 2-2. The users will focus on the functions offered by the CMS through the GUI. The users may be aware of the demands their work may make on system resources, but the CMS will reflect the availability and cost of those resources to the user in the course of CMS usage.

SCENARIOS

The following scenarios describe the cases which have been identified as design points for the CMS.

- baseline simulation
- sensitivity/uncertainty
- new science
- control strategy
- permitting
- exposure/risk assessment/cost-benefit analysis
- emergency preparedness and response

These seven scenarios show examples of how the CMS will be used. There are, of course, other applications in which air quality, emissions, and meteorological models are currently used. The CMS could also take on those duties. Also, we fully expect that more uses will be identified. For one, long-term evolution of the CMS is expected to be towards multimedia pollution impacts, i.e., the simulation of transport and fate of pollutants in air, water, soil, groundwater, and biota.

THE USER INTERFACE

The CMS user interface must provide human interaction to persons with varied levels of expertise in their own fields while possessing, in some cases, small literacy in the vernacular of the computer industry. One of the principle requirements discussed previously is the concept of “transparency” wherein the user deals with the CMS in exactly the same way regardless of the location and type of databases and computer systems being employed at the time. An important system requirement is that the user interface, as all other components of the CMS, be “platform neutral” or “platform independent”.

There are several readily available software systems which support the creation and operation of a GUI to provide the user interaction. Specifications for this system component may be found in Section 3.5.1.1 of this document. The user will see an identical interface on every terminal that they will use to access the system, whether it is a PC running Windows 95, Windows NT, Mac OS, or OS/2, or whether it is a Unix workstation running a manufacturer’s own implementation of Motif.

This “GUI” will present choices, accept and interpret user inputs, draw the operator through a series of steps to initialize, execute models, and supporting software, store and retrieve data, and analyze and publish the results. Figure 2-4 contains an example of a sequence of screen displays which are the consequences of a user starting up the CMS, selecting the photochemical modeling component, and choosing the Los Angeles basin and its database as the object of this particular run.

CMS SYSTEM SERVICES

The CMS System Services contain the real heart of the CMS. The system services operate directly on the data. In this regard, the internal design of the CMS could be termed “data centered”. The data in this case contains not only the obvious cases of model input and output data, but the programs themselves, and auxiliary data such as GIS and other graphical mapping information. Thus, an important component of the CMS will be a comprehensive database management system which is capable of dealing with a variety of data formats, providing conversions where necessary.

The database system characteristics are specified in detail in Section 5.3, but it is important to highlight here that this system must be able to provide internal security and integrity checks, even when the databases or portions thereof, are geographically scattered and accessible only through relatively unsecured network connections. Since one of our principal goals is for the CMS to be “platform independent” it is essential that this program adopt a standard database system which is demonstrably “platform neutral”.

MANAGEMENT OF PROCESSES AND COMMUNICATIONS

If the CMS was only to exist on a single computing system and be accessed by a single user at a time it would be possible to implement most of the system as a collection of programs and subroutines. The previous discussions make clear that a far more robust system is required. Hence, the CMS is divided up into entities which contain one or more programs. Many of these entities are semi-autonomous and may be executed on one or more computer systems. They can communicate with other entities and perhaps even invoke services on other computing nodes.

The relationship between a multiplicity of asynchronous processes can create chaos in even moderately complex systems. To avoid this chaos, a set of formalisms have been developed over time to describe and bound the creation, operation, and interaction of such entities. Fortunately, the CMS is far from unique in many of its requirements. Systems as varied as on-line airline reservations to banking systems share many of the same needs. In response to this, a significant effort in this area has resulted in the Open Software Specification of the Distributed Computing Environment (DCE) which is now provided by every vendor of computing hardware and operating systems, including Unix systems as well as the PC-based Windows 95, Windows NT, Mac OS, and OS/2. Thus, the underlying “backplane” of the CMS infrastructure is the DCE, providing a platform-independent system and schema for managing the entire CMS.

FILE SYSTEMS, SECURITY, AND INTEGRITY

Underlying the database management system will also be a standard mechanism for securing the integrity and access to the data files within the CMS. The DCE provides a basic distributed file system (DFS), which can be augmented with Kerberos security mechanisms. Again, these systems are available on all computing platforms and have become industry standards, so that “platform neutrality” of all of the base components of the CMS is assured.

IMPLEMENTATION PLAN

In our implementation plan, the CMS will be developed as a series of increasingly comprehensive prototypes.

The premise for our implementation approach is that it is not practical, and arguably not possible, to specify ahead of time every implementation detail and anticipate the consequences of every interaction between system components for a system such as the CMS. By developing the CMS as a series of increasingly comprehensive prototypes that are tested by actual users, we can adapt the system as we go, that is, evolve the system. This is distinctly different than the concept of “learning” prototypes. A learning prototype is considered a throw-away, designed to help understand some aspect of the problem, to fill in some missing pieces in what is still a top-down design process. Our prototypes are intended to be on the main evolutionary branch to the full CMS. In the case of the CMS, we start with a good understanding of the users’ functional

requirements for the system. From this we can develop a rough "system design" based on the team's experience in other software development projects. This reflects the state of our knowledge at the beginning of the CMS development process. By adopting an adaptive approach, we can make adjustments as we go, based both on our increasing knowledge and on changing external factors, such as new hardware and software products or revised user requirements.

Based on the experiences of the CMS design team members, there are at least five major factors that go into deciding the design goals of a particular prototype version:

1. Prioritization of the needs of the user/funder community
2. Availability of existing components,
3. The degree to which a particular aspect of the system is fundamental (i.e., it needs to be in place in order to integrate other subsystems/components)
4. Lessons learned (and fixes required) from the previous prototype
5. Level of funding available

Each version must present new air quality modeling functionality to the user. That is to say, that a new version should not only provide upgrades to the underlying foundation of the system, but also allow them to perform additional tasks. Thus, for each new prototype there is a delicate balance between new user-oriented features, structural improvements, new platforms, and new documentation; all on a limited budget. This should sound familiar to anyone who has ever been associated with developing commercial software. That is probably another good way of thinking about our implementation strategy: the CMS must compete every step of the way with other governmental and commercial development efforts. Hopefully, in many cases, the most competitive strategy will be to find ways to cooperate. The fact that the CMS must remain competitive is a healthy dose of reality for the implementers. Without this kind of "real world" stress, projects like the CMS risk turning into exercises not grounded in reality.

PROTOTYPE #1

Prototype #1 is expected to be completed in April 1996. It was developed to assist, as a parallel effort ("fast prototyping"), in the design of the CMS framework, to provide a tool for presentation/demonstration/marketing, and, most importantly, to perform actual simulation studies and be used by CAMRAQ members and other parties for practical air quality evaluations.

The prototype possesses five major characteristics:

1. Its primary focus is on the issue of tropospheric ozone.
2. It fulfills some of the needs not currently satisfied by available systems and prototypes.
3. It is portable and able to be installed on different platforms.
4. It is easy to run through a user-friendly interface.
5. It contains advanced 3D visualization features.

The primary challenge in defining the first prototype version was to put together a system on a very limited budget that displayed some of the most important aspects of a CMS. At its most basic, a CMS must provide users with the capability to setup and run an AQM with an easy to use GUI, and to interactively analyze model outputs versus observations. The model of most immediate significance to the CAMRAQ group was the UAM-IV photochemical model as it is being widely used in the production of state SIPs for the minimization, and eventual elimination of ozone episodes. It is also being used in Canada and Europe. As it was not possible within this phase to develop a full GUI to all aspects of UAM-IV setup, it was decided to provide a limited (but still useful) capability for altering the emissions inputs into the UAM-IV. UAM-IV was chosen as the first model because of its wide usage. Other photochemical models can be added with relatively minor adjustments.

PROTOTYPE #2

In Version 2.0 of the CMS prototype we would build onto the Version 1.0 base by incorporating more of the infrastructure described in the System Design (Chapters 2, 4, and 5) portion of this document, and expand the modeling-oriented functionality. This was anticipated in the design of Prototype #1, as the interface has facilities for the expansion into other models and model types. The actual screens developed for Prototype #1 would be retained, for the most part, but the underlying code would be "serverized", that is, broken into client and server processes that send requests and services through an interprocess communications mechanism. This is essential to allow the application to be distributed across multiple machines, allowing PC-based CMS users to access functionality that currently has only been implemented on workstation or "supercomputer" machines.

Prototype #2 would include a full emission model, 2 diagnostic meteorological models, 2 photochemical air quality models, and a Monte-Carlo Lagrangian particle model.

PROTOTYPE #3

In Prototype #3 we would add the Model Servers and the CORBA system¹. The major change is that the CMS would have knowledge of the model and could present the user with a GUI for model setup. (see Section 5.5) This is a very big step in the direction of making it easier to create model runs. There would be a registry function whereby new or legacy codes could be registered into the CMS system. This process includes the creation of a GUI template that the CMS would use to define the model setup interface, as well as a catalog of data and computational resource requirements. In addition, links would be established between the CMS DBMS and a GIS system with access to geographically registered information and GIS analysis capabilities.

Additional models (e.g., a prognostic meteorological model, such as RAMS, and a non-UAM-based photochemical model, such as SAQM) would be added to Prototype #3.

CMS PRODUCTS

After the development of Prototype #3, we expect real CMS "products" to be provided. These products would follow the specific design structure presented in Chapters 2, 4, & 5. We anticipate the development of a "basic" CMS product, to be followed by an "intermediate" CMS and a "full" CMS.

¹ Object management mechanisms are a software construct used to make it easier for external programs to interact with a set of procedures and data. This is done by providing some degree of encapsulation (see Appendix B). A common example is the OLE System from Microsoft. This is a rudimentary object management mechanism used to exchange data between Microsoft Windows Programs via the clipboard. It provides a means of packing the data along with a description of the data and how they should be read. This allows a user to ingest a set of database data into a spreadsheet, for example. A more sophisticated example might be the use of the SOM Object Management System in OS/2 to allow the user to "drop and drag" a document icon onto a printer or fax icon. Here the objects contain both the data and the routines (or methods in object-oriented lingo) used to access the data. The common object resource broker architecture (CORBA) is a more sophisticated object management mechanism developed by the Open Management Group. It is intended to be a platform-independent system. As its name implies, the CORBA provides a repository and a brokerage "service" for all the objects under its control. This "service" ranges from ensuring proper associations between objects which are activated by user requests, as well as managing the necessary interlocks and registries necessary for system coherence. For example, an object may be a temporal database which needs updating by one process while yet another process needs to access the same object. The CORBA might block either requester until the other one is complete or it might initiate a copy of the object so that both requests can be satisfied at the same time. It is our plan to utilize CORBA in the development of the CMS. We must be aware, however, that industry agreed upon standards often get swept away by competing commercial entities; especially if the competition is Microsoft. In this case the competition is between CORBA and a newer, more robust version of OLE being promoted by Microsoft. The CMS development team will need to remain flexible on this issue and see how things play out.

MANAGEMENT PLAN

The development team would work with a clear management structure and precise allocation of responsibilities. The development effort would be performed by a consulting team of scientists, led by a Project Manager/Principal Investigator (PM/PI) having the overall technical and budgetarial responsibility for the project and reporting directly to CAMRAQ Project Manager. The PM/PI would provide general guidance and supervision and be in charge of resolving conflicts, when required.

Several subcontractors would work on the project (see Figure 6-1). Each subcontractor group would be led by a Group Leader (GL) reporting to the PM/PI. The GL would be responsible for the technical performance of the group and the financial issues associated with the subcontract.

Each project effort would be subdivided into tasks. Each task would be performed by a “leading” team of scientists, from one or more groups, and monitored by a “monitoring” team (see below). The leading team for each task would be managed by a Task Leader (TL) reporting directly to the PM/PI. The TL would be responsible for the technical performance of the task team and the financial issues associated with the task effort.

The development team would adopt the matrix management approach. Each scientist working on the project would report to two managers, the GL (who would not change) and the TL (who would change depending upon the specific task in which the scientist is involved).

The CMS development team would operate with a certain degree of “planned redundancy” and with the use of “task monitoring” teams (see Figure 6-2). Planned redundancy means that two teams would be selected for each task. The first team would be the “leading” team, in charge of performing the actual work. The second team would be the “monitoring” team, but as capable as the leading team of performing the task. The monitoring team would be led by a Task Monitor (TM) who would report directly to the PM/PI. In each task, the monitoring team would dedicate a level of effort of 5-10% in proportion to the development effort of the leading team actually doing the work. This monitoring effort would assure that the leading team is operating in a cost-effective manner, providing results on time and on budget and in agreement with other parallel efforts. If necessary, the monitoring team would be able to provide extra support to the leading team and, in extreme cases, take full control of the task effort.

In addition to the management structure discussed above (PM/PI, TLs, and TMs), the project team would include four “oversight” leaders with the following specific functions (see Figures 6-1 and 6-2).

- Regulatory Oversight Leader (ROL), reporting to the PM/PI with the general responsibility of monitoring the regulatory aspects and implications of the CMS. The ROL would make sure that the regulatory community would be

well informed about the CMS and find it useful and applicable to their specific needs.

- Technical Oversight Leader (TOL), reporting to the PM/PI, with the general responsibility of monitoring the technical aspects of the CMS. The TOL would make sure that the CMS remains anchored to the best available science and its technical objectives are properly fulfilled.
- Contract/Subcontract Management Leader (CML), reporting to the PM/PI, with the general responsibility of assuring correct and smooth management of contracts and subcontracts. The CML would also make the effort of minimizing contractual paperwork and allow scientists to concentrate their efforts on technical issues instead of administrative details. The CML would establish simple but effective procedures for contract/subcontract management and progress monitoring.
- Public Relations Leader (PRL), reporting to the PM/PI, with the general responsibility of assuring continuous and effective public relation efforts and external exposure.

All communications among team members would be electronic. There would be a “preferred” communication system (the Internet), an “alternative” communication system (e.g., America On-Line), and an “emergency” communication system (phone, pager, fax). Written communications would be kept at a minimum. The development team would use the most modern methods and software for project and time management. Software productivity tools would consist mostly of Microsoft products.

CAMRAQ MANAGEMENT STRUCTURE

CAMRAQ's management structure has evolved as the organization has grown and is expected to undergo further changes as maturity is approached. At present, CAMRAQ is directed by an executive committee composed of those CAMRAQ members who are contributing financially to the CMS design project. Important functions of the executive committee are to set policy for the consortium, set rules and criteria for CAMRAQ membership, approve access fee structures, and approve new members. Currently, CAMRAQ also includes an associate membership class, which is composed of interested individuals and organizations that do not contribute financially to CAMRAQ operations. While voting privileges and electronic access to CAMRAQ resources are currently restricted to executive members, associate members are encouraged to participate actively in all CAMRAQ meetings.

A new membership class, a “subscriber membership,” is anticipated in the near future, to accommodate those organizations who desire CMS access but cannot contribute financially to CAMRAQ developmental activities. Under this anticipated plan, subscriber members will pay a smaller log-in and maintenance fee for CMS access and

will compose a subset of the associate membership. This anticipated plan also involves formation of a general committee, comprised of members and associate members. This committee will be the primary conduit for communicating to the executive committee community feedback on CMS attributes and capabilities, both existing and desired. It also will formulate, for approval by the executive committee, criteria and rules for CAMRAQ membership, for screening, accepting, and placing visiting scientists, and the schedule of fees for access to CAMRAQ products.

At present, the CAMRAQ coordinator is on contract with EPRI and works closely with the EPRI project manager in the planning and implementation of CAMRAQ initiatives. The coordinator serves as the point of contact for information on CAMRAQ.

CMS DEVELOPMENT EFFORTS

We call Phase II the development phase of the CMS (Phase I was the design). If adequate funding was available Phase II would consist of the following major efforts:

- Effort #1: Development of Prototype #1 (ongoing; completion expected in April 1996) (this effort was previously identified as a parallel effort [PE] but is actually the first step of Phase II)
- Effort #2: Development of Prototype #2 (Apr-Aug 1996)
- Effort #3: Development of Prototype #3 (Sep 1996 - Jan 1997)
- Effort #4: Development of a "basic" CMS (Jan-Dec 1997)
- Effort #5: Development of an "intermediate" CMS (Jan-Dec 1998)
- Effort #6: Development of a "full" CMS (Jan-Dec 1999)
- Effort #7: Periodic maintenance and upgrade (continuous effort from 2000 on)

SCHEDULE AND COST

The expected cost of each effort is presented below.

Effort	Cost (1995 \$)	Period of Performance	Cumulative Cost (1995 \$)
Effort #1	\$100K	(completion expected in Apr 96)	\$100K
Effort #2	\$350K	(Apr-Aug 1996)	\$450K
Effort #3	\$450K	(Sep 1996-Jan 1997)	\$900K
Effort #4	\$950K	(Jan-Dec 1997)	\$1850K
Effort #5	\$900K	(Jan-Dec 1998)	\$2750K
Effort #6	\$850K	(Jan-Dec 1999)	\$3600K
Effort #7	\$200K	(yearly, from 2000 on)	

In comparison with the costs experienced by previous major model development efforts, the cost estimates presented above may appear quite “optimistic”. We believe, however, that these estimates are realistic if the development effort is to be conducted by experienced and dedicated scientists and managed in a firm and cost-effective fashion.

TABLE OF CONTENTS

Abstract	iii
Preface	v
Message to the Reader	v
What is a CMS?	v
What Should a CMS Do?	vi
Intended Users of the CMS and User Requirements	viii
Regulatory Users	viii
Research Users	x
Air Quality Modeling Today and the Benefits of CMS	xi
The Attributes of the CMS	xii
Global Requirements	xii
Applications of the CMS	xii
Components	xii
Emission Modeling System (EMS)	xiii
Meteorological Modeling System (MMS)	xiv
Air Quality Modeling System (AQMS).....	xiv
Statistical Methods and Performance Evaluation	xiv
Air Pollution Effects	xv
Numerical Methods.....	xv
Design and Development Issues	xvi
CMS Development Plan	xvi
Executive Summary.....	xix
Acknowledgments	xxxix
Glossary	xli
Abbreviations.....	xlvii
1. Introduction and Overview.....	1-1
1.1 The CMS as a Concept	1-2
1.2 The CMS as a System.....	1-4
1.3 Summary of User's Requirements.....	1-5
1.4 Summary of Strategy and Goals	1-5
1.5 Evolution of the CMS Concepts.....	1-6
1.6 Framework Design	1-10
1.7 A Guide to this Report	1-11

2. General Design of the CMS	2-1
2.1 Requirements Analysis	2-1
2.1.1 The User Perspective	2-1
2.1.2 System Demands	2-1
2.1.3 Scenarios	2-3
2.1.3.1 Baseline simulation	2-3
2.1.3.2 Sensitivity/uncertainty	2-3
2.1.3.3 New science	2-4
2.1.3.4 Control strategy	2-5
2.1.3.5 Permitting	2-5
2.1.3.6 Exposure/risk assessment/cost-benefit analysis	2-5
2.1.3.7 Emergency preparedness and response	2-6
2.2 System Design	2-6
2.2.1 The User Interface	2-8
2.2.2 CMS Manager	2-8
2.2.3 CMS System Services	2-8
2.2.3.1 Database management system	2-9
2.2.3.2 Program management	2-10
2.2.3.3 Functional services	2-10
2.3 The CMS "Infrastructure"	2-10
2.3.1 Management of Processes and Communications	2-13
2.3.2 File Systems, Security, and Integrity	2-15
2.4 Typical Operation	2-15
2.4.1 The User Actions	2-15
2.4.2 System Services and Actions	2-22
2.5 CMS Interfaces	2-22
3. Implementation Plan	3-1
3.1 Evolution Through Prototyping	3-1
3.2 Prototype Goals	3-2
3.3 Prototype #1	3-2
3.3.1 Platform Independence	3-3
3.3.2 Prototype Operation	3-4
3.4 Prototype #2	3-7
3.4.1 Model Interchangeability	3-8
3.4.2 Operating Within The Distributed Computing Environment (DCE)	3-9
3.5 Prototype #3	3-9
3.6 CMS Products	3-10
4. Detailed Design	4-1
4.1 System Operational Requirements	4-2
4.1.1 Performance	4-2
4.1.2 Reliability	4-2
4.1.3 Interoperability	4-2

4.2	Development and Maintenance Requirements	4-3
4.2.1	Evolutionary Design	4-3
4.2.2	System Flexibility	4-3
4.2.3	Legacy Code Integration.....	4-3
4.2.4	“On-Line Help”	4-4
4.2.5	System Maintainability	4-4
4.3	Principle Design Decisions	4-4
4.3.1	Partitioning of System Functions	4-4
4.3.2	Priorities for Design and Implementation	4-6
4.3.3	Necessary Design Compromises.....	4-7
4.4	The CMS Subdivision/Decomposition	4-7
5.	Major CMS Elements	5-1
5.1	The CMS Client	5-1
5.1.1	CMS Client Components	5-2
5.1.1.1	The GUI.....	5-2
5.1.1.2	The 3D visualization	5-3
5.1.1.3	The CMS interface	5-3
5.1.2	Communications	5-4
5.1.3	Concurrency	5-7
5.1.4	Performance	5-7
5.1.5	Error Handling.....	5-7
5.1.6	Implementation	5-8
5.1.7	Operation	5-8
5.2	The CMS Management Server.....	5-8
5.2.1	Components	5-8
5.2.1.1	Logging	5-8
5.2.1.2	Error control.....	5-9
5.2.1.3	Startup/shutdown	5-9
5.2.1.4	Resource management	5-9
5.2.2	Communications	5-10
5.2.3	Concurrency	5-11
5.2.4	Performance	5-11
5.2.5	Error Handling.....	5-11
5.2.6	Implementation	5-11
5.3	The CMS Database Server	5-12
5.3.1	Components	5-12
5.3.1.1	Location/allocation.....	5-12
5.3.1.2	Geographical information system	5-13
5.3.1.3	Data extraction and insertion.....	5-13
5.3.1.4	Conversion	5-14
5.3.1.5	Transformation	5-15

5.3.2	Communications	5-15
5.3.3	Concurrency	5-16
5.3.4	Performance	5-16
5.3.5	Operation	5-16
5.4	The DCE Distributed File System (DFS)	5-17
5.4.1	Components	5-17
5.4.1.1	Filesets	5-17
5.4.1.2	Cache manager.....	5-20
5.4.1.3	Basic overseer.....	5-20
5.4.2	Communications	5-20
5.4.3	Concurrency	5-21
5.4.4	Performance	5-21
5.4.5	Error Handling.....	5-21
5.4.6	Implementation	5-22
5.4.7	Operation	5-22
5.5	The CMS Model Server	5-23
5.5.1	Components	5-23
5.5.1.1	Registry	5-24
5.5.1.2	Interface templates.....	5-24
5.5.1.3	Model resource requirements.....	5-24
5.5.2	Communications	5-24
5.5.3	Concurrency	5-26
5.5.4	Performance	5-26
5.5.5	Error Handling.....	5-26
5.5.6	Implementation	5-26
5.5.7	Operation	5-26
5.6	The CMS Program/Object Server (Object Management Mechanism)....	5-28
5.6.1	Rationale for the CORBA.....	5-30
5.6.2	An Idealized Object-Oriented System.....	5-30
5.6.3	The Basic Object Broker Task	5-32
5.6.4	Standards	5-32
5.7	The CMS Models.....	5-33
5.7.1	Task 1 – Existing Models.....	5-33
5.7.2	Task 2 – New Models	5-34
5.8	The Distributed Computing Environment (DCE).....	5-34
5.8.1	DCE Client	5-35
5.8.2	Directory Services.....	5-35
5.8.3	Security Services	5-36
5.8.4	Time Synchronization	5-36
5.9	The CMS Batch Server.....	5-36
5.9.1	Components	5-37
5.9.2	Implementation	5-37
5.9.3	Operation	5-38
5.10	The CMS Decision Server.....	5-38
5.11	The CMS Real-Time Server	5-38

6. Management Plan	6-1
7. Schedule and Cost.....	7-1
8. Conclusions and Recommendations	8-1
References	Ref-1

Appendices

A. Title Page and Table of Contents Reprinted from "Concept Paper: Design and Development of a Comprehensive Modeling System (CMS) for Air Pollution".....	A-1
B. A Discussion of Object-Oriented (OO) Design and Programming	B-1
C. Description of Some Existing Air Pollution Modeling Prototypes and Software Products	C-1
D. An Example of Fortran Coding Standards – Fortran Coding Standards for Models-3.....	D-1

ACKNOWLEDGMENTS

This document was prepared as part of an EPRI Project sponsored by the Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ).

We would like to acknowledge the help and guidance of Marianne Causley, Naresh Kumar, Greg McRae, Roger Pielke, Erik Riedel, David Rothenberger, Craig Tremback, Robert Walko, James Wilkinson, and Greg Yarwood in providing input during various stages of the development of this project. Their comments and feedback provided considerable insight into many of the concepts described in these pages. We also acknowledge the support provided by Jake Hales of ENVAIR – the CAMRAQ Coordinator. The EPRI Project Manager was Alan Hansen.

We appreciate the support provided by Nadine Russell for finalizing, editing, and producing the manuscript.

GLOSSARY

(this section includes the words and expressions listed in the CP)

Applications Program Interface (API)

System block diagrams containing one or more boxes labeled “API” abound in every monthly magazine devoted to computer-based products and systems. What is an API? It is just what its name describes, an interface for an applications program.

We have had applications program interfaces on our systems since almost the beginning of computer software. A developer writes a program to perform a task. In a special instance, that program might be hardwired into a computer with a single purpose and which begins execution, as soon as the hardware power is turned on. Normally, however, such a program is invoked by entering a command at a computer terminal.

One could say that even the simplest program execution from a command-line entry has an “interface”, its name as it appears on that command line. Of course, most programs require a set of parameters to be provided at the moment they are invoked. So the command-line entry gets a bit more complicated. For example, FOO PARAM1, PARAM2, PARAM3 where our application program is named FOO. Somehow these parameters must be provided to the executing FOO program. Some part of the operating system, which is lurking at the terminal waiting for the typing of the command line, has to acquire the parameters PARAM1, etc., and place those text strings where the FOO program can read them. Thus, even the most basic operating system function provides an “application program interface”.

Viewed in this light then, the more sophisticated technique of providing an interchange between a user and a program, from simple question and answer sessions, using typed-in entries to the terminal or the most beautiful graphical display with menus, buttons, dials, and sliders, are all APIs.

What if the program is not initiated from a command line terminal entry? Some other program in the system will perform a function similar to the command line, string up FOO and ensuring that it gets the PARAMN, parameters delivered to it. This too is an API.

In the CMS, the API becomes more than the simple exercise we have discussed here. As programs such as the “legacy” codes are integrated into the CMS, some form of API will be provided to bridge the gap between the environment and parameters which the program expects and the environment developed for the CMS.

Backplane

In the electronics industry a “backplane” provides the interconnection of all active electronic components. This used to be a mat of wires and is now commonly a printed circuit board, like the “motherboard” in a PC.

Component

Any part of the CMS that is intended to fulfill a specific function, e.g., the visualization component, or the emissions component. The CMS will consist of a set of components and the connections between these components. It will be possible to make use of several different modules to fulfill the functionality of a certain component.

Comprehensive Modeling System

The system to be developed as a result of this framework design. The Comprehensive Modeling System (CMS) will be used to refer to the specific system that is the focus of this concept paper. The term *modeling system* will be used to refer to the class of similar systems in development elsewhere or to the generic concept of such a system.

Framework Design

In the context of this report, *Framework Design* indicates a set of development guidelines for CMS, and not an extremely detailed specification procedure for CMS coding, interfaces, and windows specifications.

High End PC

By *high end PCs*, we refer to the new types of PCs based on chips like the PowerPC or Pentium processor, and running the newer operating systems like Microsoft Windows/NT, IBM OS/2, and Apple OS.

Instantiation

“Instantiation” is the creation, or starting up, of a new copy (or instance) of a program, object, or process in a computing system.

Model

Embodies a quantitative description of the functioning of the real world (usually in some form of computer code, at least in a set of equations). A model simulates some part of the physical world and can be used to explore hypotheses about the effects of various changes on the physical world without experimenting directly in the physical world.

Module

A specific component of the CMS. For example, the visualization package AVS could be a *module* used to provide the functionality of the visualization component.

Processes

The terms “process instance”, “inter-process communications” & “process management” are found in any modern computer system, no matter how primitive its operating system may be. A “process” is an executing program which is started by the operating system. In some cases, several copies of this program are running at one time. Each executing copy is an “instance” of that program and the procedure for a program startup may be called its “instantiation”.

Processes on platforms from PC to CRAY possess certain characteristics:

- a. A process owns its memory and is protected from incursions by other processes by the combination of hardware and operating system software.
- b. A process has several states it can be in: starting up, terminating, running, or waiting for service.
- c. Processes can exchange information with other processes on the same host computer through a feature called “shared memory”. Wherein a portion of both processes’ memory address space is created as a shared memory segment.
- d. Processes can exchange data with other processes executing on the same host or on a separate computer system through some form of message passing between the two processes.

The operating system provides primary management of processes running on their hosts, and in addition, provides tools for other processes to participate in the management of processes. Simple examples are one process starting or terminating another.

Note that, in the main, processes are quite independent from each other and are protected from each other. They can run without synchronizing with other processes, if desired. “Processes” are the basic building blocks of distributed computing systems.

Public Domain Software

We define public domain software as all freeware software (i.e., software that does not require a license and that anyone can legally download from other machines) plus all those environmental models and codes that are normally distributed free of charge or at a nominal fee.

Resolution

The smallest scale physical feature that can be properly represented in a particular model usage. It is related but not equivalent to the model mesh size. For a meteorological model for example, the effective resolution is at least four times coarser than the mesh size (Δx). A model running with a 10 km Δx only begins to effectively resolve atmospheric features with a horizontal scale on the order of 40 km or larger.

Security

A fundamental property of the CMS, which will make it possible for users to utilize all the resources of the system regardless of their geographical location, is the assurance of security for the programs and data for each user. The distributed computing environment (DCE) provides an extensive security system for user access to servers, as well as user access and controlled sharing of the CMS filesets. This security scheme is based on a MIT development named "Kerebros 5" (K5) and represents a widely accepted technology for computer systems. The title "Kerebros" is taken from Greek mythology and the name for a two-headed, very "aggressive" dog who protects its master from all enemies. In the MIT system, a "master" computer is responsible for all security actions and must be physically secured from all but the most trusted human access. MIT describes the ideal for this physical isolation as "locking the master computer in a closet, which is guarded by a very irritable dog ... Kerebros".

In such a system, once the master computer's physical integrity is assured, a software system can be put in place which will assure comprehensive security of all of the system which it manages, no matter where the other members are located.

The DCE reliance on K5 introduces some complications in normal operating systems. No longer is the user's log-in name and password managed by the CMS client host. Instead, the host carries on a dialogue with the security "master" computer, exchanging over the network only encrypted information such as passwords. This encryption is of a "one-time" form so that even the most ambitious "hackers" cannot obtain this vital data.

Although K5 is a core element in DCE security, Kerebros-based security systems will be used on most new computer applications where security is a concern.

Subsystem

Used interchangeably with the term component.

Threads

A thread is an executing program which has most of the properties of “processes” described above. The major difference between a thread and a process is that a group of threads within a process share direct access to the same identical memory locations. This makes it possible to implement very efficient communications between different executing programs but without the protection afforded by the completely independent memory spaces which a simple process possesses. Systems which provide thread execution capability must also provide a number of functions for managing the relationship between threads and resolve conflicts in access to common memory areas. “Threads” have become such a powerful and necessary component of modern computing that a POSIX standard has been established (called “pthreads”) for which all operating system vendors are now developing software support.

ABBREVIATIONS

(this section includes the abbreviations listed in the CP)

3D	Three-dimensional
4DDA	Four-dimensional data-assimilation
A&WMA	Air and Waste Management Association
ACL	Access control list
ADE	Atmospheric Diffusion Equation
ADOM	Acid deposition and oxidant model
API	Application programmer interface
API	American Petroleum Institute
AQM	Air quality model
AQMP	Air Quality Management Plan
AQMS	Air Quality Modeling System
ARPS	Advanced Regional Prediction System
AUSPEX	Atmospheric Utilities Signatures: Predictions and Experiments
AVACTA II	AeroVironment Air Pollution Model for Complex Terrain Applications
BIOEM	Biogenics Emissions Estimates Model
CAMRAQ	Consortium for Advanced Modeling of Regional Air Quality
CAPS	Center for Analysis and Prediction of Storms
CB-IV	Version IV of the Carbon Bond Mechanism
CBM-1	Carbon Bond Method or Model
CIT	Carnegie &-California Institute of Technology
CMB	Chemical mass balance
CMS	Comprehensive Modeling System
CMSA	Comprehensive Modeling System Application
CMU	Carnegie Mellon University
CO	Carbon monoxide
CORBA	Common object resource broker architecture
COST	Crude Oil Storage Tank Emissions Estimates Model
CP	Concept Paper (see Zannetti et al., 1995)
CPU	Central processing unit
DBMS	DataBase Management Server
DCE	Distributed computing environment
DFS	Distributed file system
DICCE	Distributed informatics, computing, & collaborative environment
DOE	Department of Energy
DWM	Diagnostic Wind Model

EMFAC	California Air Resources Board's Fortran model for computing mobile source emissions factors
EMS	Emissions Modeling System
EPA	Environmental Protection Agency
EPRI	Electric Power Research Institute
EPS	Emissions Processing System
ERDAS	Emergency Response Dose Assessment System
ESnet	Energy Sciences Network
EWB	Environmental Work Bench
FaAA	Failure Analysis Associates, Inc.
FREDS	Flexible Regional Data System
GB/s	Gigabytes per second
GEMAP	Geocoded Emissions Modeling and Projection
GEMS	Geographic Environmental Modeling System
GIS	Geographic Information System
GUI	Graphical user interfaces
HPCC	High-performance Computing and Communications
HPF	High-performance Fortran
HYPACT	Hybrid Particle and Concentration Model
IC/BC	Initial concentrations and boundary condition
IEEE	Institute of Electrical and Electronic Engineers, Inc.
I/O	Input/output
ISC	Industrial Source Complex
JEWEL	Joint Environmental Workspace and Emission Laboratory
KSC	Kennedy Space Center
LADCo	Lake Michigan Air Directors Consortium
LAN	Local area network
LLNL	Lawrence Livermore National Laboratory
LMOS	Lake Michigan Ozone Study
LP	Linear programming
LPDM	Lagrangian Particle Dispersion Modeling
LULC	Land use/land cover
MILP	Mixed integer linear programming
MINLP	Mixed integer nonlinear programming
MM5	NCAR/Penn State Mesoscale Meteorological Model, Version 5
MMS	Meteorological Modeling System
MOBILE	EPA's Fortran Model for computing mobile source emissions factors
MoVEM	Motor Vehicle Emissions Estimates Model
MPP	Massively parallel processor
NAPCA	National Air Pollution Control Administration
NCAR	National Center for Atmospheric Research
NCDC	National Climatic Data Center
NLP	Nonlinear programming
NO _x	Nitrogen oxides
NSR	New source review

NQS	Network Queuing System
OLE	Olefinic bond
OMG	Object Management Group
OOD	Object-oriented design
OSF	Open Software Foundation
PAR	Paraffinic bond
PBL	Planetary boundary layer
PC	Personal computer
PE	Parallel effort
PM	Particulate matter
PSD	Prevention of significant deterioration
PVM	Parallel Virtual Machine
RADM	Regional Acid Deposition Model
RAMS	Regional Atmospheric Modeling System
RAPS	EPA's Regional Air Pollution Study
RMP	Risk management plan
ROM	EPA's Regional Oxidant Model
RPC	Remote procedure call
RPM	Reactive Plume Model
SAI	Systems Applications International
SAIC	Science Applications International Corporation
SAQM	SARMAP Air Quality Model
SARMAP	SJVAQS/AUSPEX Regional Model Adaptation Project
SCAB	California South Coast Air Basin
SCAQMD	California South Coast Air Quality Management District
SGI	Silicon Graphics Inc.
SHASTA	Sharp and Smooth Transport Algorithm
SIP	State Implementation Plan
SJVAQS	San Joaquin Valley Air Quality Study
SMP	Symmetric Multi-Processing
SMS	SARMAP Modeling System
SSESCO	Supercomputer Systems Engineering and Services Company
UAM	Urban Airshed Model
UAM-IV	Standard version of the EPA's Urban Airshed Model
UAM-V	Version V of the Urban Airshed Model
UPS	UAM Postprocessing System
URM	Urban-to-Regional Multiscale Model
USGS	US Geological Survey
UTM	Universal Transverse Mercator
VMT	Vehicle miles traveled
VOC	Volatile organic compound

1

INTRODUCTION AND OVERVIEW

We use numerical models of the atmosphere for a variety of purposes for which we have few or no reliable alternatives: forecasting the weather, studying the interaction of meteorology and chemistry, predicting how air quality will respond to emissions changes, or how downwind exposures are related the emissions.

However, useful atmospheric modeling, especially when nonlinear chemical reactions are included, is fraught with difficulties and challenges. As we strive to increase the realism of the simulations by including more detailed representations of atmospheric processes at finer spatial resolution, the computations get more complex and time consuming. Their sheer complexity makes them prone to coding and conceptual errors. To deal with the computational demands, the most complex air quality models are typically run on supercomputers. For photochemical simulation, for example, there are only a few relatively complex air quality models in use today, yet their dissimilarities are probably greater than their similarities. They often use different advection schemes, coordinate systems, diffusion parameterizations, chemical mechanisms, numerical solvers, and (if they are included) radiation transfer schemes, and cloud and aerosol process representations. Generally they are not particularly well documented, not easy to modify, and not rigorously evaluated, to the extent that we know their relative performance, how well they are doing, and for what reasons. It is rarely easy to incorporate modeling code developed elsewhere into one's own model, hindering the transfer of modeling technology within the community. The job of preparing their input data is enormous. The tedium and intensity of this exercise presents the opportunity for introducing inadvertent errors. Finally, comprehensive air quality simulations today require the sequential application of different models (at least a meteorological model and an air quality model). So far, general interfaces have not been developed and, consequently, the use of model outputs as model inputs is difficult and time consuming.

With the overall goal of addressing these and other modeling problems, the Consortium for Advanced Modeling of Regional Air Quality (CAMRAQ) was formed to jointly develop a Comprehensive Modeling System (CMS) for research and for conducting air

Although many organizations have participated in the formation and definition of CAMRAQ over the years, only a subset is presently supporting the CMS framework design effort: EPRI, US Department of Energy, Environment Canada, Ontario Ministry of the Environment & Energy, American Petroleum Institute, Pacific Gas & Electric, Southern California Edison, and the US EPA. Representatives from these organizations have the opportunity to ensure that the design addresses their particular needs. EPRI serves as the contracting agency for all members of the group except EPA, who contracts separately.

quality assessments. Under sponsorship of a subgroup of CAMRAQ members, a CMS design team was formed. The design team has prepared this document which provides the Framework Design for the CMS. In the near future, a CMS development team will be formed. This team will use the concepts and guidelines presented in this document to develop the CMS.

1.1 The CMS as a Concept

The CMS concept was developed with two main objectives in mind.

- 1) To create a tool for a system-assisted, user-friendly use of advanced air quality models, in order to facilitate the use of the best science in regulatory decisions.
- 2) To provide a software platform for the application and the incorporation of the most advanced air pollution models, in order to facilitate scientific interactions, exchanges, and new developments.

The first objective clearly aims at improving the current situation in which the most advanced models are excluded from most regulatory applications. This situation is clearly caused by the actual difficulties in understanding advanced models, collecting and preparing the required input data, and interpreting the model outputs. The CMS will make these steps easy for virtually any user and, consequently, will allow the incorporation of the best science in all future regulatory assessments.

The second objective aims at creating a common computer platform and a link among atmospheric scientists and modelers worldwide (and, ultimately, among environmental scientists and modelers, if the CMS is extended to cover all environmental media and not just the atmosphere). This link will be created by providing the atmospheric sciences community with a useful and productive computer platform – the CMS – for scientific research and development. We expect a positive reaction from the scientific community and an extensive use of this “third-generation” air quality modeling system. We expect the scientific community to recognize that, by using the CMS, or some of its utility modules, scientists will be able to strongly improve their productivity. Once the CMS is well established, we also expect a large fraction of the new air quality models to be developed in a CMS-compatible fashion.

Our CMS framework design intentionally aims at satisfying the needs of two communities: the regulatory community and the scientific community. We believe that there is a great advantage in designing a system that is capable of helping both communities, since regulatory applications do need the use of the best available science and scientists need to be aware of the main regulatory concerns which drive a large fraction of the model development activities, especially in North America.

In designing the CMS, we have taken into account users' needs: both those explicitly mentioned in the user survey, that we conducted at the beginning of this project, and

those needs that are dictated by our experience and expectations. It is common for software users to discover, once a product is available, needs that they were not aware they had. We believe that this situation also applies, in part, to air pollution modeling users. The software and hardware capabilities that are emerging from the new computer revolution are indeed impressive and we expect the CMS to make these tools available to virtually any user, even those with limited knowledge of atmospheric processes.

We view the CMS development as an effort in which we have a clear ultimate goal – the final CMS system. However, we achieve this goal through the development of a series of intermediate prototypes. All prototypes have two components: a “demonstration” component and an “application” component. The demonstration component aims at showing the users the comprehensiveness and user-friendliness of the entire design, even those parts that are not fully implemented. The application component aims at providing the users with tools of immediate usefulness and applicability to current problems and needs. In the early prototypes, the demonstration component may have a large role which will fade away, however, in later prototypes. Also, in earlier prototypes, the application component may not be implemented in a fully-consistent, uniform, and elegant fashion, but is expected to provide useful tools of immediate applicability.

The CMS will be an evolutionary system. Particular attention will be paid to the mechanisms which the framework provides for rapid adaptation to changes in user requirements and methodologies, as well as to transitions in regulatory mandates and the explosive growth in computational technologies. At a minimum, the CMS will provide a readily accessible entry to using models employed in addressing urban and regional air quality management issues. As a system, it will facilitate the interoperability of its major scientific components, e.g., emissions processing, meteorological modeling, air quality modeling, and decision support. In addition, the CMS will provide a readily accessible repository for the related data. Finally, it will provide an extensible set of analysis tools to match the user’s skills, needs, and operating environment. Adherence to standards for software development, communications, security, and inter-processor operations will be fundamental to each aspect of the CMS design. An aggressive stance on adoption and use of these standards is necessary to maximize the platform/system independence of the CMS and to take advantage of as wide a range of computational hardware and software as possible.

1.2 The CMS as a System

It should be emphasized that the description that follows is for the fully-developed CMS that is envisioned by the design team. The rate at which the CMS evolves into this form will depend on the level of support and funding priorities.

This fully-developed CMS will be accessible from the user's workplace through a computer interface, either workstation- (Unix) or PC-based. The user's interface envisioned for this purpose, includes visualization software for displaying two- and three-dimensional representations of input and output data. It also includes geographical information system (GIS) capabilities for accessing and displaying data according to geographical location. Powerful statistical packages will be available for summarizing, synthesizing, and analyzing data, as well as extensive on-line help files to minimize (or avoid) reliance on operator's manuals. Not necessarily on-site, but accessible through this user's interface, are utilities and facilities, e.g., for producing video copies of animated visualizations of the data.

By pointing and clicking with a mouse, this envisioned interface will allow the user to select a model configuration (e.g., type of emissions model, type of meteorological model, type of air quality model) consistent with the intended application. If desired, the user can also select individual modules (e.g., preprocessors, chemical mechanism, boundary layer treatment, advection scheme) other than the default set. Having configured the model, the user then can select the geographic domain and time period. Then, the corresponding emissions, meteorology, air quality, topography, land use, and demographic data can be selected as needed, often from a geographically distributed archive via the Internet.

The CMS component models will be executed through the user's interface, as well as evaluated using on-line comparative and diagnostic tools. The user can generate tabular, graphical, and textual reports based on the data also via the interface. All of this can be done interactively with other collaborators through the interface's digital TV/ networking capability. Of course, such a capability also will support E-mail and easy file transfers.

Numerical models available from the CMS fall roughly into the categories of emissions, meteorological, dispersion, and air quality, and will be selected based on the application and data availability. They could be applied to assessments on local, urban, regional, or global scales of emergency response, population or ecosystem exposure, or response of visibility, ozone, deposition, and/or particulate matter to emissions control. In particular, special modules in the CMS will allow the system to assist during accidental releases of air pollutants and emergency situations. These modules will provide, among other things, real-time access to on-line meteorological data and satellite images.

Emissions models within the CMS will aggregate or disaggregate flat files of activity, land use, meteorological, biogenic, geogenic, point source, area, and mobile source

data into formats required for input to the air quality models. Diagnostic or prognostic meteorological models will be used to generate gridded wind, temperature, humidity, solar radiation, and cloud and precipitation fields, as needed, dealing with complex terrain as necessary. The coordinate and grid systems will be compatible with those of the air quality models. Choices for advanced dispersion modeling, such as Monte Carlo-Lagrangian particle methods, able to deal with buoyant, neutral, or dense plumes, will be available.

Air quality models will be formulated to allow fixed, adaptable, and/or nested grids to be used on urban, regional, and global scales. Provisions for simulating air-quality feedback on meteorology will be available. All models will be modularized to the extent feasible to provide maximum flexibility in configuring models, to facilitate diagnostic evaluation at the process level, and upgrading of their science. Coding standards will be established for all air quality modeling software to facilitate transfer of modeling technology among modeling groups and, in particular, adoption by the CMS. Coding standards will also be implemented to assure the highest possible degree of parallelization in the execution of the codes. Models approved for regulatory or compliance modeling will be accessible in a locked format.

Reflective of its comprehensive nature, the CMS envisioned here also will provide decision-analysis models to allow users to select optimal choices for responding to emergencies or implementing emission control strategies, based on the outputs from the dispersion and air quality models.

1.3 Summary of User's Requirements

CMS users will possess a wide variety of requirements. First, many of the users will be involved in some form of regulatory guidance and air quality management, including State Implementation Plan (SIP) preparation. Others will be involved in scientific experimentation. Another class of users will use the system for educational purposes. Many of the users will have limited skills in terms of computational systems expertise, air quality modeling (including emissions and meteorological modeling), policy analysis, etc., though some will be experts, having capabilities in individual areas beyond the CMS designers. Thus, the system will need to provide relatively straightforward access to the tools provided, yet be able to satisfy the experts' needs. In addition to facilitating the human interface to the CMS, issues such as data ownership and security must be addressed to ensure that the highest degree of scientific integrity and cost-effectiveness is achieved.

1.4 Summary of Strategy and Goals

As originally envisioned, the CMS would become the platform of choice for future air quality modeling, from a regulatory standpoint, as well as for scientific investigation. The CMS should provide all the desired functionality in a readily accessible package. The goal is thus to design a software system that will provide the necessary capabilities, including emissions modeling, meteorological modeling, air quality

modeling and characterization, decision support and air quality analysis (including visualization), and report preparation assistance. To support the CMS architecture, individual components will be specified, designed, and implemented where necessary. However, heavy reliance will be placed on the resources provided by universities, public domain, and commercial software developers, together with the software subsystems which accompany the computational platforms.

1.5 Evolution of the CMS Concepts

In this project, we developed initially a CP which provided our initial outline of the system (the Table of Contents of the CP is enclosed as Appendix A).

The CMS was described at its outset as a “three-legged stool” resting on the concepts of “All Users”, “All Models”, and “All Platforms”. The concept is outlined in Figure 1-1.

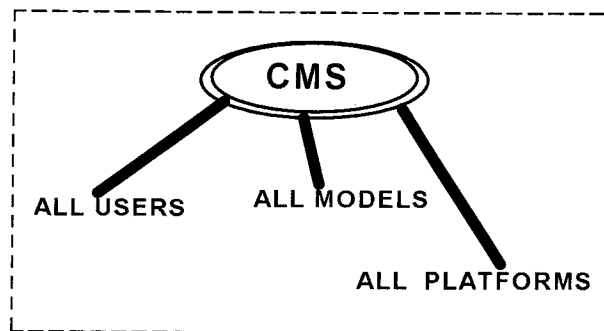


Figure 1-1
CMS as a “Three-Legged Stool”

The embodiment of the name “Comprehensive Modeling System” gives an accurate (albeit simplistic) picture of the vision for CMS established by the project founders as the “stake in the ground” for the future. The term “All Users”, of course, means that groups of users of the CMS range from the potentially unsophisticated user, who is charged with responsibility for monitoring and regulation, to the researchers in advanced computer modeling and analysis techniques. The term “All Models” was intended to describe the complete range of models used for the CMS tasks, and would include existing (or “legacy”) models, as well as new models created to exploit new science or advances in computer technology. “All Platforms” was meant to signify the inclusion of the lowly “personal computer” and the modern workstation, as well as the most advanced CRAY scale supercomputers, in the array of computing systems to be employed by the CMS.

This design document, we believe, reflects this vision, while recognizing the pragmatic constraints of budget, resources and time, and the need to deal with operational environments and the desire to conform to all relevant standards of computer system development. This document is addressed to three different communities: the CMS end-user; the CMS sponsors and potential sponsors; and the development team which will be commissioned to implement the system. Thus, the form of this document is

somewhat of a hybrid; partly a report, partly an exposition, and partly a design specification for the CMS.

In the CP, we anticipated the following outline for the Framework Design.

1. Administrative Infrastructure for the Development Phase,
including hierarchy of responsibilities and functions, regulatory oversight, technical oversight, contract management, people management, subcontractor management, internal communications, external public relations
2. Preparation of CMS Blueprint
a plan of action for the development of CMS
3. Implementation Plan
detailed development plan
4. Management Plan
administrative and technical/scientific

Based on today's knowledge, we believe in the following basic assumptions for developing the CMS.

- **Platform Independence.** The CMS will be developed using platform-independent tools, such as C++ Views. Therefore, the CMS will run transparently on different hardware/software systems (e.g., OS/2 and Unix, both with multitasking features). Of course, CMS performance will be different on different hardware/software systems, depending upon central processing unit (CPU) speed and CPU availability (if the system is shared with other users).
- **Remote computing.** In any platform, the CMS will allow users to perform remote computing, i.e., to delegate computational tasks to other machines (e.g., the running of CPU-intensive meteorological models). In any platform, user's requests will be examined for CPU time requirements and the user will know, in advance, the response time. The CMS central location should also be capable of providing users with CPU cycles, if requested.

An important component of any newly-developed computational tool will be the presence of the quality of "transparent computing". In short, this can be defined as "conducting all or portions of computational tasks on one or more

computing platforms without requiring the user to specify the exact sites or hardware/operating systems which will be used.”

There are several non-competing initiatives underway which are aimed at this objective, and which can provide the CMS with the basis for operation of its models beyond a single location and computer system. The Energy Sciences Network (ESnet), for example, is creating a program called Distributed Informatics, Computing, and Collaborative Environment (DICCE). The project involves all of the DOE national laboratories, as well as commercial developers such as Cray Research. The final paragraph of the current DICCE proposal describes many of the properties which are envisioned for the CMS and would make an excellent basis for exercising those CMS attributes.

“Imagine you are an end user working in the ESnet Community DICCE environment a year or two from now. From anywhere on the Internet, you only have to logon to the environment once. You now ‘see’ the file system for over twenty ESnet community sites. You can run your jobs on any system on which you are authenticated; in fact if some of the requested enhancements to DICCE are funded, you may run your job and not even know where it is physically running. You have direct access to numerous facilities on-line. There are whole sets of tools which appear to you as virtual laboratories or electronic places. You can remotely run experiments in a secure authenticated fashion. You can securely send e-mail including binary attachments.”

- **Central Location.** The CMS will have a “central” location – the CMS Center – where the official version(s) of the entire system is loaded on one computer system. Users will be able to download programs and files from this central location. The CMS Center will maintain the official version(s) of the CMS and all required files. This central location will be capable of providing users with remote computing cycles and data storage, when requested. The CMS Center will also serve as host location for visiting scientists on sabbatical from the home organizations to work on CMS system and science problems.
- **Minimum Computer Configuration for Access.** The two major pieces of the CMS are: the CMS Application (CMSA) which provides the interface into the system and the models/modules themselves. The CMSA will run on the users local machine, while the models/modules can be run either on the local or a remote machine across the Internet. In this section, we are referring to the system attributes required to run the CMSA. Computational requirements for the models are highly model-dependent.

The CMSA, in its ultimate configuration, may come in several flavors: the general purpose CMS system, as well as special purpose versions. The general purpose CMSA will require only the standard input/output (I/O)

devices expected on any computer: a screen, a keyboard, and a pointing device, like a mouse. Special purpose versions, such as an emergency response system, may employ other computer configurations, such as palmtop digital assistants, for logistical reasons. Such systems could run and display simple dispersion calculations and provide data input (and display) for more sophisticated models run on other machines.

In theory, there should not be any minimum hardware requirement, since the CMS central location should provide, when requested, CPU cycles and data storage. Therefore, we envision effective use of the CMS (in its final configuration) even with a laptop computer. Of course, to avoid irregular burdens on the CMS central location, most users should be encouraged to download the entire CMS system (or the modules that they need) into their computer platforms. Palmtop computers, or digital assistants, may play an important role for a special-purpose version of the CMS for emergency response, where people in the field (plant managers, fire officers, etc.) could execute quick simulations on portable computers through the CMS central location (or other computers).

- **Proprietary Issues.** The CMS will provide the software framework for performing comprehensive air quality modeling. The framework will be designed in such a way as to allow the use of either public domain or proprietary modules to perform the various tasks entailed. At least a base level of public domain capability will be provided for all the principle functionalities. Users, however, will have the option of upgrading to commercial software packages for improved performance and/or functionality (e.g., for visualization and GIS modules). Of course, the use of proprietary modules of the CMS will require the application of appropriate user fees.
- **CMS Development Strategy.** Initially, we thought that a top-down approach was the best. This would have required a precise and detailed design of the CMS system, followed by a software implementation effort, according to the design. After further investigation, we now believe that, while a general top-down framework design is still needed, the development of the CMS should be based upon a sequence of prototypes, each adding new functionalities. This approach (fast prototyping) has several advantages and a few disadvantages. The main disadvantage is that the final system may not be as well-structured as the system that could have been developed by a full top-down approach (this may probably be just a problem of form and elegance instead of substance; also, one must ask how many times, in the history of software development, a precise design – the best design – could really be made in advance, especially in the recent years when computer hardware and software have evolved at a dramatic speed). The other disadvantage is the risk that the initial prototypes could be useless. The main advantage, however, is to force developers, from the very beginning, to have a concrete approach and produce fast, useful deliverables. Also, the prototypes will be

distributed to the scientific community, thus allowing the developers to gather early feedback and constructive criticism. Last but not least, a sequence of evolving prototypes may be the only workable choice for a software development program with limited funding. In conclusion, the best strategy at this point is the definition of a clear (but flexible and not very detailed) framework design and the development of a series of prototypes evolving toward a clear goal. The development should move both bottom-up and top-down; in fact, each new prototype should allow rethinking and refining of the framework design, i.e., an adjustment of the final target. (Also, by focusing on the development of a sequence of prototypes, the developers will be encouraged to use available products/software instead of re-inventing the wheel. Prototypes will necessarily require the development of interfaces to utilize available software, instead of recoding. Again, we will sacrifice some elegance for cost-effectiveness.)

- **Parallel Efforts.** We are aware of a number of development efforts that are similar (or somehow related) to the CMS development by CAMRAQ (Peters et al., 1995; Novak et al., 1994). Especially in view of some expected limitations in the CMS development budget, it is mandatory to monitor these parallel developments and minimize duplication of effort. This requires some cost-effective compromise in the design of the system. For example, we may identify modules developed by other groups (e.g., a comprehensive emission modeling system or meteorological modeling system) which already possesses most of the needed requirements of the CMS. These modules may be implemented on a specific computer platform which is ideal for that particular module and the software be mostly platform-specific (e.g., strictly Unix- or PC-based) and, therefore, non-portable to the CMS, as we envision (i.e., platform independent). At this point, different strategies are possible. In particular, we may: 1) make the effort of porting and recoding the platform-dependent module into the platform-independent CMS, or 2) incorporate into the CMS a module which can only run on a specific platform. The second choice, though not elegant, can be extremely cost effective, especially for intermediate prototypes, and provide a workable solution which does not preclude any of the practical functionalities of the CMS system. In conclusion, to account for and benefit from parallel efforts, it is mandatory to maintain a flexible framework design and develop a CMS through a series of prototypes.

1.6 Framework Design

This report provides a Framework Design of the CMS. We define “framework”, in the present context, to include the following elements.

- The external organization and collection of resources necessary to support the access, use, maintenance, documentation, and evolution of the CMS.
- Design of the software systems and hardware comprising the CMS.

- A multi-year implementation plan for the development of the CMS.
- A management plan for the development program.

1.7 A Guide to this Report

At every juncture of a project which aspires to the range and vision of the CMS, it is imperative that a dispassionate review and discussion of all premises and basis should take place. This step is even more crucial as we finalize the design phase of the CMS program. Hence, in Chapter 1 we begin with a restatement of the fundamental aspects of the CMS on which the system architecture and design are based. Following this, is a discussion in Chapter 2 of the general CMS design and an overview of its operation. Chapter 3 discusses the CMS implementation plan, while Chapters 4 and 5 contain the detailed design specification which will be used as a basis for future evolution of the implementation of the CMS. Management plan and schedule/cost are provided in Chapters 6 and 7, respectively. Finally, Chapter 8 contains conclusions and recommendations.

Additional material in this report includes, at the beginning, an executive summary, a glossary, a list of abbreviations, and a preface which presents a summary of concepts and information that have been already presented in the CP. At the end, four appendices provide additional information (note, in particular, Appendix C which contains a description of some existing air pollution modeling prototypes and software products).

2

GENERAL DESIGN OF THE CMS

The design of the CMS proceeds from the definition of the user requirements, and then through the requirements for system efficiency, integrity, and security as prescribed by the best technologies available in the computer field. This chapter will present a brief overview of the CMS design and discuss the salient features of the system: 1) the requirements, 2) the system design, 3) the infrastructure, 4) a typical operation, and 5) the interfaces.

2.1 Requirements Analysis

To provide a foundation for the design of the CMS, the broad user requirements must be quantified in some manner. We have chosen to identify system requirements by three schema: 1) the user's desired view and perceptions of the system; 2) the demands on system capabilities and resources for representative categories of users; and, 3) a set of probable scenarios characteristic of different usage of the system.

2.1.1 *The User Perspective*

The CMS user sees the system through a series of interactions with the facilities provided by what should essentially appear to be a "black box" containing all the required functions. The primary interface will be through a set of standard graphical user interfaces (GUI) which promote the effective use of the system by even uninitiated users.

One user view of the "CMS black box" is displayed in Figure 2-1. The users may be aware of a hardware-like "backplane" of communications and software "glue", but their focus is on the functions offered by the CMS through the GUI. The users may be aware of the demands their work may make on system resources, but the CMS will reflect the availability and cost of those resources to the user in the course of CMS usage.

2.1.2 *System Demands*

Different system resources are required for each type of use. The CMS design must encompass the foreseeable range of resource requirements as outlined in Table 2-1, in

* In the electronics industry a "backplane" provides the interconnection of all active electronic components. This used to be a mat of wires and is now commonly a printed circuit board, like the "motherboard" in a PC.

which "interface detail" refers to how much of the detail of making a model run is presented to the user in the interface. The "Must be Local" data volume refers to storage directly needed to make the run, such as model input/output. The "Could be Remote" data volume refers to data that typically need to be extracted from very large datasets residing on the Internet such as the USGS topography data. CPU performance is an admittedly vague characterization of required horsepower to get the job done in a "reasonable" time. In rough terms, "low" might correspond to a Pentium 90, "medium" to a mid-range workstation such as an Indigo^2, and "high" to high-end workstations or "supercomputers", such as the SGI Challenge system.

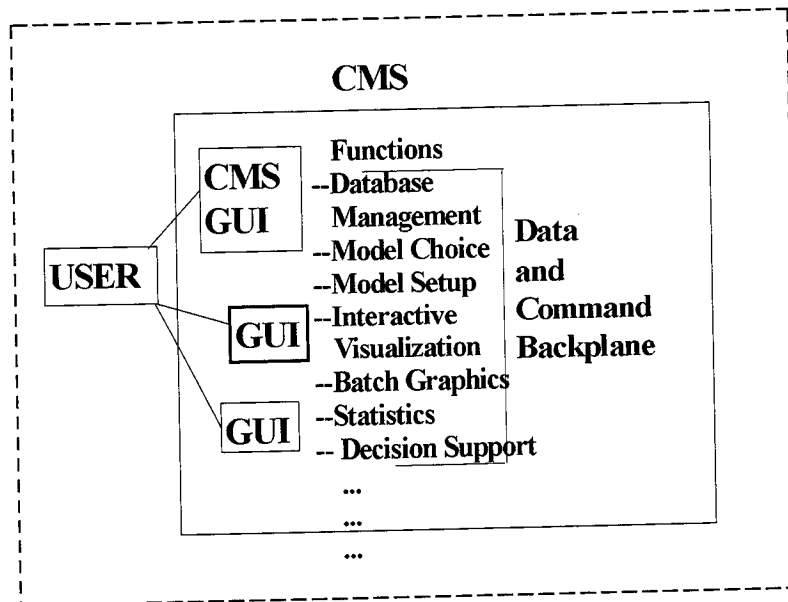


Figure 2-1
A User Perspective of the CMS

Table 2-1
Resource Requirements

	Interface Detail	(Must be local) Data Volume	(Could be remote) Data Volume	RAM Required	CPU Perf	Response Time
Regulatory Photochemical	med	1 - 10 GB	10s to 100s of MB	64-1024 MB	med - high	hours - days
Regulatory Non-Reactive	low - med	1 - 10 GB	10s of MB	16-256 MB	low - high	hours - days
Scientific Studies	med - high	1 - 100 GB	10s to 10,000s of MB	64-1024 MB	med - high	hours - weeks
Emergency Management	low	1 -10 GB	10s of MB	64-512 MB	low-med	minutes - hours
Model Developers	high	1 - 100 GB	10s to 10,000s of MB	64-1024 MB	med - high	hours - days

2.1.3 Scenarios

The system resource requirements must also be viewed in the context of how the system will be used in the most probable ways. The following scenarios describe the cases which have been identified as design points for the CMS.

- baseline simulation
- sensitivity/uncertainty
- new science
- control strategy
- permitting
- exposure/risk assessment/cost-benefit analysis
- emergency preparedness and response

These seven scenarios, which are further discussed below, show examples of how the CMS will be used. There are, of course, other applications in which air quality, emissions, and meteorological models are currently used. The CMS could also take on those duties. Also, we fully expect that more uses will be identified. For one, long-term evolution of the CMS is expected to be towards multimedia pollution impacts, i.e., the simulation of transport and fate of pollutants in air, water, soil, groundwater, and biota.

2.1.3.1 Baseline simulation. The initial use of the CMS in many applications, especially those related to photochemical simulations, will likely be for the development of base case air quality model simulations. In this scenario, the CMS user will apply the CMS to develop emissions, meteorological, and other air quality model inputs. These air quality model inputs will reflect suspected conditions in a specific regional/urban model domain (e.g., regional: Northeast United States, the Midwest, the San Joaquin Valley; urban: Los Angeles, CA, Atlanta, GA, Phoenix, AZ) for a specific model episode (e.g., August 3 through 6, 1991). The various components of the CMS will be exercised, and the results of each CMS execution will be compared against a set of observed fields available on-line. If the comparison of the CMS results against observed fields falls within the predefined acceptance criteria, the baseline scenario will be considered established, and the CMS baseline model results will be considered as a candidate for use in the other CMS scenarios.

2.1.3.2 Sensitivity/uncertainty. The purpose of a sensitivity study is to determine the influence of input parameters that affect the emissions model, the meteorological model, and the air quality model (AQM). The basic notion is that of the hundreds of inputs (there are actually thousands depending on how they are counted) to the CMS, only a small subset may make a significant difference in the actual model predictions (i.e., the other inputs get lost in the “noise” of the major factors). The purpose of a sensitivity study is to identify those major components that most affect the results of CMS execution and quantify the model response to those inputs.

The purpose of an uncertainty study is similar, except that the concern is how uncertainties in the inputs affect the outputs – again, if the major (or most sensitive)

components have an uncertainty of +/- 5% while some minor components have an uncertainty of +/- 50%, then the predictions still have a low uncertainty. However, if the major components have uncertainties of +/- 50% or 100%, then the predictions will also be significantly uncertain. There are several different strategies to determine sensitivity/uncertainty and several groups are looking at the problem. The type of study that we have been considering so far would be the "brute force" approach. In such a study, the 100 different inputs to be varied would be identified. The CMS is then run with each of these inputs varied in such a way as to assess which ones are the major components of the predictions. The major concern from an analysis point of view, is setting up the changes to the inputs (on the "front end") and then comparing the differences in the predictions (on the "back end"). More powerful methods are in development.

Several recent studies on sensitivity/uncertainty analysis should be mentioned as potential candidates for providing routines and methodologies for the CMS. These studies include: Horwedel et al. (1992), who discuss an automated sensitivity analysis of an atmospheric dispersion model; Hwang and Byun (1995), who apply automatic differentiation for studying the sensitivity of numerical advection schemes in air quality models; Jeffries and Tonnesen (1994), who developed a process analysis method in which a complete mass balance is used to quantify characteristic values of the air quality reactive system; Rao et al. (1996), who demonstrate that observed ozone time series are comprised of deterministic and stochastic components, with stochastic variations that cannot be controlled; and Isukapalli and Georgopoulos (1995), who apply stochastic response surface methods to uncertainty analysis of photochemical models.

2.1.3.3 New science. The purpose of this type of study is to examine how improving (or modifying) the science embodied in the CMS impacts the results. Recall that the AQM (or meteorological or emissions model) is a simplification of the processes that go on in the atmosphere or at the surface. As our knowledge of how the atmosphere works is refined, the CMS must be updated so that it continues to embody the "best science". There are several classes of changes that might be made to the CMS. The simplest would involve simply changing constants embodied in the program, e.g., the rate constants for the equations in the chemistry module. More relevant changes might require modification of the way a certain calculation is performed. In some cases, a change would be the addition of some new module – for instance, cloud chemistry, or aqueous-phase calculations which would sit "on top" of the existing code.

The major issues in this type of study are how to allow the modeler to change the source code of the CMS and how to validate the predictions (model performance evaluation) of the changed model. There are certain types of changes and additions that the CMS has already been designed to handle and these changes would be easy to make in a properly modularized CMS code. On the other hand, certain situations will arise where these changes are not "simple". There must be support within the CMS for changing the internals of an AQM and for doing this in a "convenient" way (i.e., not on the FORTRAN code level). There must also be support for validating the results of a

modified AQM with some “baseline” scenario to ensure that the AQM still “performs” (in terms of making valid predictions) properly overall.

2.1.3.4 Control strategy. The purpose of the control strategy study is to quantify how some change in the predicted levels of a certain pollutant respond to emissions changes. There are several different ways of phrasing goals for such a study – for instance, by lowering the peak daily levels, lowering annual exposure, and by performing a multi-objective analysis.

The system must allow the user to modify the inputs to the CMS in such a way that the effect of controls can be emulated. For instance, the modeler might want to “eliminate” certain point sources from the domain or perhaps reduce emissions of a certain source by use of some control technology. The important aspect of this type of study is to provide the user the hooks to change the inputs and then a way to compare the outputs against some “baseline” version. This type of study may well require many runs of an AQM, but will be different from the sensitivity/uncertainty case in that the runs will be more “interactive”. In the sensitivity study, it will usually be possible to specify 10s or 100s of runs at one time, whereas the control strategy study will consist of “What if?” scenarios performed interactively.

2.1.3.5 Permitting. We envision a CMS which includes those models regularly used for permitting of facilities. These models are widely used and are currently supported by a number of consulting firms. However, they can be advantageously linked into a CMS. For one, the CMS meteorological models can be used to provide better wind field information and the model components can be used to provide the corresponding data as well. These added capabilities would greatly ease the application of such models. Further, the visualization and decision support tools in the CMS would be of direct benefit as well. In fact, while much of the description of the “air quality modeling” component of the CMS may seem geared toward the more advanced models (e.g., regional photochemical models), that is in part because they are more demanding, thus, setting the requirements for the CMS structure. The simpler models are envisioned to be part of the suite of air quality models provided.

In the permitting scenario, the applicable model would be chosen from a suite of models presented to the user, along with the guidelines for appropriate use (including its status in terms of approval for permitting). The user then would link it to the appropriate inputs and conduct a set of calculations to find the impact of the emissions on local air quality. This information would then be used by the decision support tools to conduct a risk assessment.

2.1.3.6 Exposure/risk assessment/cost-benefit analysis. Exposure, risk assessment, and cost-benefit analysis is taking a more prominent position in regulatory decision making. Such calculations are conducted using a variety of models, including photochemical and toxics. In such cases, the AQMs provide an estimate of how sources impact air quality. Then, other components of the CMS use such information,

along with information on population distributions, personal activity, toxicity, control costs, etc., to provide the type of information used by decision makers and planners.

2.1.3.7 Emergency preparedness and response. Emergency preparedness is essentially a decision support usage of CMS, where meteorological and dispersion modeling are used to understand potential public health impacts of unplanned releases of toxic substances into the atmosphere. Emergency preparedness implies the use of models to understand potential effects from releases that have not occurred. Title 3 of the 1990 Clean Air Act requires all sites containing hazardous substances over a threshold limit to prepare a Risk Management Plan (RMP) looking at impacts from a “worst case scenario” as well as several more likely scenarios. “Worst Case” applies to both the release characteristics and the meteorological conditions. The EPA wording acknowledges the need to go beyond straight-line Gaussian type modeling in cases involving complex terrain or unique meteorological conditions.

One portion of the RMP is a plan to respond to emergencies in a real-time mode. This places a tremendous constraint on the run time of the model, if guidance is to be obtained in a useful timeframe. Compromises in model sophistication and resolution must be made. In many cases several models can be deployed, covering different time and space scales (or, better, models with adaptive grid size and time increments can be used). The time constraint also imposes limits on how much detail the user can be expected to deal with in getting the model started and interpreting the model output.

2.2 System Design

The CMS must deal with the user’s requirements identified in Section 2.1, on the one hand, and the realities of existing and future hardware and software systems on the other. The top level system design is illustrated in Figure 2-2. Note that we define the CMS Application (CMSA) as that part of the program that must run on the local machine (i.e., the GUI/CMS interface in Figures 2-2 and 2-3).

The user interacts with the CMS through a graphical user interface (GUI), though the type of interface may expand to include other input/output modalities. Members of the CMS community, at any one time, may be accessing data and services (e.g., models, visualization, etc.) to complete a specific task similar to one of the scenarios described in Section 2.1. Connectivity between the user and the CMS services is provided by the CMS manager, which is part of the CMS system and infrastructure. Figure 2-3 shows the CMS design in more detail and from a different angle.

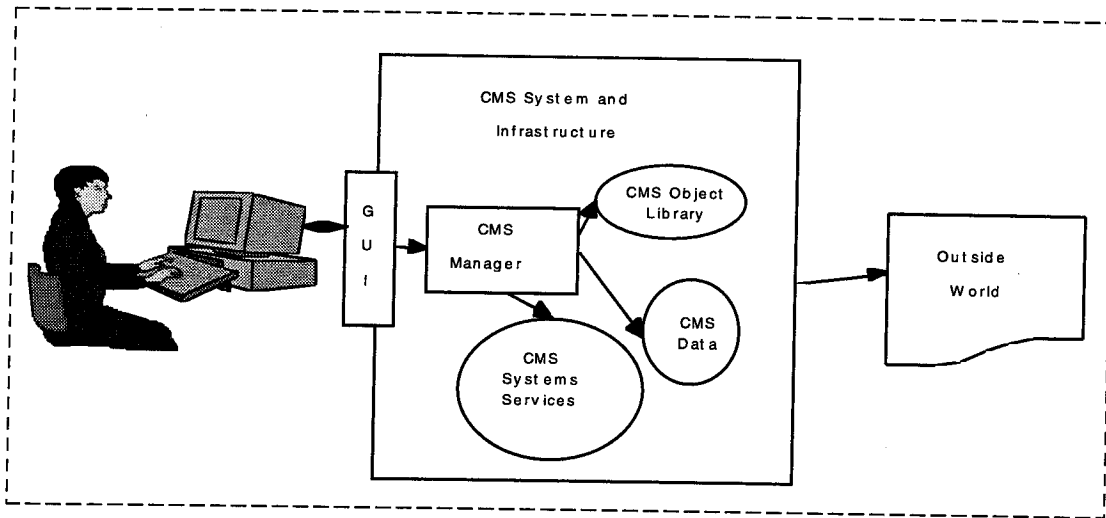


Figure 2-2
The CMS Top Level System Design

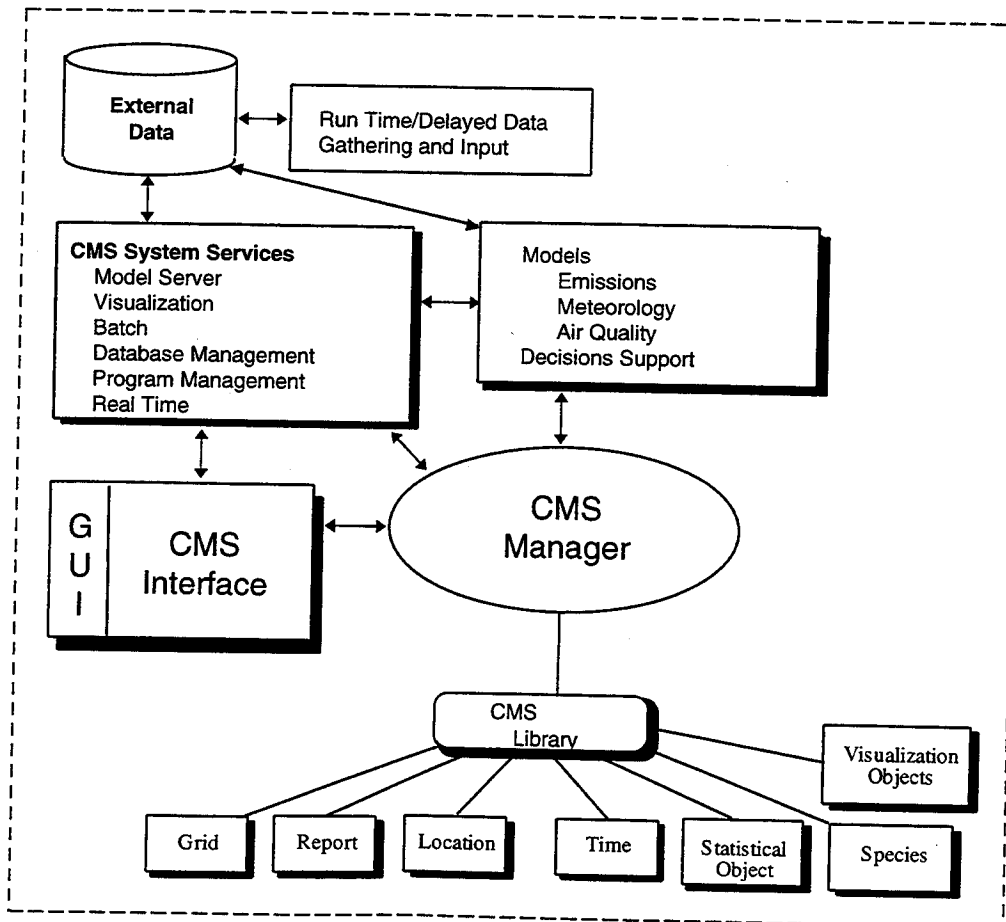


Figure 2-3
System Terminology

The design of the CMS itself has many facets, the user perspective, the system perspective, and the implementation and management perspectives to be discussed in subsequent sections of this document. This design provides the key attributes to make the CMS effective and to ensure its portable mapping onto the widest range of hardware and operating systems platforms possible. We provide in the subsections below, some preliminary discussion of a few key elements (the user interface, the CMS Manager, and a few system services) illustrated in Figures 2-2 and 2-3.

2.2.1 The User Interface

The CMS user interface must provide human interaction to persons with varied levels of expertise in their own fields while possessing, in some cases, small literacy in the vernacular of the computer industry. One of the principle requirements discussed previously is the concept of “transparency” wherein the user deals with the CMS in exactly the same way regardless of the location and type of databases and computer systems being employed at the time. An important system requirement is that the user interface, as all other components of the CMS, be “platform neutral” or “platform independent”.

There are several readily available software systems which support the creation and operation of a GUI to provide the user interaction. Specifications for this system component may be found in Section 3.5.1.1 of this document. The user will see an identical interface on every terminal that they will use to access the system, whether it is a PC running Windows95, Windows NT, MacOS, or OS/2, or whether it is a Unix workstation running a manufacturer’s own implementation of Motif.

This “GUI” will present choices, accept and interpret user inputs, draw the operator through a series of steps to initialize, execute models, and supporting software, store and retrieve data, and analyze and publish the results. Figure 2-4 contains an example of a sequence of screen displays which are the consequences of a user starting up the CMS (upper left screen display), selecting the photochemical modeling component (middle screen display), and choosing the Los Angeles Basin and its database as the object of this particular run (lower right screen display).

2.2.2 CMS Manager

The CMS management services are used to control startup, shutdown, error handling, and administration of the CMS. This facility maintains oversight of all clients and services and handles adding and removing facilities, as well as restarting clients, services, and models, where appropriate.

2.2.3 CMS System Services

The CMS System Services contain the real heart of the CMS. The system services operate directly on the data. In this regard, the internal design of the CMS could be termed “data centered”. The data in this case contains not only the obvious cases of

model input and output data, but the programs themselves, and auxiliary data such as GIS and other graphical mapping information. Thus, an important component of the CMS will be a comprehensive database management system which is capable of dealing with a variety of data formats, providing conversions where necessary. We discuss in more detail below the database management system, the program management, and the functional services.

2.2.3.1 Database management system. The database system characteristics are specified in detail in Section 5.3, but it is important to highlight here that this system must be able to provide internal security and integrity checks, even when the databases or portions thereof, are geographically scattered and accessible only through relatively unsecured network connections.

Since one of our principal goals is for the CMS to be “platform independent” it is essential that this program adopt a standard database system which is demonstrably “platform neutral”. Although a specific system has not been chosen, we have the good fortune to be initiating this program at a time when several excellent subsystems are coming to maturity from commercial as well as academic researchers. At this point, the most important aspect of the design is not the brand name of the database system but the completeness of its specifications which will need further review.

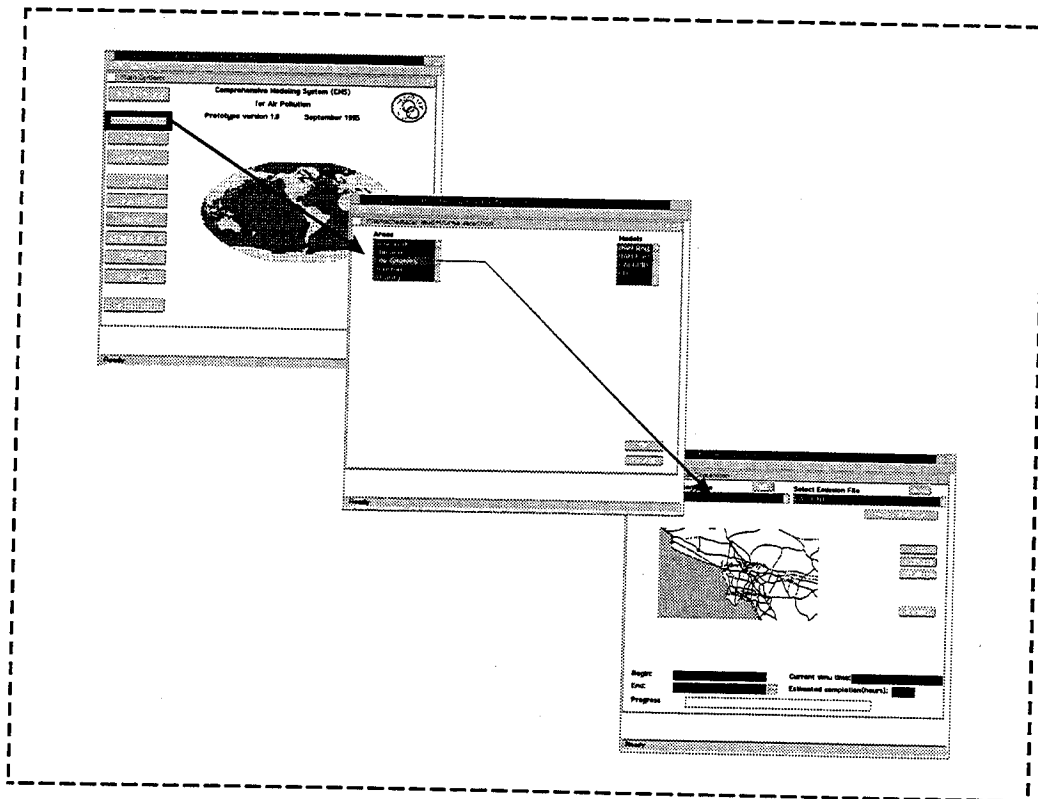


Figure 2-4
Graphical User Interface – Example

2.2.3.2 Program management. It is very likely that a general purpose database management system, which is optimized for the efficient retrieval and dissemination of large, complex datasets, will not supply the mechanisms to manage and control the actual program suites which comprises the computational core of the CMS. One goal of this CMS design is to bring into a common context all of the programs, modules, subprograms, and libraries which will appear in the system. A modern method for conceiving of and managing such a myriad of components is known as "object oriented".

When used as a design process, it is called "object-oriented design" (OOD) and almost every phase of system development and operation can have the letters "OO" pre-appended these days. Despite some misplaced hyperbole about "OO" systems in the past, the technology is now in hand and the training in its application is widespread. Thus, the CMS will utilize an object-oriented program management system which is described in Appendix B, and for which again there are several excellent candidates which exhibit not only the desirable "OO" properties but are themselves fully "platform neutral". The common object resource broker architecture (CORBA) is an example and is described in Section 5.6. A further description of the goals and processes of "OO" development may be found in Appendix B.

In the CMS, many existing meteorological, emissions and dispersion programs will be incorporated by providing a "CMS Wrapper" which consists of additional code and data structures to give them the appearance of self-contained objects. This concept of using extra code to "wrap" around "legacy" models to provide the appearance of common objects is discussed further in Section 5.7.

2.2.3.3 Functional services. In addition to the database and program management services, the CMS provides facilities for managing each of the major functions which comprises the system: meteorological service, emissions service, air quality modeling service. For example, the emissions service possesses knowledge of the various information requirements of the emissions models imbedded in the CMS and can generate the requisite database depending upon the user inputs. A detailed description of these services may be found in Section 5.5. Functional services also include the visualization service, the batch service, and the real-time service.

2.3 The CMS "Infrastructure"

The CMS infrastructure embodies the system as a whole. To illustrate the task which the CMS infrastructure must perform, the following three figures demonstrate how the CMS may actually be employed:

Figure 2-5 depicts the simplest "instantiation" of the CMS. In this case all of the CMS components reside on a single machine with all programs and data resident on a local

* "Instantiation" is the creation, or starting up, of a new copy (or instance) of a program, object, or process in a computing system.

disk. As far as security is concerned it is presumed that if a user can sit at the terminal and login, they have access to all the components for which their login has permission. This example also presumes that there is sufficient computer power and data storage to fully accomplish the tasks presented to it, perhaps one of the scenarios given in Section 2.1.3. The user perceives (and rightly so) that all of the resources are close at hand and under the control of the user and the machine with which he is interacting. In this case, there is minimal requirement for the functions we have designated the “CMS infrastructure”.

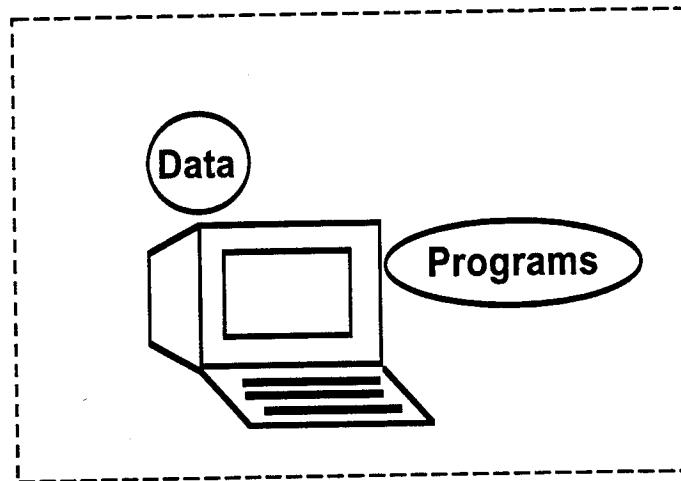


Figure 2-5
Self-Contained System

Figure 2-6 displays a more complicated system arrangement than the previous example. The terminal at which the user is working contains neither the data nor the programs which are required for the user's tasks. The function of the terminal is to provide the GUI functions. The remainder of the programs and functions provided by the services layer reside on a computer remote from the user. This implies that there is some sort of communications network between the two systems. To make this example more complex, we assume that the network is a common carrier or Internet system so that the security of information transmitted must be ensured by the user or user-directed processes at each end of the communications.

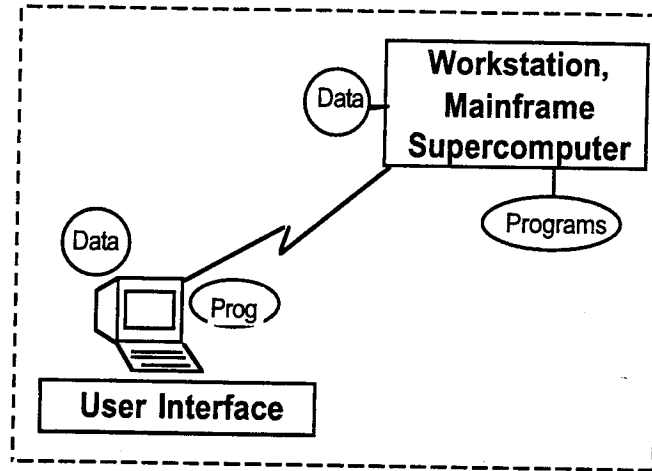


Figure 2-6
Remote Model Execution

Finally, in Figure 2-7 we have an apparently confusing, but quite realistic, use of the CMS. Since it is neither sensible nor desirable to transport all databases to a single computer site to be used in computations, the figure gives a very probable representation of the CMS topology when employed on large, complex tasks. A key aspect of the CMS design is, as we stated previously, the presence of a very robust database management system. Such a system attempts to provide as many data resources as possible, but it must perform all of its functions when the data resources are not all collocated with each other or even with a major computing resource. As in our previous example, we will assume that all of the network connections are insecure in themselves, the most common linkage being the ubiquitous Internet system.

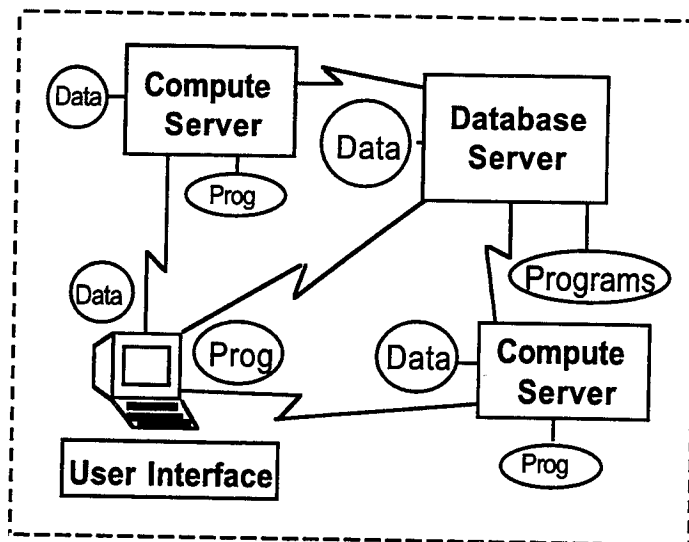


Figure 2-7
Multiple Platform Execution

Now consider Figures 2-5, 2-6, and 2-7 together with the requirements for “platform independence” and “transparency”. The platform independence herein means that all of the functions could be performed on almost any brand of hardware and operating system. This implies also that functions might move between platforms as network topologies evolve and the economics of new hardware and software and user requirements become manifest.

The transparency element is demonstrated dramatically here as the system strives to provide the identical interfaces, actions, and system reactions to the user in all three cases. Except for having to deal with, and acknowledge system prompts, relating to the different ‘costs’ associated with performing functions on more expensive machines and over higher rate communications systems, the user will see no other difference between operating with the schema shown in Figure 2-5 and those of Figure 2-6 or 2-7. We envision this transparency feature as an essential pre-requisite of the CMS, to assure its general use among the air pollution community.

As part of the CMS infrastructure, we present below some additional information about the management of processes/communications and file systems/security/integrity.

2.3.1 Management of Processes and Communications

If the CMS was only to exist on a single computing system and be accessed by a single user at a time it would be possible to implement most of the system as a collection of programs and subroutines. The previous discussions make clear that a far more robust system is required. Hence, the CMS is divided up into entities which contain one or more programs. Many of these entities are semi-autonomous and may be executed on one or more computer systems. They can communicate with other entities and perhaps even invoke services on other computing nodes.

The relationship between a multiplicity of asynchronous processes can create chaos in even moderately complex systems. To avoid this chaos, a set of formalisms have been developed over time to describe and bound the creation, operation, and interaction of such entities. Fortunately, the CMS is far from unique in many of its requirements. Systems as varied as on-line airline reservations to banking systems share many of the same needs. In response to this, a significant effort in this area has resulted in the Open Software Specification of the distributed computing environment (DCE) which is now provided by every vendor of computing hardware and operating systems, including Unix systems as well as the PC-based Windows 95, Windows NT, MacOS, and OS/2. Thus, the underlying “backplane” of the CMS infrastructure is the DCE, providing a platform independent system and schema for managing the entire CMS.

Figure 2-8 provides a simple sketch of the basics of the DCE. A much more detailed description of the DCE and the DCE-based mechanisms employed in the CMS are covered in Section 5.8. However, the reader will encounter several terms, throughout the remainder of this chapter, which are simple enough to describe here. First, as

Figure 2-8 shows, the basic DCE provides for accurate time synchronization, system-wide directory services, and elective security among all CPUs in the system. In addition, a simple mechanism for invoking or requesting services of another entity is through a remote procedure call (RPC). A RPC could be viewed simply as an extension of the use of subroutines and functions in Fortran and "C" programming. For example :

```
PROGRAM MAIN
CALL COMPUTE_NORMAL(X, Y, Z, NORMAL)
.
.
END
SUBROUTINE COMPUTE_NORMAL(X, Y, Z, NORMAL)
.
.
.
END
```

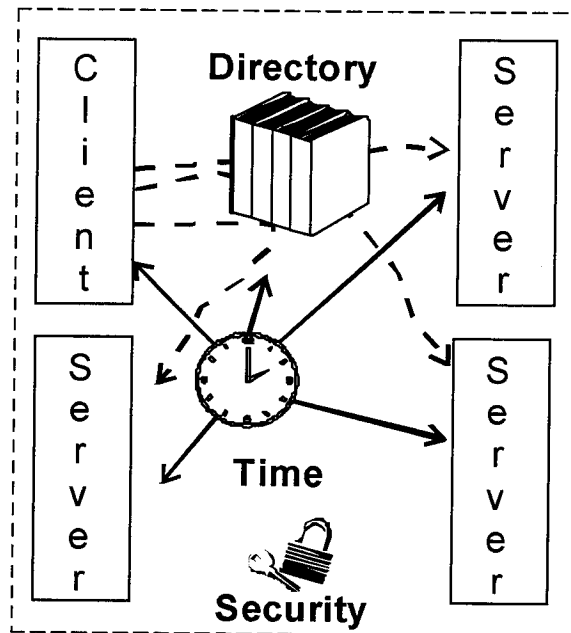


Figure 2-8
The Distributed Computing Environment (DCE)

In the 'traditional' model, this main and this subprogram would be compiled together and the execution of both would be within the same user process on the same machine. A common variant would be where the subroutine "**compute_normal**" would have been previously compiled and perhaps placed in a "library" which is later linked to the main program. The subroutine would still be executed as part of the same user process as the main program and on the same machine.

If it is desired to have the subroutine executed in a separate process from the user-process, as when there is a need for security or shared code and data, the subroutine

call can be converted by the compiler or the system to a RPC. This change can, in many cases, be made invisible to the user and the source code shown above would remain unchanged. Once a function can be invoked through a RPC, it is a simple step to permit execution on a platform different from the one on which the main program is executing.

The concept of RPC has been around for many years and is a fundamental property of all Unix systems. The DCE has carried the concept further and has created a more formal structure to the calling process which makes it possible to perform rigorous authentication and management of the intercommunications. Particular attention has been paid to the efficiency of these RPC implementations such that use of an RPC instead of a normal subroutine interface will normally not impose unacceptable execution 'overhead'. The concepts of system "clients" and "servers" has also been codified and specifications as to what type of functions each entity performs are part of the DCE standard. In the simplest terms, a "client" is an element which requests one or more services from a "server". The CMS design uses these formalisms as an underpinning to similar specifications for CMS clients and servers. For example, the database subsystem specified in Section 5.3 will operate as a "server"; however, there are occasions when it must request a function from another CMS "server".

2.3.2 File Systems, Security, and Integrity

Underlying the database management system will also be a standard mechanism for securing the integrity and access to the data files within the CMS. The DCE provides a basic distributed file system (DFS), which can be augmented with Kerberos security mechanisms. Again, these systems are available on all computing platforms and have become industry standards, so that "platform neutrality" of all of the base components of the CMS is assured.

2.4 Typical Operation

As an example, to describe a typical operation of the CMS, we will illustrate the general sequence of tasks in running a meteorological model to produce meteorological conditions fields for use in an air quality model. We will then follow through on what activities are required in the CMS to accomplish the tasks.

2.4.1 The User Actions

This run-through of user actions required to accomplish the meteorological simulation are not meant to imply that the details of the meteorological modeling GUI have been fully determined, but rather to serve as an illustration of the types of interactions expected in the CMS. A diagram of the major processes involved is also shown with a brief description of the actions, shown as subparagraphs 'a' through 'u'.

- **Login ..** To a CMS Client node and startup the initial GUI display.

In Figure 2-9, the following actions are performed.

- a. The client establishes its identity with the security system.
- b. The security system notifies the CMS that the client has permission to access the resources.
- c. The client requests linkage to the CMS management server.
- d. The client registers with the CMS management server which determines what services will be available to this client and session. A list of eligible CMS functions is delivered to the client.

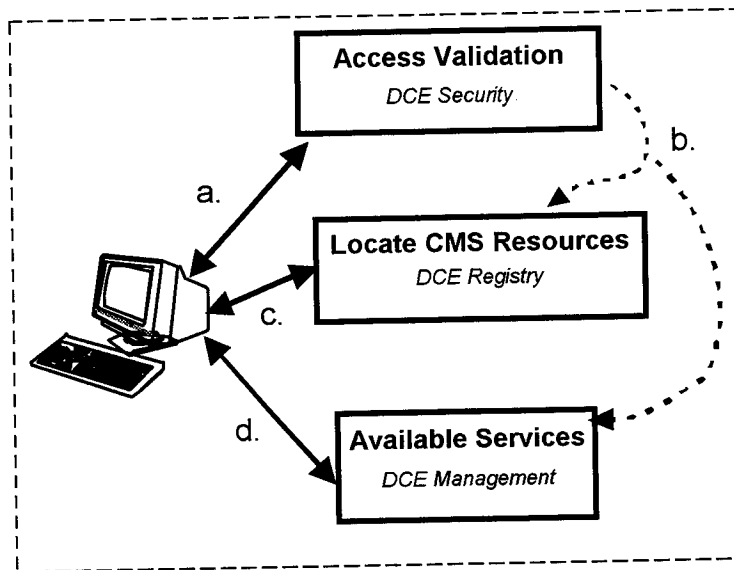


Figure 2-9
Login to CMS

- **Select ..** “Run Met Simulation” from the GUI menus. The system will display a group of supported models from which to choose.

In Figure 2-10, the following actions are performed:

- The client requests the meteorological services from the CMS manager which instantiates the module and gives permission for access by the client (j).
- The CMS meteorology server provides the client with a list of available models which is extracted from the models database upon a request to the CMS programs/objects manager (i). The client communicates the model selection (which the CMS manager then also registers).
- g & h. The model object ‘driver’ is started by the CMS meteorology server and this interface and the server provides the client with modeling requirements information.

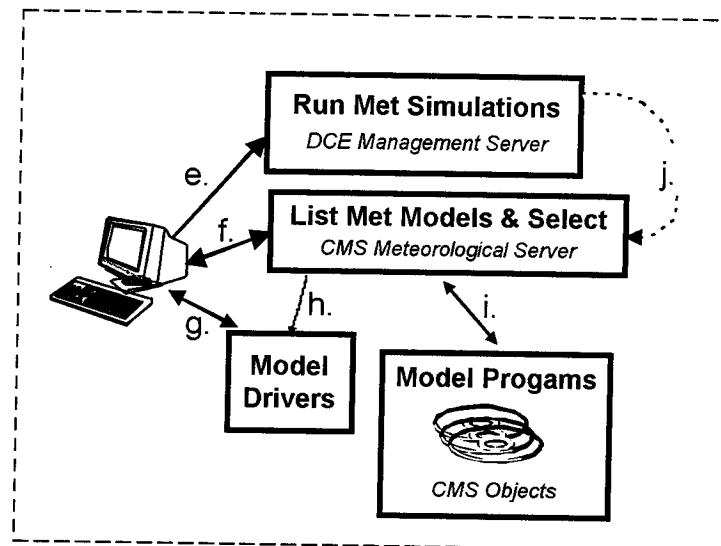


Figure 2-10
Select Model

- **Identify ..** The model is identified by double-clicking on its name in the menu. The system then displays a large scale map from which to select the modeling domain. This map will display coverage areas of local and remote datasets useable to initialize the model. On-line help for the model setup, execution, and analysis is available by clicking on appropriate menu items.
- **Establish ..** The model domain by dragging a 'rubberband box' across the map.
- **Choose ..** The horizontal and vertical structure of the grid(s).
- **Display ..** This action displays a list of all physical surface characteristics required by the chosen model. Click on "Acquire Surface Characteristics Data" button.
- **List ..** This action lists Resources that are available locally and over the Internet, and the characteristics of the datasets; i.e., resolution, accuracy, etc., by clicking on the desired surface data item.
- **Determine ..** This action allows the user to determine one of the alternative sources by double clicking on it. The system then goes out and transfers the data required for the chosen grid, and under user guidance, samples/averages/ filters the data onto the specified grid(s).
- **Repeat ..** This action is a repetition for each of the surface characterization fields. The user clicks "Done" in the "Acquire Surface Characteristics Data" window when done. At each stage the user can access the data in the CMS visualization system to interactively view/edit the model grid values. This will also be true for all data entered, including meteorological data and others.

- **Pick ..** This action picks the source data for the model. The user clicks on the “Acquire Meteorological Data” button. The system displays a list of all required and related datasets available locally or on the Internet. This includes auxiliary data such as satellite imagery.

In Figure 2-11, the following actions are preformed:

- k. The client requests map data with increasing levels of refinement. Surface data is then requested for the region selected by the user from the maps provided. The type and format of surface data needed is provided by the model “driver”.
- l & m. The client selects the model data to be used based on the requirements of the particular model. Note that the data is not sent to the client but information as to the source files is provided to both the client and the model driver (n & o).

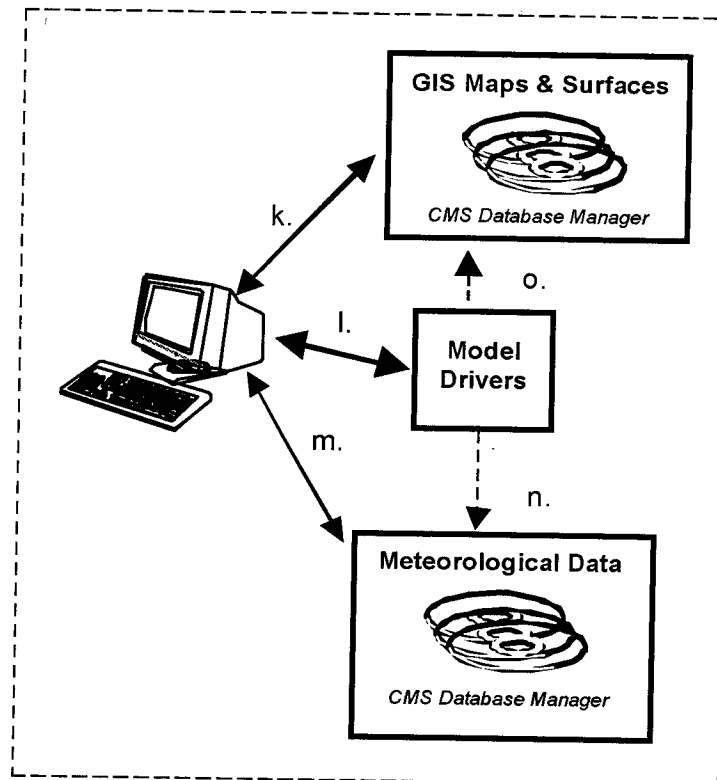


Figure 2-11
Setup Model

- **Setup ..** The user sets up the run by clicking on the “Model Setup Options” button. The system prompts the user to enter the level of detail they wish to be presented with. The system then brings up an additional window or windows for choosing model parameters.
- **Decide ..** The user decides on the operational parameters for the run. Once the model setup is complete, the system informs the user of time and cost estimates for making the run (and in some cases transferring the data) on the local system and any remote systems to which the user has access.
- **Start ..** The user starts the model execution. During the run the user may analyze/visualize the model output up to the latest frame. The model may also have facilities for corrective “steering” as it runs by tweaking parameters on the fly. The CMS must be able to support this both for local and remote systems.

- **Store ..** The user stores and disseminates the result data. After the run is complete, the user has the option to “publish” the run to other CMS users. That is, to make the input, output, and parameter selection available to other CMS users who may be looking for an appropriate met simulation to test their latest AQM on.

In Figure 2-12, the following actions are performed:

- p. The client “negotiates” with the CMS manager to determine how the model run is to be made. The costs and efficiencies involved in the site for the computations, determinations as to whether data should be moved to that site or “mounted” across the networks, will influence the decisions regarding the model execution.
- q, r, & s. The CMS manager performs any necessary data movement and scheduling of the actual model run on the selected computational resource.
- t & u. The client interacts with the model and the model database during model execution (if desired) and makes dispositions of the results data when necessary.

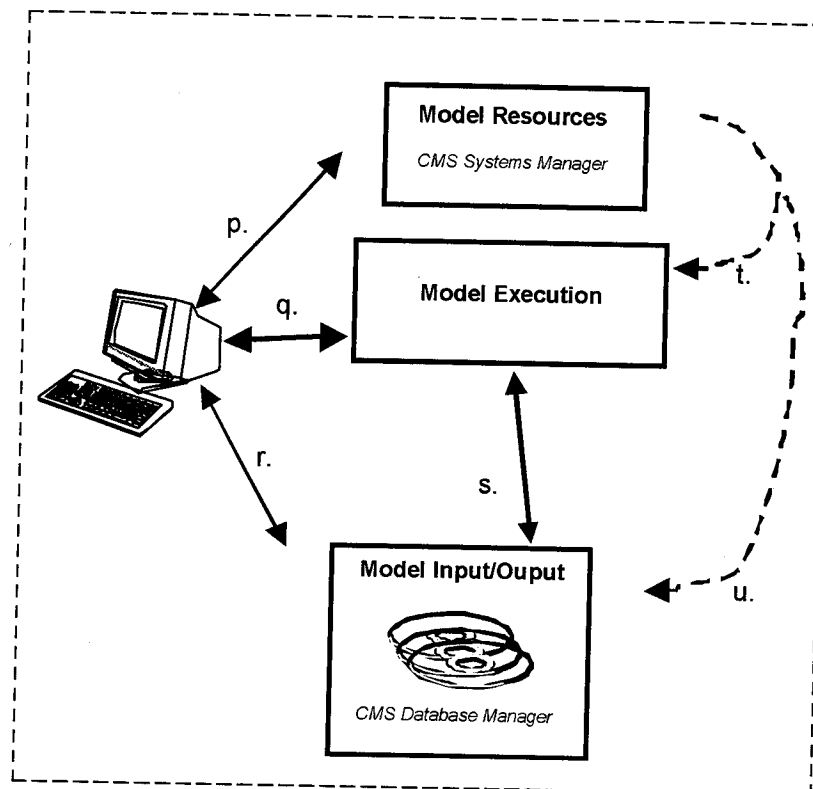


Figure 2-12
Model Execution

2.4.2 System Services and Actions

The major system activities which take place for this example and which are shown in previous figures and annotations could take place on a single CPU or across a network of CPU's. To manage such transparency; however, some care is required in the subdivision of the various functions in the CMS. Therein lies the major challenge of the CMS design, the identification of separable entities, which we have called "clients and servers". The rationale for the subdivision and constituents of the CMS is described in Chapter 5.

2.5 CMS Interfaces

The CMS interfaces consist of the low-level software systems which provide most of the interfaces between the CMS services and the native operating system on a given platform. As stated previously, these interfaces are comprised of standard, "platform neutral" elements which derive from the open system foundation's developments for the distribution of computation and file system resources as well as security across a (potentially) heterogeneous, stand-alone, network-connected machines. A sketch of this system appeared in Section 2.3, but as the reader might expect, the infrastructure itself is a series of layers, any one of which may interface to the CMS services or the user application. These relationships will be discussed in Chapter 4.

3

IMPLEMENTATION PLAN

The premise for our implementation approach is that it is not practical, and arguably not possible, to specify ahead of time every implementation detail and anticipate the consequences of every interaction between system components for a system such as the CMS. Recently there has been increasing work in the field of understanding complex systems. Nobel prize winning high-energy physicist Murray Gell-Mann, among others, have been looking at questions such as defining complexity and understanding how complex systems develop (Gell-Mann, 1995). The key to the successful development of complexity (such as a biological entity or a child learning to speak) lies in a concept he calls the "complex adaptive system". The key word here is "adaptive" and success depends on the systems repeatedly testing their latest schema against reality and adapting from that experience. In the case of biological systems, the "test" is that of successfully competing and passing along their genetic schema to the next generation. In the case of a child, it may be in the form of trying out their latest understanding of grammatical constructs by talking with their parents.

3.1 Evolution Through Prototyping

We believe the same principles apply to development of a complex software system. By developing the CMS as a series of increasingly comprehensive prototypes that are tested by actual users, we can adapt the system as we go, that is, evolve the system. This is distinctly different than the concept of "learning" prototypes. A learning prototype is considered a throw-away, designed to help understand some aspect of the problem, to fill in some missing pieces in what is still a top-down design process. Our prototypes are intended to be on the main evolutionary branch to the full CMS. In the case of the CMS, we start with a good understanding of the users' functional requirements for the system. From this we can develop a rough "system design" based on the team's experience in other software development projects. This reflects the state of our knowledge at the beginning of the CMS development process. By adopting an adaptive approach, we can make adjustments as we go, based both on our increasing knowledge and on changing external factors, such as new hardware and software products or revised user requirements.

It is important to note that, in our approach based upon the development of a series of increasingly complex and comprehensive prototypes, we also plan to pause and reassess our development strategy at the completion of each new prototype. This reevaluation process will allow, periodically, to assess the trade-off between continuing with the next prototype effort or moving to an alternative approach. The development

team will present comments and recommendations to CAMRAQ on this subject at the end of each prototype development.

3.2 Prototype Goals

Based on the experiences of the CMS design team members, there are at least five major factors that go into deciding the design goals of a particular prototype version:

1. Prioritization of the needs of the user/funder community.
2. Availability of existing components.
3. The degree to which a particular aspect of the system is fundamental (i.e., it needs to be in place in order to integrate other subsystems/components).
4. Lessons learned (and fixes required) from the previous prototype.
5. Level of funding available.

Each version must present new air quality modeling functionality to the user. That is to say, that a new version should not only provide upgrades to the underlying foundation of the system, but also allow them to perform additional tasks. Thus, for each new prototype there is a delicate balance between new user-oriented features, structural improvements, new platforms, and new documentation; all on a limited budget. This should sound familiar to anyone who has ever been associated with developing commercial software. That is probably another good way of thinking about our implementation strategy: the CMS must compete every step of the way with other governmental and commercial development efforts. Hopefully, in many cases, the most competitive strategy will be to find ways to cooperate. The fact that the CMS must remain competitive is a healthy dose of reality for the implementers. Without this kind of "real world" stress, projects like the CMS risk turning into exercises not grounded in reality.

3.3 Prototype #1

Prototype #1 is expected to be completed shortly after the completion of this report. This prototype has been developed to assist, as a parallel effort ("fast prototyping"), in the design of the CMS framework, to provide a tool for presentation/demonstration/marketing, and, most importantly, to perform actual simulation studies and be used by CAMRAQ members and other parties for practical air quality evaluations.

The prototype possesses five major characteristics:

1. Its primary focus is on the issue of tropospheric ozone.
2. It fulfills some of the needs not currently satisfied by available systems and prototypes.
3. It is portable and installable on different platforms.
4. It is easy to run through a user-friendly interface.
5. It contains advanced 3D visualization features.

The primary challenge in defining the first prototype version was to put together a system on a very limited budget that displayed some of the most important aspects of a CMS. At its most basic, a CMS must provide users with the capability to setup and run an AQM with an easy to use GUI, and to interactively analyze model outputs versus observations. The model of most immediate significance to the CAMRAQ group was the UAM-IV photochemical model as it is being widely used in the production of state SIPs for the minimization, and eventual elimination of ozone episodes. It is also being used in Canada and Europe. As it was not possible within this phase to develop a full GUI to all aspects of UAM-IV setup, it was decided to provide a limited (but still useful) capability for altering the emissions inputs into the UAM-IV. UAM-IV was chosen as the first model because of its wide usage. Other photochemical models can be added with relatively minor adjustments.

Prototype #1 is limited in the sense that it is designed to be applied only in a few regions of the US during pre-selected time periods of several days for which emission and meteorological data files are provided. New regions and new time periods, however, can be easily added to the prototype, depending upon users' needs.

3.3.1 Platform Independence

One of the key CMS attributes is the ability to run on multiple platforms. Several steps have been taken to this end in the CMS Prototype Version 1.0:

File Structure: The model output files use the MeRAF file format based on netCDF. This provides a binary, random access, hierarchical file structure that works on both Unix workstations and PC's.

GUI: All GUI work has been done using a platform independent development tool, C++/Views. This allows us to transport the application between all the major PC and workstation platforms with a recompile and minimal screen format adjustments.

3D Graphics: The savi3D was used for 3D graphics. This package runs on both workstations and PC. The main attributes of this package are its ease of use and interactive nature. It allows the user to explore a 3D dataset with a combination of isosurfaces and slices for gridded data, coupled with a variety of display options for discrete observational data. Additional discussion on savi3D is provided in Appendix C.

PC Compiler: The WATCOM C++ compiler was chosen for the PC because of its excellent multi-platform and cross-platform support. It allows a user to compile to a single set of object files, and then link to an executable targeted for any of the major and several of the minor PC operating systems. WATCOM Fortran provides the same multi-platform capabilities as well as being linkable with C++ object files. (However, WATCOM compilers are not directly portable to Unix workstations.)

The prototype will be delivered on at least two platforms: a PC-version running OS/2 Warp, and a SUN workstation version running under Solaris 2.3/2.4. The choice of a PC operating system came down to OS/2 or Windows NT, since Windows 3.1 has neither the multi-tasking nor memory protection capabilities required for an application such as the CMS. OS/2 was chosen based on its larger installed base, cheaper price, and experience base within the team.

3.3.2 Prototype Operation

The user starts the CMS Application (CMSA) written in C++ Views and is presented with a CMS logo, a map of the world and the following four buttons (see Figure 3-1):

- Beginners click here – which provides a basic description and nontechnical summary of the CMS concept and the prototype
- Simulation – to perform a new simulation (see below)
- Analysis – to analyze previous simulations or on-line data and information
- Others – which opens a new set of six buttons, as follows:
 - Tutorial – which provides a full tutorial on how to use the prototype
 - Regulations – which provides a summary of air quality regulations (national and international)
 - Education – which provides a set of electronic chapters on air pollution, atmospheric sciences, computer modeling, numerical methods, etc.
 - Communications – which provides a set of utilities to communicate with other computer systems and external data bases
 - CMS Bulletin Board – which provides access to the CAMRAQ bulletin board

- Research – which provides a technical discussion on the modules of the prototype and related references.

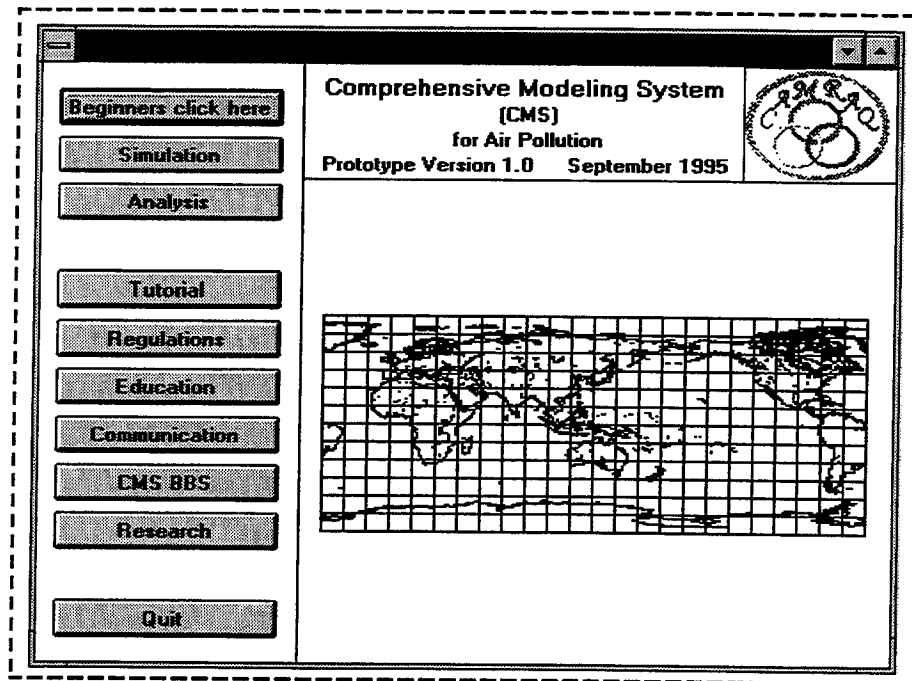


Figure 3-1
Initial Screen of CMS Prototype Version 1.0

By clicking the Simulation button, a map of the world is presented with the choice of two buttons:

1. non-reactive/first-order-reaction simulation, and
2. photochemical/higher-order reaction simulation.

If Choice 1 is selected, the map highlights those areas in which terrain data are available on-line. The user is asked to select a rectangular region by specifying latitude and longitude. Different modules are then available to simulate atmospheric dispersion using diagnostic wind models and Lagrangian simulation techniques. Simulations include first-order terms to account for deposition and chemical transformation.

If Choice 2 is selected (see Figure 3-2), the user is presented with a choice of a few regions in the US (e.g., Los Angeles, Houston, New York, Chicago, Atlanta). After the region is selected, the user is presented with a choice of photochemical models (UAM-IV, UAM-V, CALGRID, CIT, ...). Only UAM-IV is actually available for selection in this prototype. UAM-IV is provided in two versions*: the standard code, plus a fast version

* The two versions of UAM-IV are: 1) the EPA regulatory UAM, and 2) the Fast UAM. The latter version operates with a larger horizontal grid (10 km instead of 5 km) and a simplified chemistry solver.

with lower spatial resolution and simplified chemistry. Each region comes with suitable emission and meteorological data on-line.

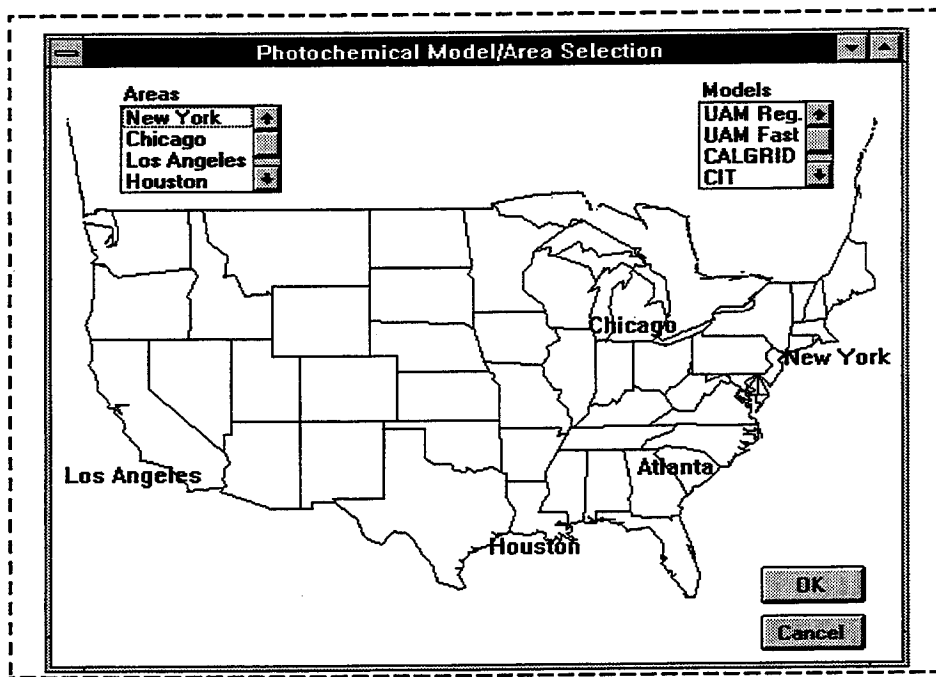


Figure 3-2
Screen from Which User Chooses the Study Region and the Photochemical Model
for Subsequent Simulations

The user selects the days of simulation and, through a simple graphical interface, manipulates the emission data by applying different controls (in percentage) to groups of emissions (e.g., traffic) and/or selected sub-regions. Through this easy manipulation of the emission file, the user can run the photochemical model under different emission scenarios during the selected days. After each selection of emission controls by the user, the CMSA calculates and provides an estimate of the associated emission control costs. In other words, the emissions base cases provided for each region can be altered to create "children" cases, i.e., alternative control strategies. For each of the three chemical groups: VOCs, NO_x , and CO , emissions are broken up into five classes: mobile, area, low level points, elevated points, and biogenics. Each of these classes can be altered by moving a slider or entering either a rate (tons per day), a percent of the base case, or a target control cost. (Note that, in Prototype #1, emission control costs are calculated for demonstration purposes only, using a quadratic relationship between emission rates and costs. However, the user can edit the constants of each quadratic function to adjust the relationship to

Preliminary results (Morris, personal communication) for ozone concentrations on July 8, 1988 for the New York region indicate that, in comparing the two UAM versions, daily maximum ozone concentrations are very similar, yet the Fast UAM runs 25 times faster (6.4 minutes versus 162.8 minutes on a 200 MHz SGI Indigo²).

After setting the model beginning and end date/times the model is started. A progress bar is displayed to help monitor the run. After each hour of simulated time, a contour map showing maximum hourly average ozone concentration is updated. The user also has the option to stop or pause/reactivate the model run at the click of a button. By entering the analysis portion of the application they can explore the model inputs and outputs interactively in a 3D environment using slices, isosurfaces, etc. This is accomplished via a link to a CMS-specific version of the savi3D environmental visualization package. Model and observational data may be viewed after and during the model run. Finally, visualization/evaluation routines allow the user to evaluate the different concentration patterns created by each emission scenario and assess the cost-effectiveness of each control strategy (e.g., by correlating ozone concentration reduction with control costs, population density and other geographically-based parameters).

All the information related to the performed simulations is automatically summarized, with text and graphics, in a text file.

The prototype is available to others than the CAMRAQ Charter Members on a fee basis through the CAMRAQ Bulletin Board. The fee includes training sessions to ensure that users obtain the maximum value from the use of the prototype.

3.4 Prototype #2

In Version 2.0 of the CMS prototype we will build onto the Version 1.0 base by incorporating more of the infrastructure described in the System Design (Chapters 2, 4, and 5) portion of this document, and expand the modeling-oriented functionality. This was anticipated in the design of Prototype #1, as the interface has facilities for the expansion into other models and model types. The actual screens developed for Prototype #1 will be retained, for the most part, but the underlying code will be "serverized", that is, broken into client and server processes that send requests and services through an interprocess communications mechanism. This is essential to allow the application to be distributed across multiple machines, allowing PC-based CMS users to access functionality that currently has only been implemented on workstation or "supercomputer" machines, such as the JEWEL emissions modeling system or components of the Models-3 project.

The most fundamental addition in Prototype #2 will be the DataBase Management Server (DBMS). In Prototype #1, all model I/O are file based and model output statements are converted to use the MeRAF file format to allow for inter-platform compatibility and compatibility with the visualization component. Other model I/O, including all data input, retain the original UAM read calls to Fortran binary files. In Prototype #2 all open/read/write call sequences will be handled by the CMS DBMS via the DCE/DFS system. Adoption of a single format will go a long way towards establishing interoperability between models. The specific makeup of the database server (combination of commercial DBM and CMS specific software) will require further

design efforts by the development team. We are considering basing the system on a commercial object-oriented DBMS such as ObjectStore. The detailed schema design also awaits further design sessions. ARIA Technologies has developed detailed ObjectStore based schema for meteorological, emissions, and AQ modeling quantities. (The development team may explore the possibility of collaborating with ARIA on this issue.)

3.4.1 Model Interchangeability

One of the key elements to the CMS modeling system is the ability to “plug and play” between different models, that is, to be able to drive different dispersion and chemistry models with meteorological or emissions data from a variety of models. This is a key to good science, as different situations may require varying degrees of sophistication in the different modeling components.

We propose to add support for additional models to Prototype #2 to allow the user to choose between: two meteorological models, three air quality models, and one emissions model.

Emissions:

EMS-95: This model is becoming very widely used.

Air Quality:

UAM-IV: This is the most widely used photochemical model. While it is true the Prototype #1 included UAM-IV, all of its file based I/O will have to be replaced to accommodate “plug and play”.

UAM-V or UAM-X or URM

UAM-V is an SAI proprietary model which is being increasingly used.

UAM-X is under development at Environ, and is expected to be made publicly available during Summer 1996. It is designed to be UAM-V compatible. Use of either of these models depends on availability.

URM is from CMU

MONTECARLO or a similar model for high-resolution simulation of transport, diffusion, and linear chemistry phenomena using Lagrangian particle modeling techniques.

Meteorological:

DWM: DWM is the simple diagnostic wind field model that is distributed with UAM-IV.

CALMET: CALMET is a widely used diagnostic meteorological model which is considered more sophisticated than DWM. Benefits include 3D interpolation of other meteorological quantities (like temperature) in addition to the wind fields.

As in Prototype #1, additions to the user interface for setting up the models would be minimal. The emphasis here is getting the models to work together. Development of

extensive model set-up GUIs must await Prototype #3, in which the CORBA and Model Servers components are added. The model user would basically go through the same steps they use today to set-up a model run. The main difference would be that they will be using the CMS DBMS to access the required data. It must be kept in mind that the inclusion of a real emissions model in Prototype #2 will make the system much more flexible than Prototype #1 for setting up real cases. Tools for importing existing model datasets into the DBMS will be developed, as will facilities for variable conversion and grid matching.

The development team will reevaluate the list of models provided above and explicitly state a priority order for inclusion of meteorological and air quality models in Prototype #2, with a discussion on advantages and disadvantages of each of the candidate models in terms of functionality for the user.

3.4.2 Operating Within The Distributed Computing Environment (DCE)

While the major thrust of Prototype #2 will be an application of the model interchangeability concept, this prototype will extend the systems design to include execution of at least one of the models on a "remote platform". This step will introduce the CMS DCE. The GUI platform will become a DCE client, and at least two other computing platforms available over a high-speed network will be available for executing the CMS DCE server. The CMS client will be authorized to access these machines and the GUI will provide a means of specifying these alternative resources.

3.5 Prototype #3

In Prototype #3 we will add the model servers and the CORBA system. The major change is that the CMS will have knowledge of each model and be capable of presenting to the user a GUI for model setup (see Section 5.5). This is a very big step in the direction of making it easier to create model runs. There will be a registry function whereby new or legacy codes can be registered into the CMS system. This process includes the creation of a GUI template that the CMS will use to define the model setup interface, as well as a catalog of data and computational resource requirements. In addition, links will be established between the CMS DBMS and a GIS system with access to geographically registered information and GIS analysis capabilities.

Additional models will be brought into the CMS in this phase: e.g., a prognostic meteorological model (NCAR/Penn State - MM5 or Colorado State University - RAMS or University of Oklahoma - ARPS), a non-UAM based photochemical model (SAQM or URM), and a puff model (CALPUFF, AVACTA II).

Prototype #3 will have a macro capability. This will allow the user to predefine large sequences of executions typically required to analyze alternative emission control strategies.

The effort in Prototype #3 will include Fortran standardization* and parallelization of selected legacy codes. (One of the members of our designing team will be starting soon a contract for parallelizing SAQM on a workstation cluster for the California Air Resources Board; we expect this activity, and others, to benefit the CMS development effort too.) Prototype #3 will include a comprehensive set of documents and tutorials on-line.

Prototype #3 is expected to be close to what in the CP report was called a "basic" CMS system, i.e., a real product for users. With additional development effort, we expect to improve Prototype #3 into the first CMS product for general use (i.e., a "basic" CMS).

The issue of the content of future prototypes is discussed further in Section 4, in the context of a more detailed description of the CMS infrastructure (see Figures 4-3 through 4-5).

3.6 CMS Products

After the development of Prototype #3, we expect real CMS "products" to be provided. These products will follow the specific design structure presented in Chapters 2, 4, & 5. We anticipate the development of a "basic" CMS product, to be followed by an "intermediate" CMS product, and finally a "full" CMS product.

* See Appendix D for an example of how Fortran Coding Standards could be established.

4

DETAILED DESIGN

The CMS design is aimed at meeting the requirements defined in Section 2.1.1. In addition, it must meet other criteria, which are not visible to the user community whose needs are spelled out in Section 2.1.2. These additional criteria derive from the performance, efficiency, and security requirements and arise from the desire to use the best tools for the CMS development, maintenance, and support. The total design must then encompass and reconcile three principle elements illustrated in Figure 4-1: user requirements, system operational requirements, and development & maintenance requirements.

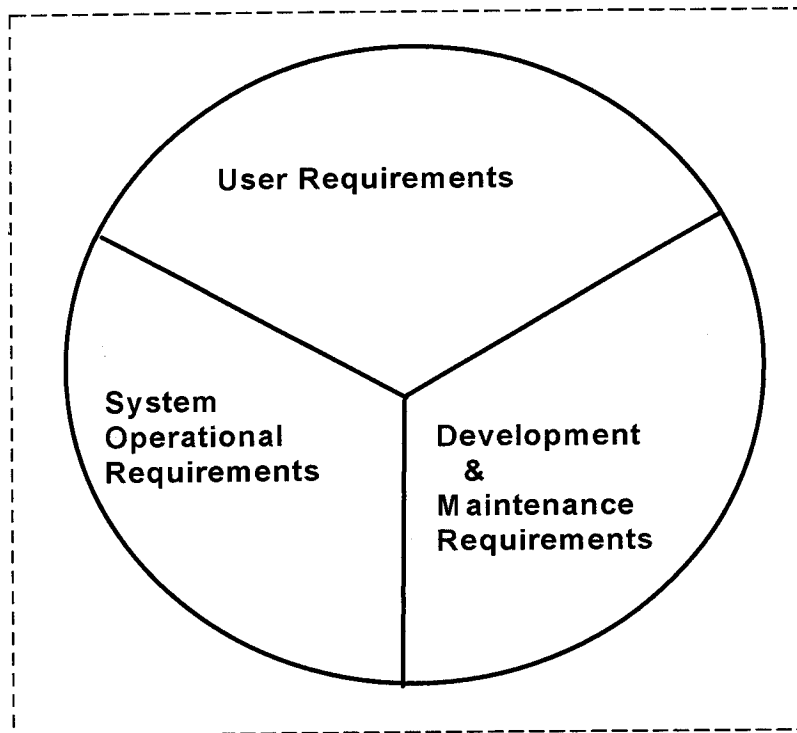


Figure 4-1
Main Design Elements

The segment of the CMS design represented in Figure 4-1 as system operational requirements includes such things as system startup, operation, error recording and recovery, system termination, communications, authentication, data locality, and integrity. The third component, development and maintenance requirements, includes the methods, technologies, budgets, and policies which will be used for the CMS. The following sections

will explore all of these issues and specify their evolution into the detailed design of the CMS.

In the subsections below, we first discuss two of the elements illustrated in Figure 4-1: Systems Operational Requirements and Development & Maintenance Requirements. (The User Requirements were discussed in the CP and summarized in the Preface of this report.) This is followed by a discussion about our principal design decisions and the CMS subdivision/decomposition.

4.1 System Operational Requirements

The CMS will possess unique operational requirements which derive from analysis of the user requirements and the nature of computer systems practices projected for the next several years. The most visible attributes of system operation include performance, reliability, and interoperability.

4.1.1 Performance

Given the interest in a broad scale of software destined for industrial, agency, and scientific use, the choice of a design and a development environment must take into account such requirements as response time, throughput, memory usage, reliability, security, and portability across platforms. For example, while some designs may address the flexibility issue (ideal for proof of concept efforts), attendant lack of speed and/or large memory requirements may negatively affect the usability of the final system. The final system should have acceptable (or better) performance on most, if not all, of the dominant computational environments, or provide means of achieving acceptable performance. The definitions of the term "acceptable performance", of course, vary with the user requirements and operating environment.

4.1.2 Reliability

There are two aspects of reliability which are addressed by the CMS: validity of model results and the inherent reliability of the system itself. This design addresses the latter issue while each of the modeling systems will provide mechanisms for assuring the reliability of their results. The CMS design must deliver services to the users at a time of their choice and with the services they require for as long as they desire. The hardware and software combination of the CMS must possess the reliability expected of any of the commercial software systems of this scale in the computing industry. The system will utilize alternative computing resources, data storage elements, and software components as automatically as possible, with minimal user intervention.

4.1.3 Interoperability

Computer-supported solutions for environmental modeling in the industrial context requires the use and integration of existing internal and vendor-developed tools and libraries which have been developed using different paradigms (such as structured analysis or object-

oriented design) or implementation languages. Historically, a significant fraction of the environmental modeling has been conducted using “legacy” codes, particularly ones that are standards for permitting and state implemented planning. There are three levels of interoperability which the CMS will address. At the lowest level, methodologies will be established to provide for future potential replacement or exchange of fundamental modules among computational models. At the next level, CMS will provide facilities for exchange or replacement of any combination of the CMS elements, models, GUI, and database with a desired alternative. At the highest level, the CMS system itself will interoperate with other full-scale systems such as Models-3.

4.2 Development and Maintenance Requirements

The success of the CMS will certainly hinge on the user’s perceptions of the system. However, those aspects of the system which should be invisible to the user will also play a key role in its long-term future. The design, development and support approach, and accompanying processes will address the issues of evolutionary design, system flexibility, legacy code integration, “on-line help”, and system maintainability as discussed below.

4.2.1 Evolutionary Design

Requirements for systems such as CMS continuously evolve as its development progresses and more knowledge of the domain and the needs of the workplace are acquired through the implementation and evaluation of early prototypes. Especially during the early phases of a project, due to the uncharted nature of the scope of the system, all aspects of the system may go through many drastic revisions before its core structure becomes stable.

4.2.2 System Flexibility

The system will continue to evolve as the available support hardware and software advance. Thus, the system should be flexible in its ability to integrate new science and its ability to take advantage of improved computational environments. A key to a successful CMS design is to provide an ability to readily hook into both legacy codes (see below) and future codes, and to provide an environment for development of new models.

4.2.3 Legacy Code Integration

The starting point for the CMS will be the incorporation of the existing model codes into the system. Where possible, this will be accomplished with a minimum of change from the original codes since it is imperative that the credibility and calibration these codes possess must be retained and exploited. As part of this incorporation process, the development team will have to decide the degree of Fortran Coding Standards that are required. To this end, we enclose as Appendix D the Fortran Coding Standards adopted for Models-3. These standards may constitute a basis for similar standards, to be adopted for the CMS in the future.

4.2.4 “On-Line Help”

In its full configuration, the CMS will present the user with an extremely robust and flexible system. To support such a system in everyday use, the CMS will have to provide an interactive “Help” facility with the features that all PC and Macintosh users have come to expect from every commercial software package.

4.2.5 System Maintainability

The CMS will utilize a myriad of model, file, and system programs which are written in an equally varied collection of programming languages. The overall design of the CMS will utilize the object-oriented design approach and all modules integrated within the CMS will be placed under a common code control system. All new programs will be developed with object-oriented tools and wherever possible, existing “legacy” codes will be furnished with programming interfaces (or wrappers) which will give them the properties of new system objects. Further, legacy codes may be optimized for use in the CMS, providing the same results, but with better performance.

4.3 Principal Design Decisions

The boundaries for the design are thus drawn: the user requirements, system requirements, development requirements, and the hardware and software systems with which they will operate form the basis for this CMS design. There are several major decisions made during the formative stages of the design which should be illuminated here. These are: system partitioning, priorities for design and implementation, and necessary design compromises.

4.3.1 Partitioning of System Functions

With the complex combination of applications programs, databases, operating system, and user interfaces in the CMS, it is clear that some form of “modularization” or “partitioning” of the entire CMS must be the first step in the design. This is even more true when most, if not all, of these functions may operate asynchronously and, further, may be instantiated on separate computational platforms. In parallel processing jargon, the phrase “functional decomposition” is sometimes used to describe the process of dividing applications computations into two or more somewhat independent modules. In the remainder of this detailed design discussion, all three descriptive terms will be used (“partition”, “modularization”, and “decomposition”). The primary criteria used when partitioning is simple: “What makes sense?”. To better understand the role of partitioning in the design of the CMS a few questions are presented, and answered, below.

- **What elements should be combined together in a single entity?** There are obvious choices implied in this question. It would seem clear that it makes little sense to combine a module for printing map files with the advection subroutine from a meteorological model. On the other hand, all functions related to initiating,

operating, and terminating any set of emissions models would be likely candidates to end up in the same partition. Thus, the identification of a general 'emissions' function springs forth. This process of agglutination could continue to the point where this emissions function could be incorporated directly into the GUI or into the overall CMS management function.

It is at this point, where the system architect decides that a further accretion of functions would make the entity cumbersome and unwieldy from a programming and system maintenance point of view. So we rest on a stand-alone 'emissions function'.

- ***What entities should operate over a system of shared resources and networks?*** The zeal to partition functions into small subsystems for ease of system's management as well as increasing the opportunities for parallel operation (and perhaps consequent performance improvements) must be tempered with the knowledge of the performance and system integrity impacts for the design. A simple example is where a Fortran application is being divided into smaller and potentially parallel modules. If a key element of this application is a large, global, data array which must now be accessed by all modules, the performance of the program could be seriously affected if the modules are in separate execution spaces, or worse, on separate machines connected by a slow network.

To decompose the user interface into different modules such as the GUI, the graphical rendering system and non-graphical command line functions, each of which must access and manage aspects of the terminal screen, and are thus best done within a single functional entity.

- ***What needs to be developed for CMS and what can be obtained from other sources?*** The goals of the CMS development effort is to maximize interoperability between many different platforms and, at the same time, optimize the design and implementation resources of the CMS development team. The plan mandates widespread, albeit prudent, use of components provided by hardware and software vendors, both commercial and public domain. Selection of major modules in this category will influence the process of defining the CMS modules. For example, we have chosen the distributed computing environment (see Section 5.9) as the basic, standard mechanism for distributing and managing the system functions, even on a single platform. Since the DCE provides an integral security system, the security function need not appear in the CMS features which need to be developed.

In a similar fashion, the DFS which accompanies the DCE can supply many functions that would otherwise have to be developed for the CMS database scheme. As the reader will note, we have chosen some key elements of the CMS to be provided by developers other than the CMS team. The DCE, the database manager, the program object manager, the GIS manager, and the GUI

are all major modules which will be acquired from these other sources. As it is, the CMS will have its hands full with the integration and adaptation of these facilities as well as the creation of the CMS customized functions.

- ***What are the modern software technology guidelines for partitioning?***
Current practices in system development prescribe mechanisms for identifying the salient characteristics of 'objects' and their definition and implementation. Further, the coordination of several software development groups, each with its own special backgrounds and experience, will be made much easier if the appropriate functional decompositions are established. Software development and testing of a complex system will also require that each of the defined functional partitions have sufficient stand-alone capability so that it may be developed relatively independently of other functional components, and rigorously tested without requiring that other members of the system be present.

It is very helpful to define the general nature of these CMS components and to specify their relationships to each other. Hence, we introduced (in Section 2.1) the concept of 'client' and 'server' functions to describe certain intrinsic behavior of these components. As mentioned earlier a "client" requests service, while a "server" provides the requested service. There is a good deal more technical substance to these designations than the casual one we just used. For example, if an element of a system is a "pure" client, it will never provide any kind of service to any other entity and thus never receive 'requests'. The client generates requests and awaits responses from one or more other elements in the system. The server expects requests and provides responses. If it is a "pure server", it will never make requests of any other subsystem, nor does it ever expect to receive messages of the "response" category.

These definitions assist in both the system design and in maintaining the system stability and integrity during its operation. Of course, an entity may adopt differing personalities depending on how the function is being utilized at the moment. In the CMS design, we describe the user interface as the CMS client and specify that it be a "pure" client. This means that if the function is being executed at a personal PC or workstation, the user may issue requests to the CMS through the CMS client but none of the resources of the personal workstation would be accessible to the CMS or its other users. In the case of the CMS management server, it will respond to requests from the CMS client but may also become a client to one of the other servers, such as the database server. This is possible only if the management server can "trust" the other servers, and vice-versa; a situation which must be ensured by the CMS security system.

4.3.2 Priorities for Design and Implementation

This document represents a "first level" of detailed design of the CMS. Our realistic assessment of available funds and key resources (and common sense) contributed to our decision of developing the CMS as a series of evolving prototypes. The approaches of prototyping and evolutionary development have been discussed earlier and

permeate every aspect of this document. Our Prototype #1 has been relatively easy to specify since it limited itself to existing models which will constitute subelements of the CMS. The next phase of this project will have to address the actual priorities of work for the detailed design and implementation. The descriptions of the prototypes and the amount of detail in the following design specifications reflect our recommended sequence for the development.

4.3.3 Necessary Design Compromises

In the best of all possible worlds, “legacy codes” would only be present in early implementations of the CMS. It is clear that some of these codes will continue to be key operational components of the CMS. All aspects of the CMS must then present as facile an interface for these legacy codes as it will for a new generation of codes. This means that there will be some compromises made in the programming interfaces, but not the performance or reliability of any of the components.

A more subtle but essential set of compromises made in this design are those involving the use of as many existing elements as possible to reduce the development time and cost for the CMS. While those described here are acknowledged to be at the edge of the computing state-of-the-art, they present established interfaces and functionality which might be accomplished more efficiently or elegantly in a fully-customized CMS design. The design in this document attempts to minimize these compromises, but they will be apparent to a knowledgeable software engineer who can envision an “idealized” CMS.

4.4 The CMS Subdivision/Decomposition

Figure 4-2 provides a diagram of the major CMS elements as they will appear in the remainder of this design document. This block diagram forms the basis for subsection headings in Chapter 5. An identifier ([1] -> [11]) has been assigned for use in later discussions. A heading of “CMS” indicates that the particular module will be designed and implemented by the CMS team. The heading “DCE” indicates that the module will come from the standard distribution of the distributed computing environment. A heading of “CORBA” indicates a module provided by the Object Management Group (OMG) of the Open Systems Foundation.

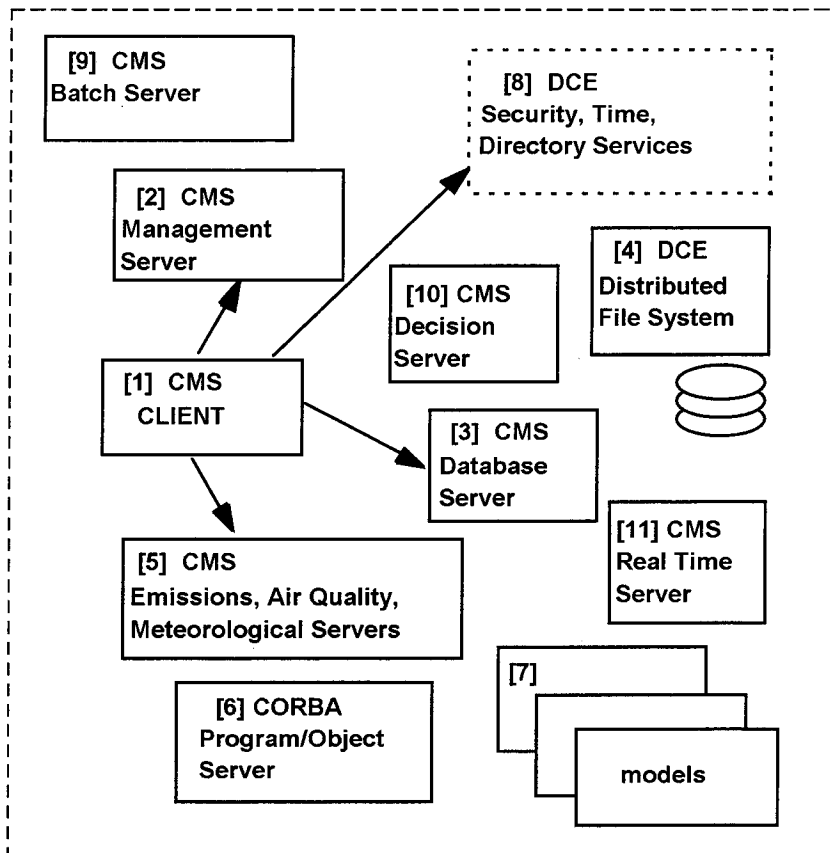


Figure 4-2
Major CMS Elements

The relationships between the modules involves one or more communications paths between modules which would clutter up the figure intolerably. We have chosen to describe these relationships in Table 4-1, using the module identifiers "1" through "11". Abbreviated notes have been placed in many of the table entries to indicate significant design issues for intercommunications. In each cell of the table a two character field surrounded by brackets indicates the communications path. For example, "[1-2]" identifies a communications from Element #1 (the CMS client) to Element #2 (the CMS management server). The communications "pairs" from this table will appear in each of the subsections of Section 5 where appropriate.

**Table 4-1
System Communications Table**

to/ from	client [1]	manager [2]	database [3]	DFS [4]	model servers [5]	CORBA [6]	models [7]	DCE [8]	batch [9]	decision [10]	real-time [11]
client [1]	----	[1-2] all resource requests	[1-3] specify extract dispose	[1-4] data	[1-5] setup params	[1-6] model elect	[1-7] run sync	[1-8] login, find CMS	[1-9] queue jobs	[1-10] setup	[1-11] initialise params
manager [2]	[2-1] per- missions	---	[2-3] startup shutdwn	[2-4] open system files	[2-5] startup shutdwn	[2-6] startup shutdwn	[2-7]	[2-8] locate resource permit	[2-9] startup shutdwn	[2-12] startup shutdwn	[2-11] startup shutdwn
database [3]	[3-1] data	[3-2] errors	---	[3-4] files data subsets	[3-5] extant structs	[3-6] ###	[3-7] extract data	[3-8] RPC to DFS	[3-9] ###	[3-10] data extracts	[3-11] ###
DFS [4]	[4-1] data	[4-2] errors	[4-3] data	---	[4-5] data	[4-6] objects	[4-7] data	[4-8] DCE internal	[4-9] ###	[4-10] data	[4-11] ###
model servers [5]	[5-1] data sync	[5-2] errors	[5-3] ###	[5-4] files data	---	[5-6] objects	[5-7] setup xforms	[5-8] locate DFS permit	[5-9] ###	[5-10] ###	[5-11] ###
CORBA [6]	[6-1] errors	[6-2] errors	[6-3] ###	[6-4] request objects	[6-5] setup API	---	[6-7] ###	[6-8] locate DFS permit	[6-9] ###	[6-10] ###	[6-11] ###
models [7]	[7-1] errors data	[7-2] errors	[7-3] requests data	[7-4] requests data	[7-5] status	[7-6] ###	---	[7-8] ###	[7-9] status	[7-10] ###	[7-11] sync requests
DCE [8]	[8-1] time location permit	[8-2] time location permit	[8-3] time location permit	[8-4] time location permit	[8-5] time location permit	[8-6] time location permit	[8-7] time location permit	---	[8-9] time location permit	[8-10] time location permit	[8-11] time location permit
batch [9]	[9-1] status	[9-2] errors	[9-3] requests setup	[9-4] requests data	[9-5] ###	[9-6] ###	[9-7] startup shutdwn	[9-8] locate DFS permit	---	[9-10] setup run	[9-11] ###
decision [10]	[10-1] data	[10-2] errors	[10-3] requests data	[10-4] requests data	[10-5] ###	[10-6] ###	[10-7] run decison models	[10-8] locate resource permit	[10-9] ###	---	[10-11] ###
real-time [11]	[11-1] data status	[11-2] errors	[11-3] data	[11-4] data	[11-5] status data	[11-6] ###	[11-7] sync data	[11-8] locate DFS permit	[11-9] ###	[11-10] ###	---

Note: "###" indicates no communications; "----" indicates not necessary

The reader may be quite confused when faced with the diagram in Figure 4-2 and Table 4-1. “Does the CMS system have to be this complicated?” one might ask. To put things into the right perspective, three considerations should be made.

1. Table 4-1 is included as a basis for future design discussions and exposes many low-level interfaces.
2. The simplest use of the system (with models, visualization, and control, self-contained on a single workstation) does not require more than three components.
3. In a fully-configured, multi-user, multi-site CMS, some aspect of each of the modules shown must be present.

The clear amelioration for the apparent complexity and depth of the CMS is that the bulk of the system consists of software and concepts which have already been developed and tested. The task of the CMS development team is to assemble these components and configure them into an integral scheme which can operate efficiently on small modeling problems on single platforms as well as very complex modeling problems on multiple computing platforms.

Figures 4-3, 4-4, and 4-5 show the elements of the system which will be exercised during the Prototype #1, #2, and #3 phases of this project.

The following chapter presents a detailed breakdown of each element of the CMS. Each of the aspects of the design which must be dealt with before commencing implementation are given a separate subsection. While not all of the subsections in Chapter 5 are addressed in the same level of detail, the subsections remain as “placeholders” to be filled in during the next phase of the program, i.e., Phase II - The Development Phase.

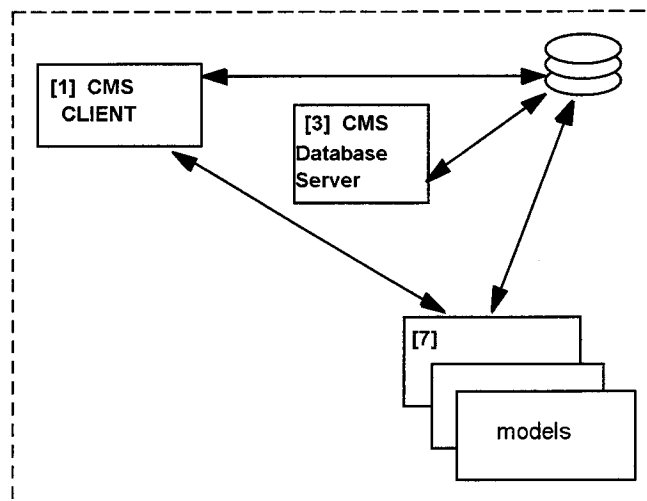


Figure 4-3
Prototype #1 Elements

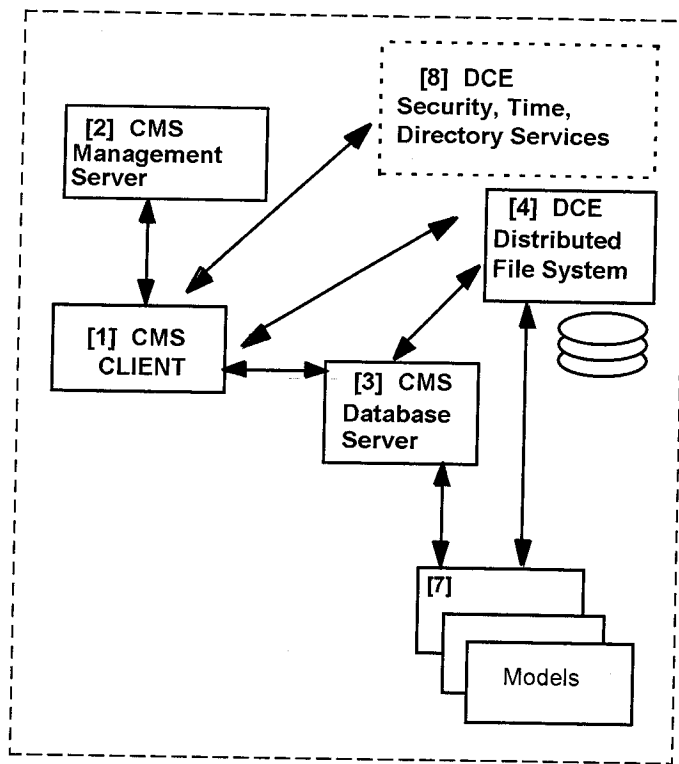


Figure 4-4
 Prototype #2 Elements

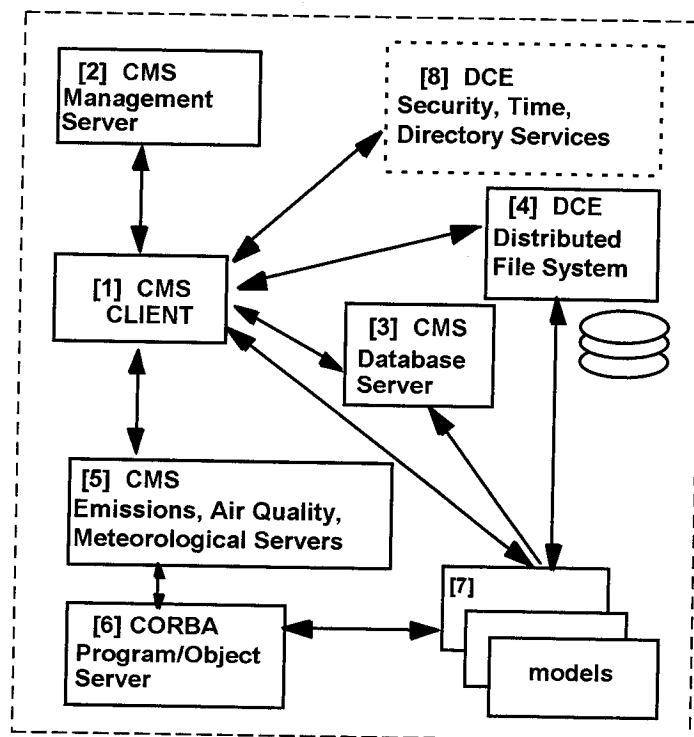


Figure 4-5
 Prototype #3 Elements

5

MAJOR CMS ELEMENTS

This chapter expands the discussion of the eleven CMS elements illustrated in Figure 4-2 and their interactions summarized in Table 4-1. These elements are:

1. The CMS client
2. The CMS management server
3. The CMS database server
4. The distributed file system (DFS)
5. The CMS model server
6. The CMS program/object server
7. The CMS models
8. The distributed computing environment (DCE)
9. The CMS batch server
10. The CMS decision server
11. The CMS real-time server

All elements are discussed below. Discussion generally includes a description of 1) element components, 2) communications (internal, to CMS manager, to database server, to the DFS, to the CORBA, to the model server, to the models, to the DCE, to the batch server, to the decision server, to the real-time server, and responses to requests), 3) concurrency, 4) performance, 5) error handling, 6) implementation, and 7) operation.

5.1 The CMS Client

The CMS user interface, which we shall call the CMS client function, is shown in Figure 5-1.

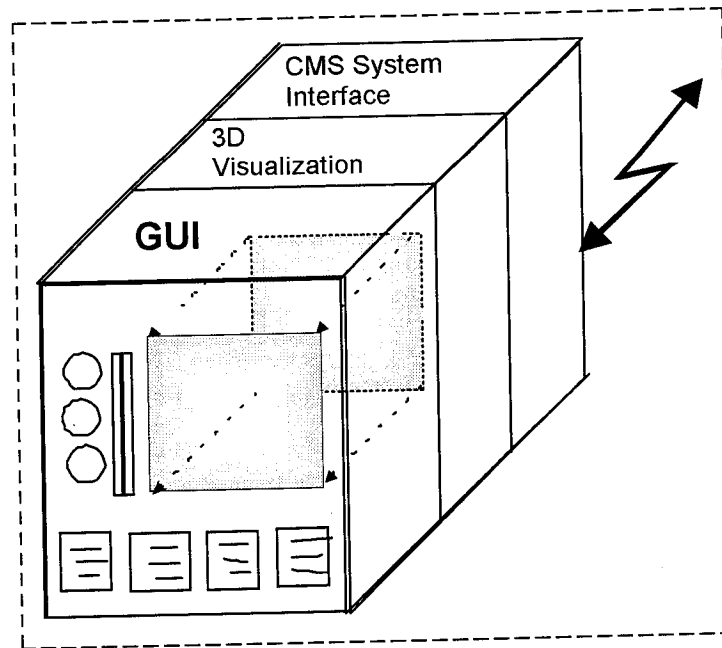


Figure 5-1
The CMS Client [1]

5.1.1 CMS Client Components

There are three separate components in the CMS client. They are: 1) a standard GUI, 2) a 3D visualization system; and 3) the interface module between the client and the remainder of the system. The CMS client provides an integrated 3D graphical environment for the analysis and synthesis of all relevant spatially referenced data, including input or set-up data and model outputs. This includes meteorological, emissions, air quality, topographical, land use, engineering, landscape, and architectural data. The ability to explore multiple datasets simultaneously in a highly-interactive environment is crucial in identifying their mutual relationships and in weeding out inconsistent or implausible data. Users will be able to set a flag identifying such data as bad from within this graphical environment.

5.1.1.1 The GUI. An important principle of the CMS is that the user will encounter a "common look and feel" in the user interface. This "common look and feel" is intended to bridge all of the hardware and software platforms and all of the modeling systems within the CMS. The GUI uses a "standard" system for presenting menus, tables, dials, entry boxes, etc., to the user. This appearance and the response of this interface will be virtually identical on PCs, Macintoshes, and Unix workstations. While each model setup, execution, and analysis will differ in many ways, those components which share commonality (for example, setting windfield velocities) will have similar, if not exactly identical appearances.

Many tools for building, modifying, and updating these GUIs are available for multi-platform implementations. The CMS GUI will be built upon these tools. The GUI for the CMS Prototype #1 is being built using the Intersolv Corporation's C++/Views. (Other systems available are: Z-App from Inmark, Open Software's OpenUI, XVT from XVT Software, Open

Incorporated's Aspect, Neuron Data's Open Interface. The usage of all of these "GUI builders" is very similar.)

The use of a 'platform neutral' GUI builder makes it possible for users to tinker with the "look and feel" of their interface very quickly and have that interface appear almost automatically on all of the other CMS client platforms. Each CMS user community may then tailor the interface according to their needs and working styles, and not be forced to deal with an unfamiliar or uncomfortable scheme which sounded good during the initial project stages.

The appearance and "feel" (which is how menus and buttons respond visually and physically to user manipulations) couples with the "semantics" of the interface. The "semantics" are those actions which the CMS client, and the remainder of the system, undertake as a response to the user. The GUI builders listed above all base their 'semantics' on the use of C++ objects. Since C++ objects will be the basic building blocks for CMS implementation (Fortran is used for most of the scientific models), 'semantics' of CMS activities will be described in C++ objects and methods.

5.1.1.2 The 3D visualization. The basic component of the CMS client for the visualization of data is the OpenGL system which originated at SGI and has become an industry-wide standard, supported by hardware vendors and commercial software development groups. (While OpenGL is not available for DOS and Windows 3.1.1, it is available for OS/2 and Windows NT, and soon for Windows 95 and Macintosh.) OpenGL provides for the 3D transformations and rendering of objects. (Two dimensional displays are a simple subset of the 3D.) The OpenGL system provides reasonable graphics performance even with the simplest graphics hardware available. This is particularly true of the OS/2 version of OpenGL. Thus, CMS has available a very powerful and "platform-neutral" tool for displaying its data, on PCs or high-end graphics workstations.

The graphics visualization system interacts with the GUI to select, orient, and interpret the data being displayed. In addition, it can create requests to the database for information to be loaded for a requested display field.

5.1.1.3 The CMS interface. The CMS client accesses other CMS functions (such as the database manager) through a set of programming protocols defined by the CMS infrastructure. For multi-platform instances these protocols use the DCE described in Section 5.8. This module of the CMS client is, in fact, made up largely of the DCE client functions.

The best way to illustrate this concept is to describe what happens when a user accesses the CMS. Let us assume that the point at which the user interfaces with the CMS is a PC or Unix workstation.

- The user first logs into the workstation by giving a unique name and password. This process may be skipped when using a personally owned PC, but good system practice mandates even login procedures for this level of machine. The

user may choose to perform a number of non-CMS operations, which will not invoke the CMS interface.

- When access to the CMS is desired, a separate login/authentication operation must take place to validate the user's ability to use any or all of the CMS. Triggered by a user command, the DCE client portion of the CMS interface contacts the DCE server and conducts the necessary system handshaking and 'caches' authentication for future use. The DCE server notifies all of the CMS servers that this particular user is active and specifies what services they may access and with what permissions.
- After the client is validated, the next step for the CMS interface is to locate and contact the CMS management server to register itself with the manager which initiates a variety of procedures, including monitoring of the client and initiating a log for the client session.
- Other aspects of the DCE client do not appear as actual functional modules in the CMS interface. Instead they involve the DCE mechanisms for using RPC in place of the standard Fortran, C, and C++ subroutines and function calls. This is described in Section 5.8.
- The CMS interface function also registers all inter-system resource usage with the CMS management server so that it can deal with error conditions and negotiate resource allocations when there are conflicting requirements among the members of the system.

5.1.2 Communications

The CMS client can communicate with every other member of the system. Table 4-1 contains an entry for communications to every destination. While the client must seek authentication from the DCE to access a resource and must register this service connection with the CMS management server for the first communication, all subsequent communications take place directly between the client and the target resource.

In general, communications cover 12 types of actions.

1. Internal communications
2. Communications to the CMS manager
3. Communications to the CMS database server
4. Communications to the DFS
5. Communications to the CORBA
6. Communications to the model servers
7. Communications to the models
8. Communications to the DCE
9. Communications to the batch process server
10. Communications to the decision server

11. Communications to the real-time server
12. Responses to requests from the CMS clients and servers

These actions are discussed below.

- **Internal communications.** The GUI, the 3D visualization process, and the CMS interface all operate as independent processes on the CMS client host. Coordination among these three elements is through messages between cooperating pairs of these functions. The implementation will avoid using other coordination techniques, such as shared memory variables, so that the functions could be distributed to separate processors in the future if new hardware or system developments make such a step possible and desirable.
- **Communications to the CMS manager.** The CMS client initiates communications with the CMS manager for the following functions.
 - Initial registration as a CMS client
 - Initial request to access a CMS resource
 - Notification to free up resources
 - Reporting errors detected by the client
- **Communications to the CMS database server.** The CMS client initiates communications with the CMS database server for the following functions.
 - Requests for GIS and map data.
 - Requests for access and location of datasets. The client will usually be unaware of the organization and whereabouts of physical files which make up a dataset. Where access controls are required on certain files and datasets, the database manager will provide the requisite permissions.
 - Requests for subsets of data. Where subsets such as “slices” from a gridded dataset or portions of discrete datasets are required, they are requested from the database server which performs the necessary operations on the files themselves and returns the data to the client.
 - Requests to dispose of datasets by copying, removing, and/or creating one or more physical files.

- **Communications to the DFS.**
 - Requests to retrieve data files.
 - Requests to store data files.
- **Communications to the CORBA.** Selection information for the models being setup and executed.
- **Communications to the model servers.**
 - Parameters for the models.
- **Communications to the models.**
 - Parameters
 - Start/run
 - Pause
 - Synchronize
- **Communications to the DCE.**
 - System login
 - System logout
 - Request to locate a resource
- **Communication to the batch process server.**
 - Job setup information
 - Job queuing information
 - Job queue changes
- **Communication to the decision server.**
 - Setup information for the decision models.

- **Communication to the real-time server.**
 - Send initialization parameters.
- **Responses to requests from the CMS clients and servers.**

5.1.3 Concurrency

The CMS client operates asynchronously and concurrently with all of the CMS servers it is employing.

5.1.4 Performance

The host machine for the CMS client must be able to deal with keyboard, mouse/pointer, and other input devices and display the effects on a color graphics screen with sufficient speed as gives the user a distinct interactive feeling. Modern PCs running at 100 MHz or better, Apple machines utilizing the PowerPC chip, or any modern Unix workstation easily fill this requirement. If other elements of the CMS such as the database server or the models themselves are to run on this same platform, they will be sharing the computing power of these machines and may adversely impact the interactive aspects for the client. Performance requirements of these other elements are discussed in their respective sections.

The ability for a moderately priced PC-based system (i.e., a Pentium CPU running at 120 MHz) to perform model computations and database extraction, as well as the GUI and display functions, is being demonstrated in Prototype #1.

5.1.5 Error Handling

The CMS client will attempt to handle error conditions according to the standards expected of all modern commercial software products, without relying on user input. There are two sets of errors sources which must be dealt with, those internal to the CMS Client itself and those arising from the external servers. Wherever possible, the GUI will be used to report errors to the user and request guidance on how to proceed.

- **Internal errors.** The types of internal errors which the CMS client will have to manage will be those arising from the host machine and its software and those stemming from errors coupled to the CMS client software system. Errors which impact the CMS will be reported to the CMS management server.

Fatal errors, of course will most likely be dealt with by the platform system, resulting in termination of the CMS client program. When this doesn't occur, the client will attempt to retain at least the GUI so that, if needed, termination procedures may be under the control of the user.

5.1.6 Implementation

The CMS client will be implemented with C++ tools and C++ objects. Newly developed code for the CMS will meet project specifications for programming and documentation.

5.1.7 Operation

A typical CMS operational day may find many clients logged in and accessing one or more CMS server functions. Each of these clients may have several activities in progress on their workstation which may give rise to a multiplicity of processes throughout the CMS on behalf of a single client. When several clients are competing for the use of the resource, it is the responsibility of that resource manager (for instance, the database manager) to establish priorities for service with the help of the CMS management server and, possibly, interaction with the clients involved.

5.2 The CMS Management Server

Figure 5-2 presents a block diagram of the CMS management server. Its principle functions are logging, error control, startup and shutdown, and management of all of the CMS resources.

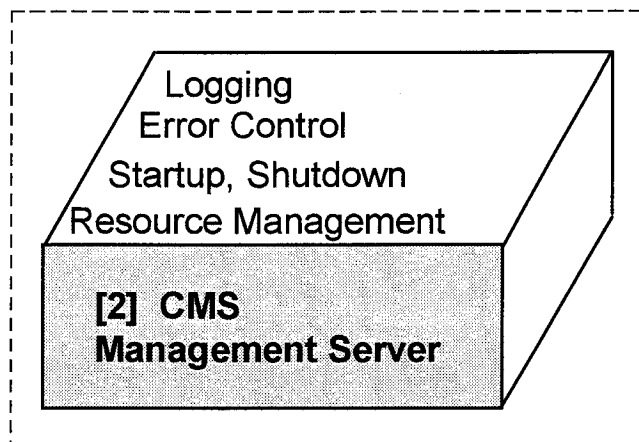


Figure 5-2
The CMS Management Server [2]

5.2.1 Components

The CMS management server is the first element of the CMS to be instantiated. It contains methods to manage its own processes and to provide four major functions to the system at large. These functions are logging, error control, startup/shutdown, and remote management, as discussed below.

5.2.1.1 Logging. All CMS servers and clients and all newly developed CMS models will contact the CMS management server when starting up and terminating or when an error is encountered. This information is logged by the management server for post-mortem

analysis, system reporting, and, where appropriate, accounting processes. For this reason, the events being logged will contain more than time stamps. Data such as resource utilization by the server, message traffic, etc., will be included.

5.2.1.2 Error control. The CMS manager is responsible for tracking all errors detected within the CMS. All servers and clients will report errors. If the problem cannot be resolved by the individual client or server and has not caused the termination of the affected server, the CMS manager will attempt to resolve the problem in a system-wide way. For example, one of the file systems could experience a fatal error but that server may still be able to contact the CMS manager, the manager will attempt to restart the server, if necessary. If the file system could not be restarted, the CMS manager would contact all clients and servers which it has recorded as currently requiring that resource as to the problem. If an alternate strategy is available to replace that resource (as in a backup file system), the CMS manager will bring that resource on-line and notify all active clients and servers of the change.

5.2.1.3 Startup/shutdown. The CMS manager is responsible for the initiation and termination of all servers in the system. Normally there will only be a single instance of CMS servers in a system. However, when a CMS consists of more than one geographical site, there could be several instances of a server of the same type active at one time. The CMS manager ensures that sufficient services are available to complete CMS user tasks.

At the outset, the only CMS servers which must be active are the DCE server and the CMS manager. As resources are required for the first time, the manager will start their respective servers on the requisite hosts. Once started, a server will persist in operation until terminated by the CMS manager.

When a server must be shutdown, other than because of a catastrophic situation (crash) of the server's host, the CMS manager is contacted and a request for a shutdown is made. This permits the CMS manager to contact any other affected resources before terminating the requester. The manager may choose to shutdown a server for a number of system reasons unrelated to errors. It may decide, for example that a particular server has been unused for a specified length of time and can be terminated until a later request.

5.2.1.4 Resource management. As we described earlier, the CMS management server maintains cognizance over all active elements of the CMS. The information which is part of this management chore includes details about the resources available to each active element. In addition, the CMS manager is also aware of the resources required by each member of the system. Some of this information is static and built into the management task (the database server must have at least one DFS facility available to it). Many other requirements are dynamic and derived from the client or server when it registers a request for a resource. These can range from very obvious items, such as the amount of file space predicted for the results of a model run, to the more subtle properties of a server requiring a high-bandwidth connection to its database, if high-execution performance is desired.

When clients and servers make requests for resources, the CMS manager will determine the most appropriate location from which to access that capability. An example of this would be the decision as to what platform to use for a particular model run, depending on the model's needs for computational performance and the locality of the source and result data files within the system. When several CMS elements are competing for the same, singular resource, the CMS manager must arbitrate the contest and allocate critical elements depending on a CMS team established policy.

5.2.2 Communications

The CMS management server communicates with all other CMS components except for the models, as shown in Table 4-1. These communications are demonstrated below (note that communications to the models are not applicable).

- **Communications to the client.**
 - Shutdown command: The manager can shutdown CMS clients when dire circumstances arise. No client or server can unilaterally terminate itself without first contacting the management server.
 - Status request: The manager may inquire of the client as to its resources and the status of its resources.
 - Permissions: The CMS manager may have to change the resources and permissions for CMS access dynamically during a CMS client session. It is possible to conceive of situations where a client may be pre-empted in its use of CMS facilities at which point the user must be informed through the GUI.
- **Communication to the CMS database manager.** Startup/shutdown.
- **Communication to the DFS.** Open the CMS system files: Files needed for CMS management will be local to the host running the management server. All system critical files will need to have backup images which are not stored locally and would need to be accessed by another host running a 'backup' manager in the event of a major failure of the manager host system. These system files will be maintained by the DFS.
- **Communication to the CORBA.** Startup/shutdown.
- **Communication to the model servers.** Startup/shutdown.

- **Communications to the DCE.**
 - Permission request: Although the manager is responsible for all CMS activities the primary mechanism for authentication and location of the CMS elements rests within the DCE.
 - Locate resources
- **Communications to the CMS batch server.** Startup/shutdown.
- **Communications to the CMS decision server.** Startup/shutdown.
- **Communications to the CMS real-time server.** Startup/shutdown.
- **Response to requests from CMS clients and servers.**

5.2.3 Concurrency

The CMS management server operates asynchronously and in parallel with all of the CMS clients and servers. Since the manager is the cornerstone and possible bottleneck for the system, its behavior must at the same time be very deterministic and stable as well as responsive to a variety of asynchronous requests. At this juncture, a multi-threaded design appears to be the best means for achieving these goals. This aspect of the system needs further consideration.

5.2.4 Performance

The performance of the CMS manager hardware and software system must be sufficient to ensure that it never becomes a bottleneck to any constituent of the CMS, except where system integrity considerations causes the manager to slow down or terminate operations. The tasks assigned to the management server are reasonably self-contained, simple, and should require modest amounts of machine resources even under heavy request loads. High-performance PC or PC-based system servers as well as modern Unix workstations are more than adequate for this function.

5.2.5 Error Handling

Section 5.2.1.2 describes the role of the CMS management server in handling errors for the entire CMS. This description includes the methodologies for internal CMS management server errors as well.

5.2.6 Implementation

The CMS management server will be developed from scratch. The development environment will be C++.

5.3 The CMS Database Server

Figure 5-3 depicts the CMS database server and its primary relationships to other CMS elements.

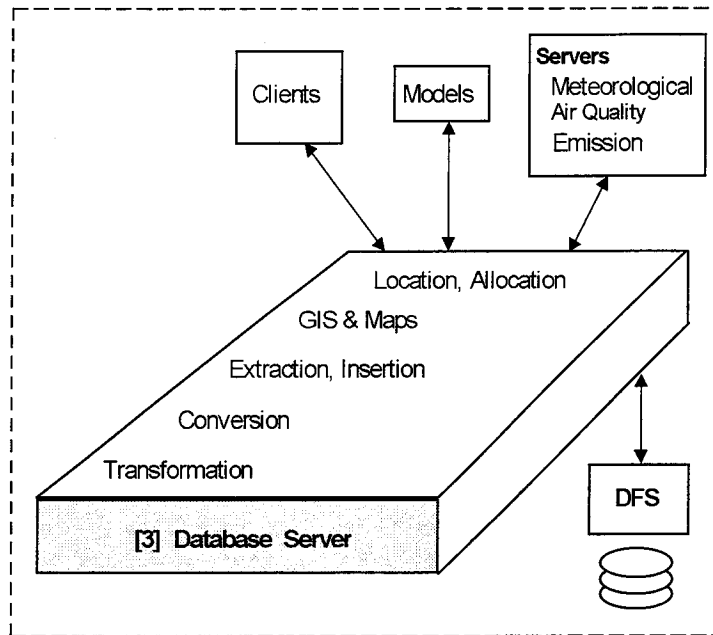


Figure 5-3
The CMS Database Server [3]

5.3.1 Components

The database server has one of the most complex jobs in the CMS. The CMS development of the database server will consist of creating an integrated design of existing software systems and establishing the interfaces to other CMS servers. New software development by the CMS development team would be limited to the location/allocation mechanism described below, while the other elements (GIS, data extraction and insertion, conversion, and transformation) will be extracted from software developed elsewhere.

5.3.1.1 Location/allocation. In the past, environmental modelers have had to deal with a variety of naming conventions for their data files. Usually these were limited by a specific operating system limitation. We have all had to deal with PC file names like TFDGF.FIL which is further qualified by the disk partition on which it resides: "D:\TFDGF.FIL" and even further by the subdirectory in which the file actually appears: "D:\MODELATFDGF.FIL". This scheme ends up encoding into the file name its actual location. Instead, a CMS user filename should only be required to convey the contents of the file and its relationship to the CMS. "The_First_Data_Grid_File" might look strange and inappropriate but it expresses how the CMS user should be able to deal with system-wide data.

The DFS provides the capability for unique, location-independent file naming. However, system conventions for the base hosts may require fragmentation of the naming system. In addition, many logically-associated datasets may be aggregated for one or more uses. This means that the output of a meteorological model and an emissions model for a particular event under study could consist of three, four, or more distinct files. The user must be aware of these separate entities and address each from their setup systems and models.

As an example, some large meteorological runs require more storage space for their output than can be accommodated in a single disk file allocation. It is not unusual to find output files scattered about various file systems and even various host machines. One might see time steps one through ten in a file called "hosta/home/user1/hour1_10" and the next set of timesteps in file "hostb/home/user1/hour11_20". Further, many modern meteorological models produce more than just a set of files containing fields which may be contained in separate files. Some files contain time-varying values for each grid point; others might be time-invariant or might represent domain subsections only.

In the CMS, these files will not only be referenced at the time models are run or results analyzed. Many other activities will deal with filesets and files in the CMS. Moving, copying, and deleting filesets should require that the user only be aware of and need to use a single fileset name.

The mapping of fileset names to the family of actual file names will be performed by the CMS database server for all CMS files. When a new fileset is being created, the database server must be able to discern the structure of the fileset without prompting the user. For example, if one chooses to use a particular mesoscale meteorological model, the meteorological server will provide information to the CMS client about the requisite input and output datasets. The database manager may then allocate space for the output files and return the internal DFS names to the client/models.

5.3.1.2 Geographical information system. The CMS will require an array of data to support the model setup, execution, and analysis. A large amount of this information will have a geographical basis. Maps, inventories, population, and business locations form a large but diffuse set of data which must be organized in unique ways for different types and sets of CMS applications. There are several candidate GIS available which provide the functionality and can operate on different computing platforms. These include ARC/INFO and ATLAS/GIS, among others. The selection of a GIS will be made during the next phase of this project. Principle criteria for choosing a GIS will be system reliability, ease of use, cost, performance, and the availability of multi-platform implementations.

5.3.1.3 Data extraction and insertion. Quite often, modeling and analysis processes require only a portion of a database on file within the CMS. Typical examples are when only one hour's or one day's worth of data is needed from the output of an emissions or meteorological run; or when an analysis or visualization of one or more slices of a grid will suffice. Past practices have usually required the fragmentation of the datasets into smaller, separate files. This is particularly true when the data must be transferred over low or moderate speed networks.

Oftentimes, an entire large file is transferred from site to site so that a model or analysis program on the target machine can extract the necessary data. A better solution would be to perform the extraction process at the location of the data file and transmit only the subset. It would be desirable that this process be “transparent” to the user at the CMS client. The database server will make these determinations, on the basis of “high-level requests” from the user, and perform the operations of extraction (and insertion) of data subsets from (and into) the relevant CMS files. As an example, the user’s model setup process on the CMS client might indicate that this particular execution will require meteorological data for a period of 120 hours, in one hour steps, beginning on a specific day and time, from a particular run of a mesoscale model. The user GUI interaction would provide an unambiguous dataset name (like: “RPS-1.0_run#21_jan26-jan30”). The database server would first locate the dataset as described in Section 5.3.1.1.

Then the database server would determine the following factors:

- if the datasets will be local to the computing nodes (locale for model runs will have been previously negotiated between the CMS client and the CMS management server),
- if they will not be local should the dataset be moved or copied to another site, and
- if the dataset can and should be segmented into subsets for this particular run.

The database server sets up links to this database which will be used by the models, clients, and other servers during the user’s session. During that session, I/O references will be redirected to the appropriate database interface to access the data. The major implementation effort for these functions will rely on the facilities of the DFS core server furnished as part of the DCE.

5.3.1.4 Conversion. Since the CMS will integrate a large number of existing (“legacy”) programs for models and ancillary programs, it is clear that many of the programs will be “non-conformal”. That is, the data output from one module will not conform to the input specifications of another. A simple case might be where one module produces its results in 64-bit floating point format while a possible user function expects the file to be in 32-bit single precision, floating point form. (The very common conversions of formats amongst Cray or VAX or Intel or IEEE Standard will be automatic, invisible, and performed at a level below the CMS design.) Several options are available to the system.

- Convert the entire file from 64 bit to 32-bit floating point prior to executing the receiving module(s).
- Convert the data as it is input to the receiving module(s).
- Establish a standard whereby all stored data is in a single, common format and provide conversions during writing and reading of files.

This latter option has an intrinsic charm, since all stored data would be in a canonical form. Unfortunately, this would require using the largest storage mode used by every function and would consume unacceptable amounts of data storage and impose unnecessary conversion penalties on many programs. The best solution for all future CMS modules would be to have all stored data be in “self describing” form and all I/O routines in the new programs would recognize these descriptions and perform conversions when necessary.

5.3.1.5 Transformation. When integrating different models, a major complication arises when the datasets are “non-conformal”. This happens, for example, when the grid topology and resolution needed by one model for input do not match the structure in the output file produced by the model providing this data. While this situation exists for many of the “legacy” models, it will necessarily arise even in newly developed models. The database server will maintain information about these data “form factors” and provide facilities for transformation of the data from one schema to another, either during actual model input/output or by creating a new form of the database in the transformed state.

5.3.2 Communications

The CMS database server initiates communications with the CMS clients and other servers, as shown in Table 4-1. The communication are summarized below. (Note that communications to the CORBA, the batch server, and real-time server are not applicable.)

- **Communication to the clients.** Transmit GIS data and extracted, converted, or transformed data.
- **Communications to the CMS management server.** Report errors.
- **Internal.**
- **Communications to the DFS.**
 - Full data files
 - Data subsets
 - Requests for data and subsets

- **Communications to the model servers.** Requests for data extents, boundaries, and formats.
- **Communications to the models.**
 - GIS data
 - Transformed, converted data, and data subsets
- **Communications to the DCE.** Requests for locale and access of the various DFS.
- **Communications to the decision server.**
 - GIS data
 - Transformed, converted, and subsets of data
- **Response to requests from CMS clients and servers.**

5.3.3 Concurrency

The database server must be able to provide all of its functionality to a number of asynchronously operating clients and servers. There can be several instances of the transformation function in operation, for example, in support of a single client.

5.3.4 Performance

More than any other server, the database functions can easily become a bottleneck for the system, unless there is sufficient computer power and data bandwidth present in the host computers.

5.3.5 Operation

Given the envisioned design of the system – a data-centered system – the database server is critical for acceptable performance. In this case, the database server would consist of both the necessary hardware and software to provide the data accessibility and flow necessary to support the CMS. An important part of the database server is the GIS component which provides the user with the ability to relate the predicted meteorological, emissions, and air quality fields to geography, county and city boundaries, highways, population, topography, land use, and so on. All required inputs for the various models (including topography, soil type, geopolitical boundaries, land use, vegetative cover, and soil moisture) will be accessible from the database server at the highest potential resolution.

The user will be able to take a vertical profile at any point in the modeling domain and time and know the soil type, vegetation, soil moisture, elevation, county, nearest town, latitude,

longitude, solar elevation angle, soil temperature profile, meteorological profile from the surface to the tropopause, and similar profiles for relevant air quality parameters. The GIS component is tightly integrated with the database server to provide this functionality throughout the system.

5.4 The DCE Distributed File System (DFS)

Figure 5-4 displays the major components of the DFS which uses DCE services to provide interfaces to the remainder of the CMS. The DFS is based on the Andrew File System which was developed at Carnegie Mellon University. While the DFS itself is not being designed by the CMS team, it is a fairly new concept and fundamental to the CMS infrastructure. For this reason, the DFS will be described here in some detail and in the same format as the remainder of the CMS design.

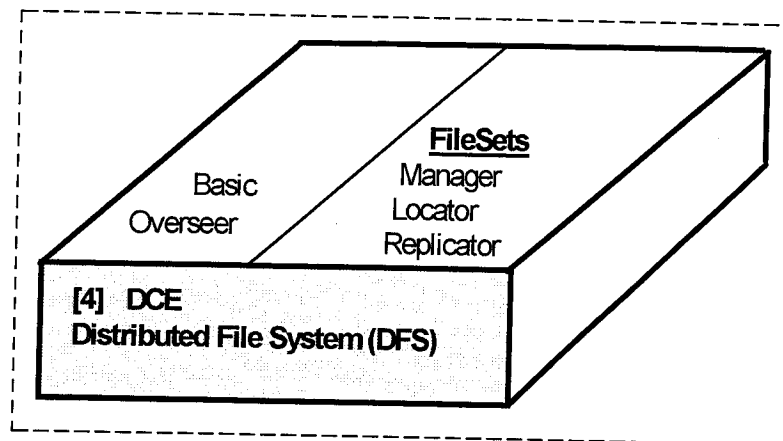


Figure 5-4
The DCE Distributed File System (DFS)

5.4.1 Components

The DFS actually resides in each element of the CMS, unlike the depiction in all of the figures in this design document. There are one or more DFS servers in any system. Any host which needs access to files managed by the DFS will have a small client function operating in conjunction with the operating system's input-output (I/O) functions. Unlike all of the other DCE and CMS elements, this DFS I/O function (the "cache manager") has to be installed within the operating system, instead of operating as a 'layer' on top of the operating system.

We discuss below the three components of the DFS: filesets, cache manager, and the basic overseer.

5.4.1.1 Filesets. From the user's perspective, the DFS offers flexible file naming conventions for all their files and eliminates the need for location information that is commonly encountered in Unix and PC-based systems. For example, the user may have a file containing the input fields for a dispersion model which they have chosen to name

“Dispersion__Data_1-13-95”. Even though this file might be unique in the system or for this user, the user usually must provide a “full path name” which includes additional information about the file’s locale. On a PC this might become:

d:\userhome\application\Dispersion__Data_1-13-95

and similarly on a Unix system it might be:

/user/username/home/application/Dispersion__Data_1-13-95

In a system based on DFS, the name “Dispersion__Data_1-13-95” would be sufficient to identify the file. The DFS would take care of determining the actual location and establishing the necessary operating system links to those files on the requesting host systems.

In a non-CMS environment, the term “fileset” is used to describe the entities which the user needs to access, since in many cases more than one actual system file may make up the requested item. This is particularly true for large, complex modeling systems where the input and/or output data may be larger than any single physical space may provide. In these situations, the user would have to subdivide the data because of system constraints and not for reasons related to the needs of the models or the science. In a DFS environment, the user is not constrained by such artificial boundaries and may consider model setup and execution independent of the physical subdivision of their data. A DFS server provides three main functions for filesets: management, location, and replication as further discussed below.

- **Manager.** The management function is principally concerned with the integrity of all files under its control and the efficiency of storage and access. One of the major achievements of DFS has been its “coherency”, unlike other systems, such as NFS. Model developers on Unix systems have had to cope with the problem that NFS file data being read by one process may not reflect the latest data actually written by another process. This lack of coherency has led to some remarkable chaos when running a sequence of models simultaneously. The DFS manages operations on all of its files through a mechanism which utilizes process-to-process and host-to-host “handshaking” using a software device called “tokens”.

In Figure 5-5, Process B is writing data to a fileset which is being read synchronously by Process A. In typical Unix systems, the user performs a write operation which moves data from the user space to a Fortran I/O buffer. The Fortran I/O routines will make system calls to write this data to disk at times which may be asynchronous with the user I/O calls. Of course, the system does not immediately write this data to the physical media. Instead the data is further buffered in the memory of the host computer until some operating system criterion is met: whereupon it is actually sent to the disk file. On NFS systems

where the disk is on a remote machine, the data are again buffered by that host before actually being written to magnetic storage.

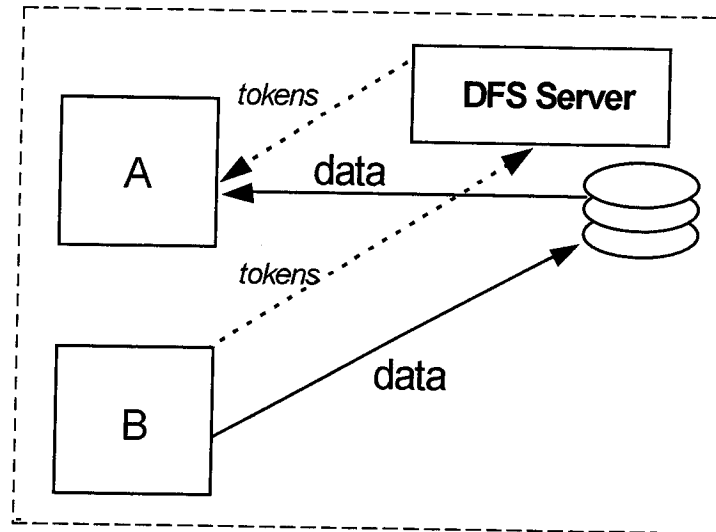


Figure 5-5
Use of "Tokens"

In this example, if Process A attempts to read "the latest" data which has been written by Process B, the information delivered could reflect an older record/time step, or even invalid data. The DFS server, on the other hand, uses a set of tokens and protocols to keep the data for both processes synchronized. When data is moved to the system buffer (or cache) by Processor B, a token interchange occurs between that host and the DFS server even though the actual data may not yet have been transferred to the file systems. When Process A attempts to read that record/time step, the system first looks in the buffers/caches of that host to see if the data is already there. If not, a request token is sent to the DFS server which may discover that the data has been "written" by Process B, but not yet delivered to the actual files. The DFS manager will then contact Process B and "demand" that the data be flushed to the file system and will not send the appropriate record/time step token to Process A until the data is ready.

The "tokens" referred to here are packets of information which include a globally-unique user id (in contrast to the user id for a single host or group of hosts) and time information. These tokens can then also be used to provide authentication credentials for accessing the data, when DCE/DFS security is enabled.

- **Locator.** In the filesets section, we discussed the concept of "global" or "location independent" file names. Obviously, some agent in the DFS must be able to relate these filenames to the actual file or files they reference. The locator function provides this facility and returns to the DFS client sufficient information to provide linkage to the requested data for all subsequent accesses.

- **Replicator.** The DFS manages its files with internal information which is considerably more robust than Unix or PC file systems. A feature of this mechanism is the ability to copy or move files about a distributed system without actually moving all of the data. If, for example, users wish to make their own working copy of a large fileset, the copy function contacts the replicator function of the DFS which creates a set of internal pointers to the actual file information. If the user modifies some of this dataset, the modifications are placed in a separate file area and the corresponding pointers are updated. With this approach, only one copy of the unmodified data exists, potentially saving not only storage, but the computer time necessary to transport the potentially large volumes of data. The replicator can also update the unmodified file with cache data.

5.4.1.2 Cache manager. The cache manager module provides the interface between the DFS client and server functions. It operates within the host system just like the Network File System (NFS) commonly found at most Unix sites. If a host deals with NFS files as well as DFS files, the cache manager can operate in parallel with the NFS on that host, if necessary. The main purposes of the cache manager is to maximize file transmission performance and to manage the DFS token interaction to maintain security and coherence in the system.

5.4.1.3 Basic overseer. Every significant subsystem in a distributed environment possesses a function which manages all the other facilities of its ilk. The DFS core management could be called the "basic overseer". This function is parallel to the DCE and CMS manager and provides similar assistance. The manager starts all other DFS processes in its "cell" and ensures that these processes continue to be active and in communication with the DFS server.

5.4.2 Communications

Virtually every task which accesses data within the CMS will communicate file requests to the DFS server. Once a fileset is located and opened, the majority of data transfers occur between the lower level functions, minimizing further overhead. As hunks of the filesets are moved between the DFS clients and servers, there is a constant stream of token information between those two agents to maintain the levels of "coherency" and security chosen for the CMS.

The communications are summarized below (note that communications to the batch server and the real-time server are not applicable).

- **Communication to the client.** The DFS server will transmit data and tokens to clients as they request fileset information.
- **Communication to the manager.** The DFS sends error information to the CMS management server. In addition, the CMS manager may request

regular updates on storage space and fileset allocations which it may need during negotiations in the event of competition for system resources.

- **Communication to the database manager.** Data and tokens
- **Internal.** A major source of information interchanged within the DFS server is the pointers and locator information which results from file replication and file movement requests.
- **Communication to the CORBA.** If possible the entire CORBA database will be maintained within the DFS. Data and tokens will be sent to the CORBA when requests are made by that system for objects and object information.
- **Communication to the model servers.** Data and tokens.
- **Communications to the models.** Data and tokens.
- **Communication to the DCE.** Re-authentication requests.
- **Communications to the decision server.** Data and tokens.
- **Response to requests from clients and servers.** The DFS server returns fileset pointers from locator requests by CMS clients and servers. Requests to retrieve and store data are serviced through the cache management layer of the DFS.

5.4.3 Concurrency

A DFS server will normally be dealing with numerous CMS clients and servers at the same time. All of these activities are necessarily asynchronous and parallel. These characteristics imply the need for multiple processes and some multiple threads within the DFS.

5.4.4 Performance

The evolution of the DFS from the AFS has aimed at improving the data transfer performance and data security and integrity of the filesets. The CPU platform and the network interconnection with CMS clients and servers must be the highest in the system within the economic constraints of any given CMS implementation.

5.4.5 Error Handling

The DFS possesses a number of error handling facilities and recovery and backup of critical filesets can be accomplished within the DFS. Extensive error reporting is provided to the CMS manager and to the requesting CMS client or server. Error

reporting includes the tools for tailored recovery techniques and, when all else fails, for the “graceful” termination of processes.

5.4.6 Implementation

The DFS is implemented in the “C” language with IDL header files which create a framework which is very similar to C++. This approach was taken with the DCE and DFS developments to use the native “C” compilers on the widest array of platforms, as C++ systems were still in their infancy on some vendor’s machines at that time.

5.4.7 Operation

To illustrate the operation of the DFS, we will use a simple scenario which might occur during a model setup and execution, as described in Section 2.4.

- **Client fileset selection.** Once the CMS client has been logged in to the CMS and authenticated for access to the DFS and the database manager, the user will wish to obtain a list of the filesets available for input to the model of interest. The first step is for the client to retrieve the “fileset type” from the model server. Then the database manager is asked to provide a list of these filesets. This list appears in the client GUI window and the user makes one or more selections.

The client then requests access to these files from the DFS. This involves the sending of requests to the DFS locator. This process ensures that the filesets are actually available in the current CMS, that the user is permitted to access the files, and determines the location of these filesets.

The user in this scenario, interacting with the client, determines that the files may be left in place where they are physically situated and that the data will be accessed over the network from the CPU that is executing the model.

The full CMS name for these files is then passed on to the models to be used to open files, read and write data, and close the files.

- **Model reading data.** The model receives the filenames from the CMS client and opens the files in the normal Fortran manner. These open functions are passed to the DFS client which turns them into DCE RPC which invokes the DFS server open functions. If CMS security is enabled, the requests are accompanied by authentication information. The cache manager on the host running the models will make read requests for ‘file chunks’ as the local cache becomes depleted. Again, each of these requests may contain further authentication information. In some cases, this authentication information may contain time-stamps. Some data may be marked “stale” after an elapsed time in the DFS cache and will have to be “refreshed” by the cache manager. This operation is invisible to the user. If the model is reading data produced

by other processes which are being executed simultaneously with the model, the data requests from the model may not be fulfilled immediately. In this case, the DFS I/O operations will be blocked. The model execution itself will not be blocked until it has exhausted the particular DFS cache.

- **Model writing data.** The model writes data using Fortran, C, or C++ I/O calls exactly as it does in today's Unix or PC environments. The DCE/DFS system turns these calls into RPC at the time of compiling and linking. As data are produced by the model and a "write" operation is performed by the program, the output is first cached on the host executing the model. As the cache manager determines that sufficient information has been accumulated for efficient transfer to the DFS server, actual communications are established with a DFS server. If no demands are being made for the model output, this process continues until model termination. If, however, the model output is being read by another process, the cache manager may be told to flush its information to the DFS "immediately". If the fileset "hunk" is available, this will be done even if the data segment size is not optimal for network transfer. (If properly implemented, we do not expect this to cause a bottleneck.)

5.5 The CMS Model Server

Figure 5-6 provides a block diagram of the CMS model servers and their relationship to the client and the models themselves.

5.5.1 Components

The CMS client possesses no inherent knowledge of the models it will invoke. Model parameters, input and output data requirements, etc., must be provided by the model server. To accomplish this, the model server 1) must have knowledge of the presence of models in the system; 2) must provide the CMS client with GUI instructions; and 3) must possess the resource requirements for each model. These three components are discussed below.

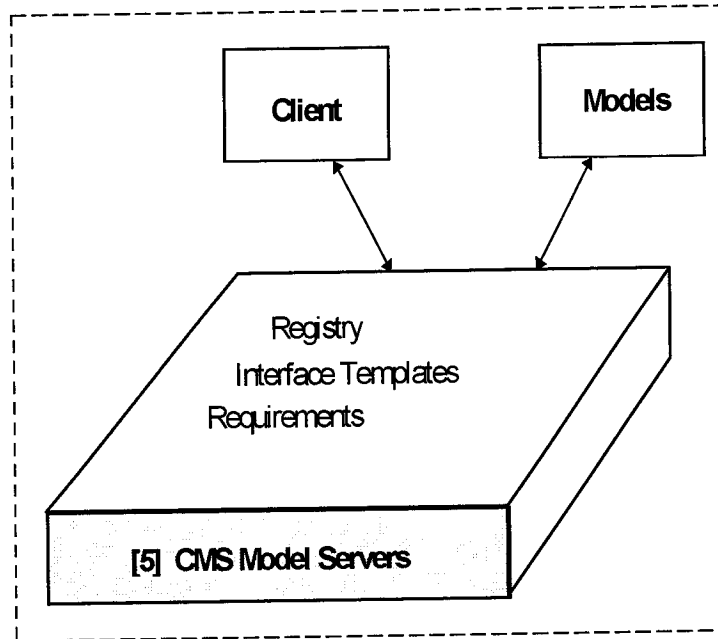


Figure 5-6
The CMS Model Server [5]

5.5.1.1 Registry. The model server registry function is a GUI-based module which is used by model developers and the CMS administrator to identify a model to the CMS and provide all the information needed for client setup and execution of the model. The CMS client function will request a list of models of a given type. The model server will provide this data to the client based on the client's authorization to access one or more of the models.

5.5.1.2 Interface templates. From the model server registry function, a set of templates are created and stored for each model. When the client selects a model, it requests these templates which are used to create and guide the user's interaction in the process of model setup.

5.5.1.3 Model resource requirements. In addition to the GUI templates, the model server maintains a database of data formats and means of forecasting filesizes for the execution of each model. In addition, this database contains information on the computational and I/O bandwidth requirements for each model. These data are provided to the client during model setup to aid in determining the appropriate locale for the model execution. The CMS management server may request these data when confronted with the need to arbitrate between two or more clients competing for these resources.

5.5.2 Communications

The principle communications of the CMS model server will be with the CMS client, since that agent seems to be the natural resource on which to run the registry process,

as well as the model setup procedure. However, newly-developed models may take advantage of the database maintained by the model server to establish their most optimal input and output formats for data on a dynamic basis. (In many cases, the model itself is a more efficient place to perform some data transformations, than to require an intermediate file “conversion”.)

The communications are summarized below (note that communications to the database manager, the batch server, the decision server, and the real-time server are not applicable).

- **Communication to the client.** Transmits GUI templates for the particular model. Transmits data on resource requirements for the model.
- **Communication to the manager.** Transmits model resource requirements, when requested. Sends urgent error information or error log data for non-crucial errors.
- **Communications to the DFS.** The model server may operate on two or more platforms and hence, will need to share data through a mounted file system. All registry, template, and resource data will be written to and retrieved from the DFS.
- **Communications to the CORBA.** The model registration process for CMS will produce information which will be sent to the CORBA. The principle elements will be the parameterizations, their formats, and contents for the models which will be used within the CORBA to create appropriate “stubs” for other processes when they wish to link to the models.
- **Internal.** There is a need for the three functions to share the directories for models, as well as the template and resource databases. This can be accomplished through file sharing and will not require inter-process communications for these functions.
- **Communications to the models.** The model server will request dynamic information from all newly-developed models during the registry process. While most of the information could be obtained from the user via the interactive registry session, model version and resource requirements (such as CPU memory demands) are best drawn from the models themselves to minimize user intervention to the maximum extent possible.

- **Communications to the DCE.** Authentication requests for user/model combinations.
- **Response to requests from CMS clients and servers.** The model servers all accept requests for service only from the CMS clients and from the CMS management server.

5.5.3 Concurrency

Except when attempts are made to perform create, update, or delete operations on a model's registry, all functions of the model server can be operated in parallel and with multiple instances of the function.

5.5.4 Performance

The model server registry, template, and resource requirements retrieval must be sufficiently responsive to sustain the interactive nature of the client GUI session. The resources required to do this are minimal. However, since the function can be replicated on multiple platforms, there is no apparent performance limitation foreseen regardless of the host CPU capacity.

5.5.5 Error Handling

The model server will rely on the error detection and recovery facilities of the DFS for the bulk of its own error handling. Internal consistency checks (such as, a model itself being absent from the object database while it appears in the registry) will be performed on a periodic basis and error reports sent to the CMS management server.

5.5.6 Implementation

The CMS model server will be one of the unique facilities in the CMS and will have to be designed and developed by the CMS development team. It will utilize all of the system functions, such as the DFS, and will be implemented with object-oriented programming technologies.

5.5.7 Operation

There are two distinct modes of operation of the model server: 1) model registration; and, 2) model information retrieval. We will discuss a simple scenario for each one, involving an admittedly simplistic model.

- **Registration.** For this example, we shall presume the existence of a simple diagnostic wind model which was written many years ago. For practical purposes, it has been decided that the model must be used "as is", without making any effort to modify its source code (a classic "legacy code"). The CMS administrator, or a CMS developer with appropriate authorization, would

log in to a CMS client and invoke the model registry function. Since this is a new model, they would select that function from a menu.

- A model “class” would be identified: air quality, meteorological, emissions.
 - Model name and version number: supplied by the user.
 - Since this was a “legacy model”, the user will provide a description of the input parameter types, ranges (if applicable), and defaults (if any) for the command line invocation of the model.
 - For each of these parameters, the user can select a GUI menu, dialogue, or other interface to be used when the model is being setup. This can include the invocation of an alternative GUI-based database creation. In this case, this user would elect to start a 3D graphical system during model setup. This interface would allow the user to establish geographical coordinates, grid layout, and perhaps, boundary values for each run.
 - The location of the binary(s) for the compiled and linked model are given by the user and they are moved into the CMS object database.
 - The user is asked to provide ranges of the expected sizes for each dataset which is designated by the input parameters. Information on CPU and network requirements are also provided by the user.
 - The GUI templates and the resource data are stored in a DFS database for the model server.
 - The administrator can create an access control list (ACL) for the model, giving explicit permission to selected users to access the model.
- **Model characterization.** When a user logs in to a CMS client to setup a model, they are presented with a GUI menu from which they select the model type. The system begins the process of interactively assisting the user in model setup and execution.
 - The model type information is sent from the CMS client to the model server.
 - The model server retrieves the list of models of that type and returns to the CMS client the names of those models for which the user has access permission. In addition to the names of the models, a brief narrative profile of the models is supplied to the CMS client.

- The user makes a selection from the list.
- The client requests the model characterization information from the model server.
- A series of GUI “templates” are supplied to the client. These templates present the user with menus, selection boxes, and other schema to acquire the information needed to setup and execute the model.
- In this example, one of the “templates” invokes the 3D graphical process to interact with the user to define the locale and the grids needed for this model.
- Once the parameters have been established, the user must determine where the model will run.
- From the database requirements and locale of the model, the user is asked to choose a number of options prior to starting the model. The input dataset may be moved to the selected CPU host or the user may choose the model to perform its I/O to one or more distributed filesets.
- When the model is finally invoked, a log entry is sent to the model server giving the user id, start-time, and location of execution for possible diagnostic and statistical analysis.

5.6 The CMS Program/Object Server (Object Management Mechanism)

Object management mechanisms are a software construct used to make it easier for external programs to interact with a set of procedures and data. This is done by providing some degree of encapsulation (see Appendix B) . A common example is the OLE System from Microsoft. This is a rudimentary object management mechanism used to exchange data between Microsoft Windows Programs via the clipboard. It provides a means of packing the data along with a description of the data and how they should be read. This allows a user to ingest a set of database data into a spreadsheet, for example. A more sophisticated example might be the use of the SOM Object Management System in OS/2 to allow the user to “drop and drag” a document icon onto a printer or fax icon. Here the objects contain both the data and the routines (or methods in object-oriented lingo) used to access the data. The common object resource broker architecture (CORBA) is a more sophisticated object management mechanism developed by the Open Management Group. It is intended to be a platform independent system. As its name implies, the CORBA provides a repository and a brokerage “service” for all the objects under its control. This “service” ranges from ensuring proper associations between objects which are activated by user requests, as well as managing the necessary interlocks and registries necessary for system

coherence. For example an object may be a temporal database which needs updating by one process while yet another process needs to access the same object. The CORBA might block either requester until the other one is complete or it might initiate a copy of the object so that both requests can be satisfied at the same time. It is our plan to utilize CORBA in the development of the CMS. We must be aware, however, that industry agreed upon standards often get swept away by competing commercial entities; especially if the competition is Microsoft. In this case the competition is between CORBA and a newer, more robust version of OLE being promoted by Microsoft. The CMS development team will need to remain flexible on this issue and see how things play out.

In the CMS the object management system will be used to encapsulate interactions between models, functional modules (like advection), and the data sets they utilize. For example, consider a modeler assembling a set of computational modules (advection, radiation, etc.) into a simulation system specifically tailored to his needs. The modules would be objects and the CORBA would be used to manage the interactions between the modules. At a higher level, one can imagine an icon representing the observational meteorological data for a particular day being dragged onto a RAMS icon to initiate a simulation. Of course running a meteorological model (and maintaining a group of models) is a much more complicated process than printing a document. The process must allow for numerous user choices of parameters. It may not make sense to embed this much intelligence into the general purpose object management mechanism. For this reason, we have introduced the model server into the system, as described in Section 5.5.

The distinction between the CORBA and the DCE environment must be made clear. These are not competing systems. The DCE provides a set of low-level capabilities (essentially an extension of the OS) in a networked environment, such as: file access, time synchronization, directory services, remote process control. The CORBA operates at a higher level, i.e., with application oriented data structures rather than at the OS level. It utilizes the underlying DCE in a transparent fashion. Some semantic confusion on this issue may arise from the fact that the DCE provides capabilities for allocating DCE server resources and services among competing nodes (CPUs), which has been called "Resource Broker" by the Open Software Foundation.

Figure 5-7 presents an overview of the common object request broker architecture (CORBA) as defined by the designer's architecture and specification document. As the CMS develops into a fully featured and robust system, its hallmark will support dynamic growth in a variety of models. The evolution of the CMS from its first prototype to a truly comprehensive modeling system will finally require a sophisticated system for managing all of the pieces of the CMS in this dynamic environment.

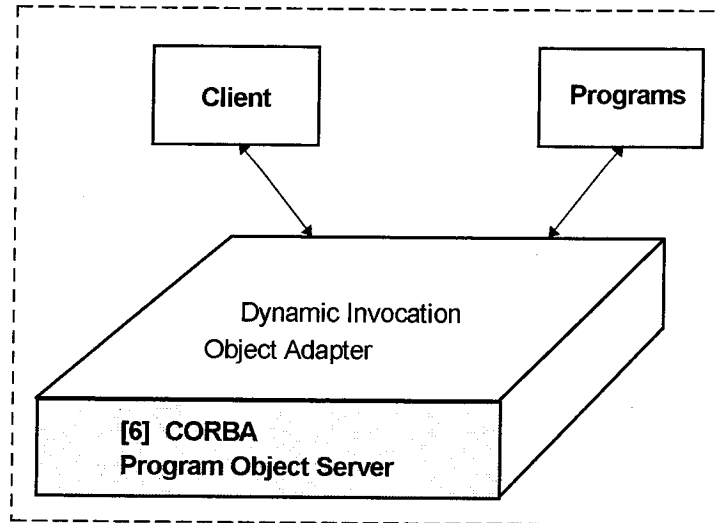


Figure 5-7
The CORBA Program/Object Server [6]

5.6.1 Rationale for the CORBA

The rather involved block diagram displayed in Figure 5-7 masks a rather simple principle cited several times previously – “Plug and Play”. We want to have, in the CMS, the ability to interchange components at several levels of the system with as much transparency as possible. In the best of all possible worlds, this would extend to interoperability among subsets of the CMS and other systems, such as Models-3. There are three principle opportunities for this kind of interoperability as illustrated below.

The simplest mechanism for inter-system interoperability is presented in Figure 5-8. It provides data for use of both systems in a clean, interchangeable manner. The approach to this would be to establish a common format for all files and a common, shared storage system for this data. Use of widely held standards in the implementation of the system is one of the key CMS design principles. The ultimate goal here is to make this process almost automatic in the future.

In Figure 5-9, the concept of inter-system interoperability is extended to include the interchange of models or other major components of the system. One of the first steps in bringing the CMS to reality is to make all of its components interchangeable. If this can be done in accordance with modern system standards, then it can be hoped that all other systems such as Models-3 will be able to provide subsystems which will conform to these standards with consequent interoperability opportunities.

5.6.2 An Idealized Object-Oriented System

The diagram in Figure 5-10 describes the idealized situation which our long-range planning for modeling systems should be targeting. In this “best of all possible worlds”,

models and model support software would consist of well-crafted scientific modules which could be used in any one of many models depending on the requirements of a specific user. A necessary step toward this goal is the establishment of a standard framework for model and model component development and exploitation. The CORBA provides this structure and the guideposts needed for future model development and integration.

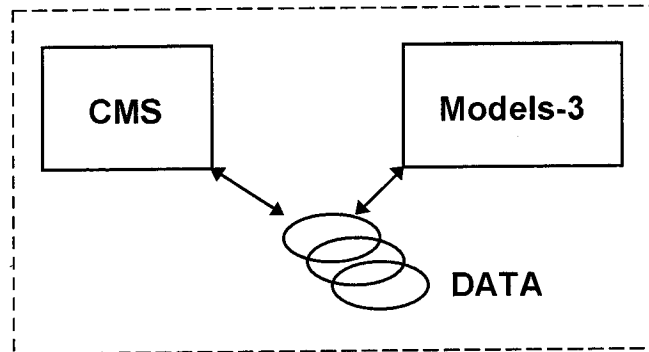


Figure 5-8
Inter-System Interoperability

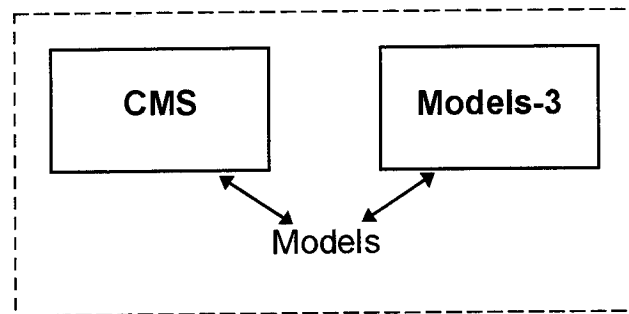


Figure 5-9
Inter-System Interoperability

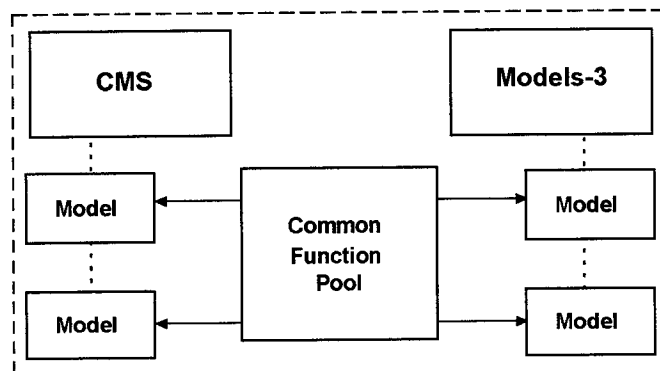


Figure 5-10
Fine-Grain Interoperability

5.6.3 The Basic Object Broker Task

A usage of Figure 5-7 can be seen in Figure 5-11, where an application program makes a subroutine call to a major function, “advection” for example. This function may be implemented in several different ways, different physics, different grid structures, or different forms of parameters. Future CMS applications will partition the functional elements into “objects” (see “Object-Oriented Design” in Appendix B). These objects will be catalogued by the object broker. Invocations of these functions will be possible through both static and dynamic means. In either case, the object broker locates the desired functional object and provides automatic (and thus, transparently) conversion of parameters, not only mathematically, but perhaps, in terms of coordinate system transforms.

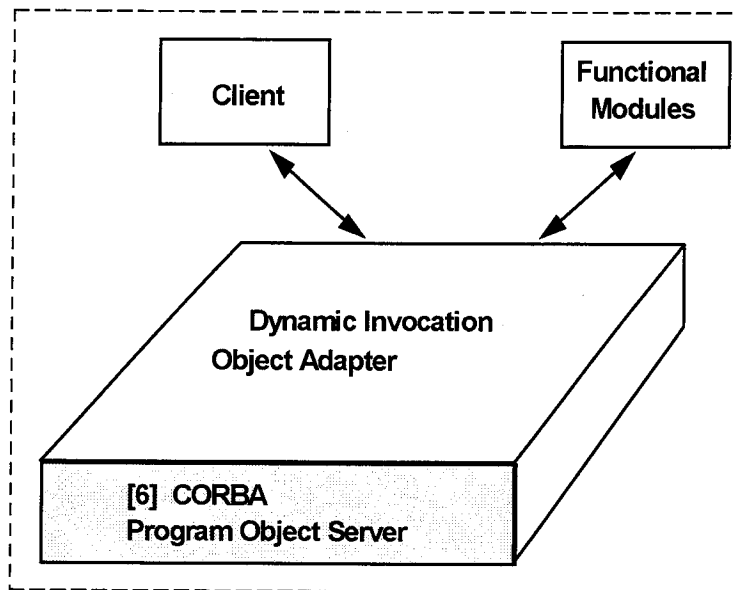


Figure 5-11
Illustration of a Usage of Figure 5-7

5.6.4 Standards

The most important means for CMS to maintain a high level of systems technology throughout its life is to rely on the use of standards in every aspect of its development and operation. Interoperability and interchangeability are encouraged through the philosophy of “open systems design”. An “open system” is one that contains no proprietary components and which is intended to share software with other systems. Since this is the premise for the Open Software Foundation (OSF), the employment of the DCE from the OSF is a major first step. As the CMS becomes more formally object-oriented through new development, the evolving CORBA will become the basis for an even more flexible and open system.

5.7 The CMS Models

Figure 5-12 displays the main group of models to be used in the CMS and their interaction with the client, database (manager and DFS), and the CORBA. There are two major tasks to undertake in the area of models for the CMS, as discussed below.

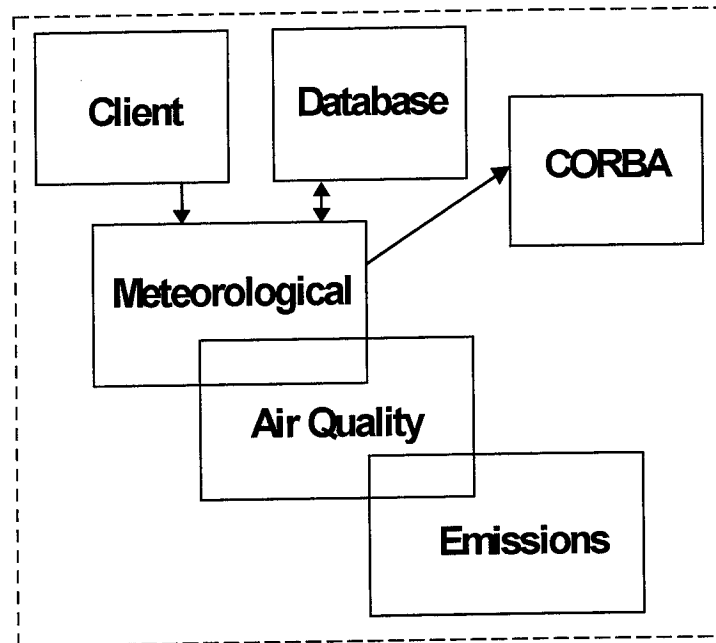


Figure 5-12
Meteorological, Air Quality, and Emission Models [7]

5.7.1 Task 1 – Existing Models

The CMS will initially incorporate a collection of existing models, most of which have been developed independently of one another. Adaptations of these models through “wrapping” will then be employed to provide simple interchangeability, wherever possible. In the Prototype #2, demonstration of two models which have not been run in concert before will be adapted in this manner. The design goals for this part of the project are to provide similar “wrapping” for a few of the models listed below.

- **Emissions models**
 - JEWEL (CMU)
 - Models-3 modules
 - FREDS
 - EPS
 - EMS-95

- **Meteorological models**
 - RAMS (Colorado State University)
 - MM5 (Penn State)
 - ARPS (University of Oklahoma)
 - HERMES (ARIA Technologies)
 - DWM (SAI)
 - MATHEW (LLNL)

- **Air quality models**
 - CIT/URM (CMU)
 - SAQM (see SARMAP in Appendix C)
 - ADOM (Environment Canada)
 - UAM-V (SAI)
 - CALGRID (California ARB)
 - MONTECARLO (FaAA)
 - AVACTA II (AeroVironment)
 - MESOPUFF (EPA)

5.7.2 Task 2 – New Models

A key design task for the CMS will be to establish guidelines and standards for future model developers. These will prescribe methodologies wherein future models and their principle components are conceived of as objects from the outset and programmed with modern Fortran-90 or C++ techniques and practices.

5.8 The Distributed Computing Environment (DCE)

Figure 5-13 presents a block layout of the DCE, which has a number of layers, much as does the CMS. These layers may be accessed directly by clients and servers. The DCE provides industry-wide standard means for controlling and accessing computing resources which may be distributed across administrative, organizational, and geographic boundaries. The DCE consists of a number of functions which reside on each CMS machine and are designated DCE client processes. One of the hosts in the system performs the functions of DCE server.

We expand the discussion below of DCE client, directory of services, security services, and time synchronization.

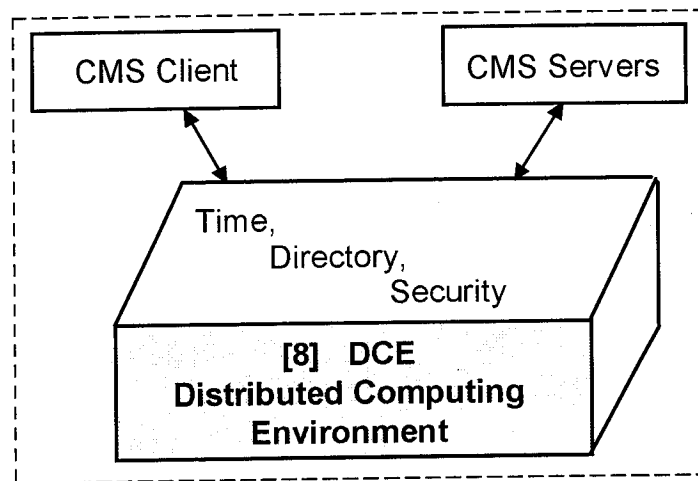


Figure 5-13
Distributed Computing Environment (DCE) [8]

5.8.1 DCE Client

The fundamental building blocks of the DCE are two programming concepts: “pthreads” and “remote procedure call” (RPC). While the general user may ignore these facilities, they will, in fact, be operating in concert with the user interfaces, the user’s programs, and the many systems programs which make up the CMS. In Section 2.3.1, the RPC concept was introduced as simply another form of implementing a subroutine call for a Fortran, C, or C++ program. “Pthreads” (or POSIX threads) are only of interest to the system programmers of the CMS. A “thread” is one of the ways in which separate program executions are managed on today’s Unix, OS/2, Windows NT, and Windows 95 operating systems. The POSIX standard for threads has provided an extensive set of functions (54 functions) for creating and managing these threads of control. The abbreviated title “pthreads” has become the accepted name for this standard.

While internal CMS processes may employ “pthreads” when they are implemented by the CMS team, user models will not require any special handling because of the use of this system control mechanism. In rare cases, a model developer may use a different procedure for compiling and linking their codes in order to incorporate the RPC. The CMS design will attempt to minimize this visibility of the RPC as much as possible.

The DCE client then consists of one or more programs which have been compiled and linked with the hosts software, after a minor amount of “preprocessing” by a DCE function which incorporates a set of “include” files. A continually-running background process, called a “daemon”, provides the physical communication between the DCE programs (on the client) and the DCE server and all of the other CMS servers.

5.8.2 Directory Services

When a DCE client is first introduced to the system, it does not know where any of the DCE or CMS functions are operating. The DCE client does know about the presence of

the DCE server on the network. Therefore, the CMS client contacts that server to obtain permission to use the services of that DCE and the CMS servers which it manages. Once permission is secured, the location or network address of a desired CMS function is then obtained. For example, the CMS client may have a program which is making a function call to open a file through the DFS. Once this network address is provided to the client, all other "open DFS file" calls will be directed to that DFS host, thus bypassing any further directory service inquiries.

5.8.3 Security Services

A DCE-based system can invoke several optional layers of security. As a minimum, the DCE client must be operating on a host known to the DCE server and the user must have permission to log into that DCE client. At the other extreme, the DCE server maintains the only repository of user login passwords and carries on all authentication actions with clients and servers with encrypted messages. Further, any or all data transferred between clients and servers (including DFS file data) can be encrypted with the Kerberos 5 (K5) standard methodology (Kohl and Newmann, 1993).

5.8.4 Time Synchronization

File integrity, system security, interprocess, and inter-host management demand that all elements of the DCE/CMS maintain close time synchronization. For example, file creation times in a distributed system can be a source of chaos if one of the cooperating host machine's clock is substantially behind the clock of the host who owns the file and a user chooses to purge all files older than ten minutes from the host with the "slow clock". DCE/DFS authentication and re-authentication depend on tight time synchronization. Hence, the DCE supplies a master time distribution and update function.

5.9 The CMS Batch Server

The CMS batch server is shown in Figure 5-14. Its primary function is to schedule jobs for execution on one or more computing systems within the CMS. Batch service development for distributed computing systems began with Stirling Software's development of the Network Queuing System (NQS) for NASA Ames Research Center in 1985. Since then, a number of robust systems have become available, in addition to the continually evolving NQS. These systems include the commercially available LoadLeveler from IBM and LFS from LFS Company in Toronto, Canada. Public domain batch servers are "CONDOR" from the University of Wisconsin (which became the basis for LoadLeveler) and the Distributed Queuing System from the Supercomputer Computations Research Institute at Florida State University. All of these systems are mature and have been operating at many sites worldwide. Therefore, the CMS has completed technologies available for this function.

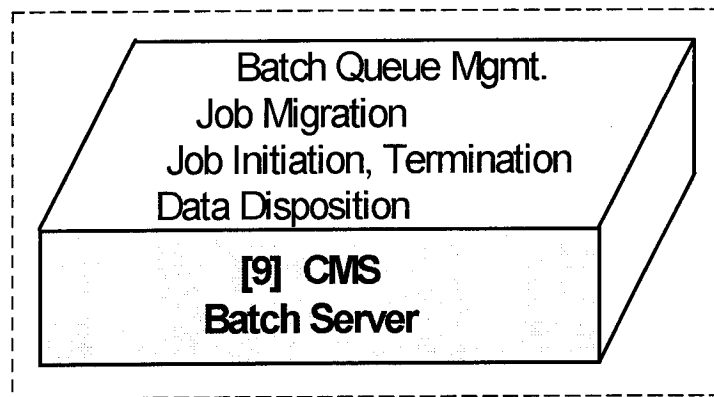


Figure 5-14
The CMS Batch Server [9]

5.9.1 Components

The CMS batch server consists of a master queue “daemon” (background process) running on at least one CMS host, and one or more batch server daemons running on CMS hosts which will provide the actual services. The batch server acts as a “surrogate” for the user and attempts to optimize the resources under its control as well as maximizing the throughput of all jobs submitted to it for scheduling. The major functions of a batch server are:

- Batch Queue Management – maintaining control of batch server daemons , job scheduling
- Job Migration – Moving queued and/or running jobs from one host to another
- Job Tracking – Tracking and reporting the status of all jobs, maintaining accounting information
- Initiation and Termination – starting up and shutting down single and multiple processor jobs
- Data Disposition – copying stdout/stderr, moving output files, removing temporary files

5.9.2 Implementation

All of the candidate batch queuing systems are implemented in standard “C” so that they can be compiled and executed on literally any Unix platform. With the exception of the IBM LoadLeveler, all of the candidates can operate under OS/2 and Windows NT. Windows 95 implementations are currently underway.

5.9.3 Operation

Batch Servers provide two major functions: 1) scheduled delivery of services that would require time consuming resources; and, 2) chores, such as the production of high resolution plots on mechanical graphing equipment, or the production of animations for video tapes and CD ROMS; or production of multiple instances of elements such as hard copy graphic images.

5.10 The CMS Decision Server

Figure 5-15 provides a rough sketch of the CMS decision server and its relationship to the client and database subsystems of the CMS. The decision server will provide a database and GUI templates which will be invoked by the user to assist in proceeding through the many complex decision paths which may have to be traversed to accomplish certain regulatory or scientific functions. The nature of these functions is only understood at the conceptual level at this time. Detailed design of this segment must await the design of the preceding servers and an extensive analysis of user needs in this area. Implementation of this component can be delayed to a later stage of the CMS program, as it is not critical to the overall operation of the system.

The decision support server contains the procedures that would be used in the decision support process. This can include batch, economic analysis, optimization, etc. Output would go both to the batch server as well as the database server.

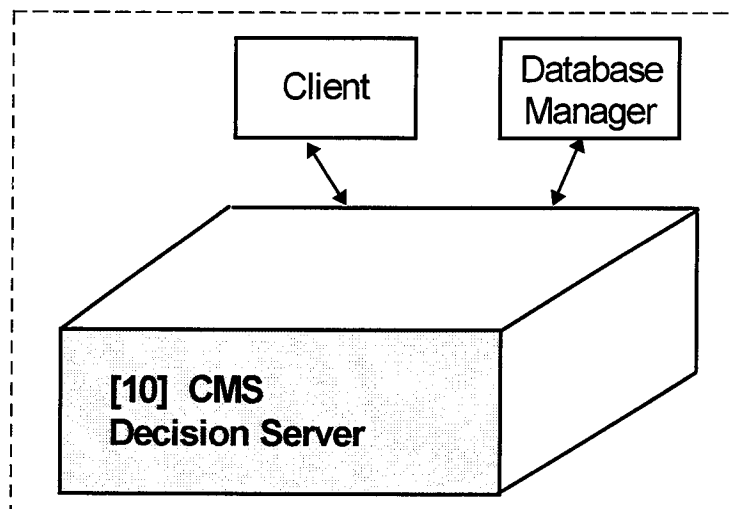


Figure 5-15
The CMS Decision Server [10]

5.11 The CMS Real Time Server

Figure 5-16 provides a general view of the CMS real-time server and its relationship to the client and database systems, as well as remote sensing systems, which may be used in part of the CMS, in the final configuration. The design and implementation of this

component will be performed at a later stage of the CMS when the requirements for real-time data acquisition and incorporation into CMS programs are better defined.

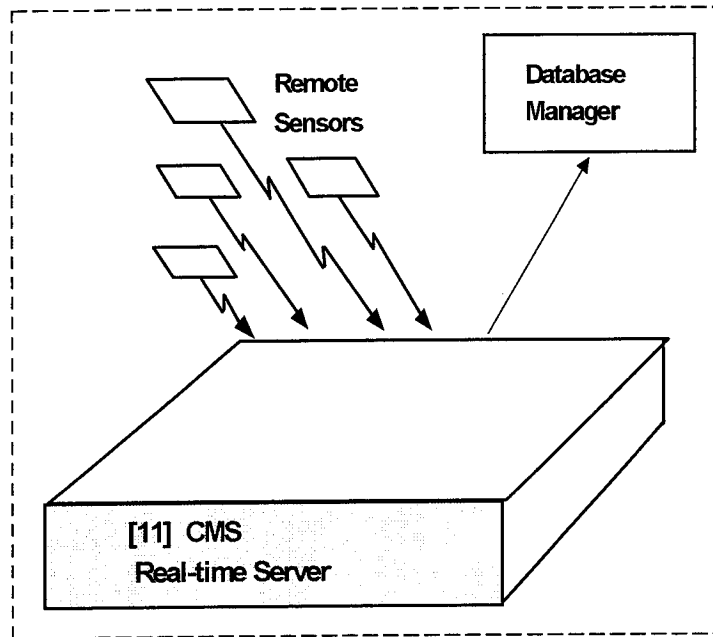


Figure 5-16
The CMS Real-Time Server [11]

6

MANAGEMENT PLAN

The management plan for the development of the CMS is based upon the following key concepts. We provide here some general structure and recommendations. The CMS development team is expected, of course, to formulate and finalize the management plan and adapt our recommendations to fit their needs and budget availability.

- **Management Structure**

The development team will work with a clear management structure and precise allocation of responsibilities. The development effort will be performed by a consulting team of scientists, led by a Project Manager/Principal Investigator (PM/PI) having the overall technical and budgetarial responsibility for the project and reporting directly to CAMRAQ Project Manager. The PM/PI will provide general guidance and supervision and be in charge of resolving conflicts, when required.

- **Subcontractors**

Several subcontractors will work on the project (see Figure 6-1). Each subcontractor group will be led by a Group Leader (GL) reporting to the PM/PI. The GL will be responsible for the technical performance of the group and the financial issues associated with the subcontract.

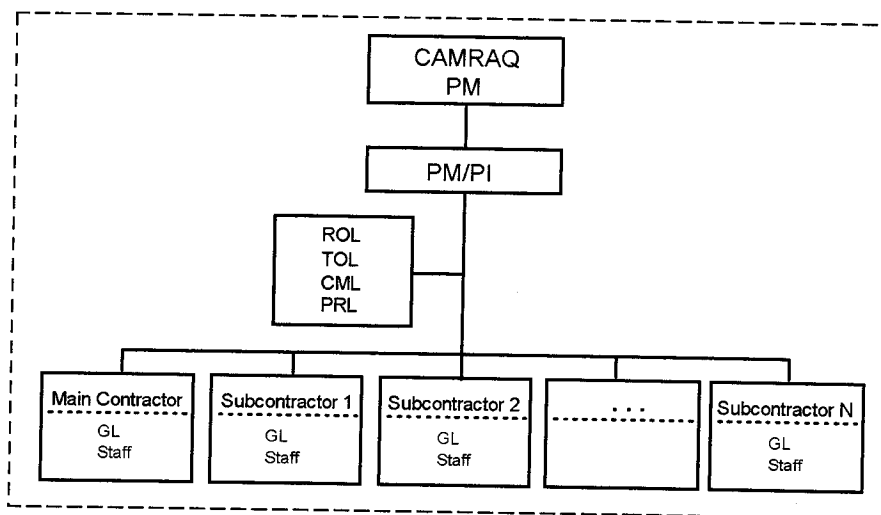


Figure 6-1
Management Plan – Main Contractor and Subcontractors

- **Technical Tasks**

Each project effort (see below) will be subdivided into tasks. Each task will be performed by a “leading” team of scientists, from one or more groups, and monitored by a “monitoring” team (see below). The leading team for each task will be managed by a Task Leader (TL) reporting directly to the PM/PI. The TL will be responsible for the technical performance of the task team and the financial issues associated with the task effort.

- **Matrix Management**

The development team will adopt the matrix management approach. Each scientist working on the project will report to two managers, the GL (who will not change) and the TL (who will change depending upon the specific task in which the scientist is involved). The TL will focus its supervision on the objectives to be achieved in the specific task and the deliverables that need to be provided on time and on budget. The GL will focus its supervision on assuring that resources are properly allocated and that the scientists in its group are performing well as a whole in all tasks in which they are involved.

- **Planned Redundancy and Task Monitoring**

In a project as complex as the CMS, problems should be expected in some of the tasks. Problems may create delays, crises, and some redesign of work segments and even entire tasks. Serious problems may create the need for restructuring task teams or, in extreme cases, rearranging the groups involved in the project.

Problems will never be very serious if they are identified in time. For this reason, the CMS development team will operate with a certain degree of “planned redundancy” and with the use of “task monitoring” teams (see Figure 6-2). Planned redundancy means that two teams will be selected for each task. The first team will be the “leading” team, in charge of performing the actual work. The second team will be the “monitoring” team, but as capable as the leading team of performing the task. The monitoring team will be led by a Task Monitor (TM) who will report directly to the PM/PI. In each task, the monitoring team will dedicate a level of effort of 5-10% in proportion to the development effort of the leading team actually doing the work. This monitoring effort will assure that the leading team is operating in a cost-effective manner, providing results on time and on budget and in agreement with other parallel efforts. If necessary, the monitoring team will be able to provide extra support to the leading team and, in extreme cases, take full control of the task effort.

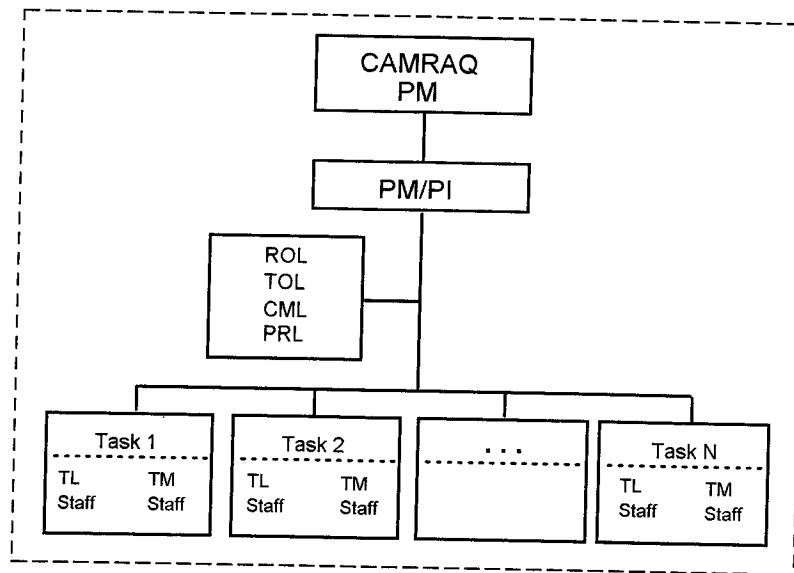


Figure 6-2
Management Plan – Task Management and
“Planned Redundancy” Through Task Monitoring

- **Oversight Leaders**

In addition to the management structure discussed above (PM/PI, TLs, and TMs), the project team will include four “oversight” leaders with the following specific functions (see Figures 6-1 and 6-2).

- Regulatory Oversight Leader (ROL), reporting to the PM/PI with the general responsibility of monitoring the regulatory aspects and implications of the CMS. The ROL will make sure that the regulatory community will be well informed about the CMS and find it useful and applicable to their specific needs.
- Technical Oversight Leader (TOL), reporting to the PM/PI, with the general responsibility of monitoring the technical aspects of the CMS. The TOL will make sure that the CMS remains anchored to the best available science and its technical objectives are properly fulfilled. (If appropriate, two TOLs could be selected, the first for atmospheric sciences and the second for computer sciences.)
- Contract/Subcontract Management Leader (CML), reporting to the PM/PI, with the general responsibility of assuring correct and smooth management of contracts and subcontracts. The CML will also make the effort of minimizing contractual paperwork and allow scientists to concentrate their efforts on technical issues instead of administrative details. The CML will establish simple but effective procedures for contract/subcontract management and progress monitoring.

- Public Relations Leader (PRL), reporting to the PM/PI, with the general responsibility of assuring continuous and effective public relation efforts and external exposure.

The PM/PI will arrange, at least on a quarterly basis, a meeting/conference call with the four oversight leaders mentioned above. To avoid an excessive number of managers/leaders, some scientists are expected to take more than one position. For example, it will be possible for a senior scientist to be TL in one task and TM in another (and even one of the four oversight leaders, if appropriate). We do not anticipate nor recommend the PM/PI to take any TL or TM responsibility, except in exceptional situations (the PM/PI could, however, assume one of the oversight tasks, e.g., the PRL, if appropriate).

- **Communications**

All communications among team members will be electronic. There will be a "preferred" communication system (the Internet), an "alternative" communication system (e.g., America On-Line), and an "emergency" communication system (phone, pager, fax). Formal written communications will be kept at a minimum. On a monthly basis, the PM/PI will provide a one-page progress report to the CAMRAQ PM and an informal project status report to all scientists (a direct communication channel between the PM/PI and the scientists is important to assure that all members are informed about the project and to emphasize openness among the team). Each GL and TL will provide a monthly one-page progress report to the PM/PI. TL reports will also be copied to each corresponding TM. Each TM will provide a monthly one-page monitoring report to both PM/PI and the TL. The PM/PI will arrange, at least on a quarterly basis, a meeting/conference call with all TLs and a separate meeting/conference call with all TMs. Additional communications will, of course, be required if problems arise.

- **Management Tools**

The development team will use the most modern methods and software for project and time management. For project management, all plans will be developed and monitored using Microsoft Project. For time management, the software package In Control will be adopted.

- **Productivity Tools**

The development team will adopt Microsoft Office as the standard productivity tool for report preparation and technical presentations.

- **CMS Development Efforts**

We call Phase II the development phase of the CMS (Phase I was the design). Phase II will consist of the following major efforts:

- Effort #1: Development of Prototype #1 (ongoing; completion expected in February 1996) (This effort was previously identified as a parallel effort, PE, but is actually the first step of Phase II)
- Effort #2: Development of Prototype #2 (Mar-Jul 1996)
- Effort #3: Development of Prototype #3 (Aug-Dec 1996)
- Effort #4: Development of a "basic" CMS (Jan-Dec 1997)
- Effort #5: Development of an "intermediate" CMS (Jan-Dec 1998)
- Effort #6: Development of a "full" CMS (Jan-Dec 1999)
- Effort #7: Periodic maintenance and upgrade (continuous effort from 2000 on)

- **CAMRAQ Management Structure**

CAMRAQ's management structure has evolved as the organization has grown and is expected to undergo further changes as maturity is approached. At present, CAMRAQ is directed by an executive committee composed of those CAMRAQ members who are contributing financially to the CMS design project. Important functions of the executive committee are to set policy for the consortium, set rules and criteria for CAMRAQ membership, approve access fee structures, and approve new members. Currently, CAMRAQ also includes an associate membership class, which is composed of interested individuals and organizations that do not contribute financially to CAMRAQ operations. While voting privileges and electronic access to CAMRAQ resources are currently restricted to executive members, associate members are encouraged to participate actively in all CAMRAQ meetings.

A new membership class, a "subscriber membership," is anticipated in the near future, to accommodate those organizations who desire CMS access but cannot contribute financially to CAMRAQ developmental activities. Under this anticipated plan, subscriber members will pay a smaller log-in and maintenance fee for CMS access and will compose a subset of the associate membership. This anticipated plan also involves formation of a general committee, comprised of members and associate members. This committee will be the primary conduit for communicating to the executive committee community feedback on CMS attributes and capabilities, both existing and desired. It also will formulate, for approval by the executive committee, criteria and rules for CAMRAQ membership, for screening, accepting, and placing visiting scientists, and the schedule of fees for access to CAMRAQ products.

At present, the CAMRAQ coordinator is on contract with EPRI and works closely with the EPRI project manager in the planning and implementation of CAMRAQ initiatives. The coordinator serves as the point of contact for information on CAMRAQ.

7

SCHEDULE AND COST

In the previous chapter, we described seven major efforts for the development of the CMS. The expected cost of each effort is presented below.

Effort	Cost[*] (1995 \$)	Period of Performance	Cumulative Cost (1995 \$)
Effort #1	\$100K	(completion expected in Feb 96)	\$100K
Effort #2	\$350K	(Mar-Jul 1996)	\$450K
Effort #3	\$450K	(Aug-Dec 1996)	\$900K
Effort #4	\$950K	(Jan-Dec 1997)	\$1850K
Effort #5	\$900K	(Jan-Dec 1998)	\$2750K
Effort #6	\$850K	(Jan-Dec 1999)	\$3600K
Effort #7	\$200K	(yearly, from 2000 on)	

In comparison with the costs experienced by previous major model development efforts, the cost estimates presented above may appear quite "optimistic". We believe, however, that these estimates are realistic if the development effort is to be conducted by experienced and dedicated scientists and managed in a firm and cost-effective fashion. Costs can be contained if all members work toward the goal of the project without separate R&D agendas. Project management must be capable of properly directing and supervising the staff, detecting early or dangerous deviations in priorities and goals, and performing corrective action. Under these conditions, we believe that the CMS can be successfully developed, on-time and on budget, with the costs allocated above.

* Costs include the 5 to 10% extra cost due to the planned redundancy and task monitoring activities discussed in Section 6.

8

CONCLUSIONS AND RECOMMENDATIONS

The CMS design team has provided, in this report, a detailed design of the CMS and a plan for its implementation. We emphasized the need for a development plan based upon a sequence of increasingly comprehensive prototypes.

Even though the design proposed here is comprehensive and detailed, many issues are expected to be discussed and redefined by the CMS development team. In fact, computer sciences are evolving at a very fast pace and recent developments, such as the dominant presence of Windows 95 and the availability of Internet-based cross-platform programming languages such as Java, may allow the development team to adjust and improve the development plan in a cost-effective fashion.

In conclusion, while we believe that this document provides a solid basis for the development of the CMS, we recommend the development team perform a constant and systematic re-evaluation of available software tools and hardware products. This approach will ensure that the CMS remains at the cutting edge of both computer and atmospheric sciences and will provide the CMS user with the most cost-effective tools for air quality simulation, analysis, and decision support.

REFERENCES

(this section includes the references listed in the CP)

- Alcamo, J. and J. Bartinicki (1987). "A framework for error analysis of a long-range transport model with emphasis on parameter uncertainty," *Atmos. Environ.*, **21** (10), pp. 2121-2131.
- Bianco, D.J. (1995). "How I Learned to Stop Worrying and Love the DCE", DECORUM-95. NASA Langley Research Center, Hampton, VA.
- Boehm, B.W. (1987). "A Spiral Model of Software Development and Enhancement," *Software Engineering Project Management*, IEEECS, ed. Thayer, R.M., pp. 128-142.
- Brooks, F.P., Jr. (1975). *The Mythical Man-Month*. Addison. Reading, MA.
- Bruegge, B., R. Riedel, G.J. McRae, and A.G. Russell (1993). GEMS: A Geographical Environmental Modeling System, *IEEE Computer Journal*. (In press -- preprints are available.)
- CAPS (1992). *ARPS Version 3.0 User's Guide*. Center for Analysis and Prediction of Storms, University of Oklahoma, Norman, OK.
- Cass, G.R. and G.J. McRae (1981). "Minimizing the Cost of Air Pollution Control," *Environmental Science and Technology*, **17**, pp. 748-757.
- Comer, D.E. (1991). *Internetworking with TCP/IP Volume 1*, Prentice-Hall, Inc., Englewood Cliffs, NJ.
- Dennis, R.L., D.W. Byun, J.H. Novak, K.J. Galluppi, C.J. Coats, and M.A. Vouk (1995). "The Next Generation of Integrated Air Quality Modeling: EPA's Models-3", accepted for publication in *Atmospheric Environment*.
- Digital Equipment Corporation, Hewlett-Packard Company, HyperDesk Corporation, NCR Corporation, Object Design, Inc, and SunSoft, Inc. (1993). "The Common Object Request Broker: Architecture and Specification", Object Management Group, Inc., Framingham, MA.
- Dilley, J. (1993). "Object-Oriented Distributed Computing with C++ and OSF DCE", OSF DCE SIG#49, Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Droegemeier, K.K. et al. (1993). "Design and Implementation of a Scaleable Parallel Storm-Scale Numerical Weather Prediction Model," Center for Analysis and Prediction of Storms and School of Meteorology, University of Oklahoma, Norman, OK.

- EPRI (1993). "Request for Statement of Qualifications RSQ 3189-09," Electric Power Research Institute, Palo Alto, CA.
- Gell-Mann, M. (1994). "The Quark and the Jaguar, Adventures in the Simple and Complex", W.H. Freeman and Co.
- Gibbs, W.W. (1994). Software's Chronic Crisis. *Scientific American*, pp. 86-95.
- Hahn, R.W., G.J. McRae, and J.B. Milford. (1988). "Coping with Complexity in the Design of Environmental Policy," *J. Environmental Management*, **27**, pp. 109-125.
- Hanna, S.R. (1988). "Air quality model evaluation and uncertainty," *JAPCA*, **38**, pp. 406-412.
- Hanna, S.R. (1992). Technical Work Plan for Research on Comprehensive Modeling Systems. Sigma Research Corporation, Report A200-140.
- Hanna, S.R. and P.J. Drivas (1987). *Guidelines for Use of Vapor Cloud Dispersion Models*. New York: Center for Chemical Process Safety, American Society of Chemical Engineers.
- Hansen, D.A. (1992). The CMS Concept. Attachment A to EPRI RSQ 3189-09, November 23, 1992.
- Hansen, D.A. et al. (1994). The Quest for an advanced Regional Air Quality Model. *Environ. Sci. Technol.*, **28** (2), pp. 71A-77A.
- Horwedel, J.E., R.J. Raridon, and R.Q. Wright (1992). "Automated Sensitivity Analysis of an Atmospheric Dispersion Model," *Atm. Environ.*, **26A**, 1643-1649.
- Hwang, D. and D.W. Byun (1995). "Application of Automatic Differentiation for Studying the Sensitivity of Numerical Advection Schemes in Air Quality Models," *High Performance Computing*, A. Tentner (ed.), 52-57.
- IBM Corporation (1995). "IBM Distributed Computing Environment 2.1 for OS/2", IBM Corporation.
- IBM Corporation (1995). "IBM Distributed Computing Environment 2.1 for OS/2 Distributed File System Client Guide and Reference", IBM Corporation.
- Isukapalli, S.S. and P.G. Georgopoulos (1995). "Application of Stochastic Response Surface Methods to Uncertainty Analysis of Photochemical Models," presentation for US EPA-NERL.
- Jeffries, H.E. and S. Tonnesen (1994). "A Comparison of Two Photochemical Reaction Mechanisms Using Mass Balance and Process Analysis," *Atmospheric Environment*, **28**(18), 2991-3003.
- Johnson, R. and V. Reusing (1991). "Object-Oriented Design", UIUCDCS-R-91-1696, University of Illinois, Department of Computer Science, Urbana, IL.
- Kohl, J.T. and B.C. Neuman (1993). "The Kerebros Authentication (V5)" RFC-1510, DDN Network Information Center.
- Kumar, N. (1994). Multiscale Modeling of Urban and Regional Photochemical Air Pollution. Ph.D. Dissertation. Carnegie Mellon University.

- Kumar, N., A.G. Russell, T.W. Tesche, and D.E. McNally D.E. (1994). Evaluation of CALGRID Using Two Different Ozone Episodes and Comparison to UAM results, *Atmos. Environ.*, in press.
- Leser, N. 1990. "Towards a Worldwide Distributed File System", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Levin, E. (1989). Grand challenges to computational science. *Comm. ACM*, **32**, pp. 1456-1457.
- Lyons, W.A., R.A. Pielke, W.R. Cotton, C.J. Tremback, and R.L. Walko (1992). "Operational numerical weather forecasting: a goal within reach," *Preprints, 8th International Conference on Interactive Information and Processing Systems for Meteorology, Oceanography and Hydrology*, American Meteorological Society.
- Lyons, W.A., R.A. Pielke, W.R. Cotton, M. Uliasz, C.J. Tremback, R.L. Walko, and J.L. Eastman (1993). The Application of New Technologies to Modeling Mesoscale Dispersion in Coastal Zones and Complex Terrain. *Air Pollution*, P. Zannetti, et al., Eds., Elsevier Applied Science, London, pp. 35-85.
- Lyons, W.A., R.A. Pielke, C.J. Tremback, R.L. Walko, D.A. Moon, and C.S. Keen (1995). Modeling the Impacts of Mesoscale Vertical Motions upon Coastal Zone Air Pollution Dispersion. *Atmospheric Environment*, **29**, pp. 283-301.
- Lyons, W.A., C.J. Tremback, and R.A. Pielke (1995). The Application and Evaluation of the Regional Atmospheric Modeling System (RAMS) in the Photochemical Modeling System for the Lake Michigan Ozone Study (LMOS), *J. Appl. Meteor*, **34**, pp. 1762-1786.
- McRae, G.J. (1990). "High-performance Computing Environments for Atmospheric Dispersion Modeling," *Complete Modeling Environments for Atmospheric Dispersion Modeling*, ed. Osa, R., Electric Power Research Institute, Palo Alto, CA, pp. 1-31.
- Novak et al. (1994). EPA Third-Generation Air Quality Modeling System - Volume 1: Concept. EPA/600/R-94/220a.
- Odman, M.T., M.A. Tatang, N. Kumar, L.A. McNair, G.J. McRae, and A.G. Russell (1992). "An Investigation of Error Propagation in the California Air Resources Board Air Quality Model," Final report to the California Air Resources Board.
- Open Software Foundation (OSF) (1993). "DCE RPC Internals and Data Structures", Open Software Foundation 11, Cambridge Center, Cambridge MA.
- Open Software Foundation (OSF) (1991). "File Systems in a Distributed Environment", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Open Software Foundation (OSF) (1995). "DCE Product Catalog", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Open Software Foundation (OSF) (1994). "DCE 1.2 Contents Overview", Open Software Foundation 11, Cambridge Center, Cambridge, MA.

References

- Open Software Foundation (OSF) (1993). "Authentication." Chapter 40 in OSF DCE Applications Development Guide, Part 6: DCE Security Services, Prentice Hall, Englewood Cliffs, NJ.
- Open Software Foundation (OSF) (1991) "Interoperability: A Key Criterion For Open Systems", Open Software Foundation 11, Cambridge Center, Cambridge MA.
- Open Software Foundation (OSF) (1992). "Distributed Computing Environment", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Open Software Foundation (OSF) (1992). "Security in a Distributed Computing Environment", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Open Software Foundation (OSF) (1992). "The OSF Distributed Computing Environment: Building on International Standards", Open Software Foundation 11, Cambridge Center, Cambridge, MA.
- Oreskes et al. (1994). Verification, Validation, and Confirmation of Numerical Models in Earth Sciences. *Science*, **263**, pp. 641-646.
- Parnas, D. L., and P.C. Clements (1986). A Rational Design Process: How and Why to Fake It. *IEEE Transactions on Software Engineering*. **SE-12** (2), pp. 251-257.
- Peters, L.K. et al. (1995). "The Current State and Future Direction of Eulerian Models in Simulating the Tropospheric Chemistry and Transport of Trace Species: A Review," *Atmospheric Environment*, **22**(2), pp. 189-222.
- Peterson, M.T. (1994). "DCE; A Guide to Developing Portable Applications", McGraw-Hill Inc., New York, NY.
- Pielke, R.A. (1984). *Mesoscale Meteorological Modeling*. Academic Press, New York, NY, 612 pp.
- Pielke, R.A. (1989). The status of subregional and mesoscale models for air quality and meteorology: Vol. 2. Mesoscale meteorological models in the United States. Environment, Health and Safety Report 009.5-90.4, Pacific Gas & Electric Company, San Ramon, CA.
- Pielke, R.A., W.R. Cotton, R.L. Walko, C.J. Tremback, W.A. Lyons, L. Grasso, M.E. Nicholls, M.D. Moran, D.A. Wesley, T.J. Lee, and J.H. Copeland (1992). A comprehensive meteorological modeling system - RAMS. *Meteorol. and Atmospheric Physics*, **49**, pp. 69-91.
- Pielke, R.A., W.A. Lyons, R.T. McNider, M.D. Moran, R.A. Stocker, R.L. Walko, and M. Uliasz (1991). Regional and mesoscale meteorological modeling as applied to air quality studies. *Air Pollution Modeling & Its Applications VIII*, H. van Dop and D.G. Steyn, Eds., Plenum Press, NY, pp. 259-290.
- Rao, S.T., I.G. Zurbenko, P.S. Porter, J.Y. Ku, and R.F. Henry (1996). "Dealing with the Ozone Non-Attainment Problem in the Eastern United States," *Environmental Manager*, January, 17-31.
- Rosenberry, W. and J. Teague (1993). "DCE and Windows NT", O'Reilly & Associates, Inc., Sebastopol, CA.

- Rosenberry, W., D. Kenney, and G. Fisher. "Understanding DCE", O'Reilly & Associates, Inc., Sebastopol, CA.
- Rubin, A.D. and P. Honeyman (1993). "Long Running Jobs in an Authenticated Environment," CITI Technical Report 93-1, Center for Information Technology Integration, University of Michigan, Ann Arbor, MI.
- Sabot, G., S. Wholey, J. Berlin, and Oppenheimer (1993). Parallel execution of a Fortran-77 weather prediction model. *Proceedings, Supercomputing '93*, in press.
- Seigneur, C. (1994). Review of mathematical models for health risk assessment: VI. population exposure. *Environmental Software*. 9(2).
- Shirley, J., W. Hu, and D. Magid (1994). "Guide to Writing DCE Applications", O'Reilly & Associates, Inc., Sebastopol, CA.
- Stauffer, D.G. and N.L. Seaman (1990). "The use of four-dimensional data assimilation in a limited-area mesoscale model, Part I: Experiments with synoptic scale data," *Mon. Wea. Rev.*, 118, pp. 1250-1272.
- Stroustrup, B. (1991). "The C++ Programming language, 2nd Edition", Addison-Wesley Publishing Company.
- Tanenbaum, A.S. (1995). "Distributed Operating Systems", Prentice Hall, Englewood Cliffs, NJ.
- Taylor, D.A. (1991). "Object-Oriented Technology: A Manager's Guide", Addison-Wesley Publishing Company.
- Tremback, C.J., R.L. Walko, and W.R. Cotton (1994). The Parallelization of an Atmospheric Simulation System. *Preprints, 19th Intl. Conf. on Interactive Information Processing Systems for Meteor., Ocean. and Hydrol.*, AMS, Nashville, 3 pp.
- Tso, Schiller, Linn, Neumann, Machin, Wrabe (1994). "Sandia National Laboratory Report of the ESnet Authentication Pilot Project", RFC-2313, DDN Network Information Center.
- US EPA (1989). User's Guide for RVD 2.0 - A Relief Valve Screening Model. Report Number EPA/SW/DK-89/014 (a).
- Watson, J.G. et al. (1990). "The USEPA/DRI Chemical Mass Balance Receptor Model, CMB 7.0," *Environmental Software*. 5(1).
- Zannetti, P. (1990). *Air Pollution Modeling--Theories, Computational Methods and Available Software*. Van Nostrand Reinhold, NY.
- Zannetti, P., B. Bruegge, D.A. Hansen, W.A. Lyons, D.A. Moon, R.E. Morris, E. Riedel, and A.G. Russell (1995). *Concept Paper, Design and Development of a Comprehensive Modeling System (CMS) for Air Pollution*. Prepared for the Electric Power Research Institute, Palo Alto, CA. FaAA Report SF-R-94-10-11. (The Table of Contents of this report is enclosed as Appendix A.)

APPENDIX A:

**TITLE PAGE AND TABLE OF CONTENTS OF
“CONCEPT PAPER: DESIGN AND DEVELOPMENT OF
A COMPREHENSIVE MODELING SYSTEM (CMS) FOR
AIR POLLUTION”**

FINAL REPORT 1.2

Concept Paper

**Design and Development
of a Comprehensive
Modeling System (CMS)
for Air Pollution**

Prepared by

Paolo Zannetti¹, Bernd Bruegge⁶, D. Alan Hansen²,
Walter A. Lyons³, Dennis A. Moon⁴, Ralph E. Morris⁵,
Erik Riedel⁶, Armistead G. Russell⁶

¹Failure Analysis Associates, Inc., 149 Commonwealth Drive, Menlo Park, CA 94025

²Electric Power Research Institute, 3412 Hillview Avenue, Palo Alto, CA 94304

³Mission Research Corporation, ASTER Division, P.O. Box 466, Ft. Collins, CO 80522

⁴SSESCO, 511 Eleventh Avenue South, Box 212, Minneapolis, MN 55415-1536

⁵ICF Kaiser, Systems Applications International Division, 101 Lucas Valley Road, San Rafael, CA 94903

⁶Carnegie Mellon University, Pittsburgh, PA 15213

for

ELECTRIC POWER RESEARCH INSTITUTE
3412 Hillview Avenue
Palo Alto, CA 94304

6 January 1995

FaAA Report SF-R-94-10-11

TABLE OF CONTENTS

Executive Summary.....	1
Glossary.....	5
Abbreviations.....	7
1.0 The Concept of CMS.....	10
1.1 What is a CMS?.....	10
1.2 What should a CMS do?.....	11
1.3 CMS Users.....	14
1.4 Is environmental modeling possible? (Yes, it is!).....	15
1.5 Open Issues.....	16
2.0 The Users of CMS: Needs, Requirements, Costs and Benefits.....	18
2.1 Intended Users and User Requirements.....	18
2.1.1 Regulatory Users.....	18
2.1.2 Research Users.....	20
2.2 Air Quality Modeling Today and the Benefits of CMS.....	21
3.0 The Attributes of CMS.....	25
3.1 Overall Attributes and Global Requirements.....	26
3.1.1 Ability to Incorporate the Best Science.....	26
3.1.2 Presentation of Results.....	26
3.1.3 Distributed Access to Data.....	27
3.1.4 Software Infrastructure.....	27
3.2 Applications of the CMS.....	27
3.2.1 Research Applications.....	28
3.2.2 Regulatory Applications.....	28
3.2.3 Decision Support.....	29
3.2.4 Source Apportionment.....	29
3.2.5 Emergency preparedness and response.....	29
3.3 User Components.....	30
3.3.1 User Interface.....	31
3.3.2 Visualization.....	31
3.3.3 GIS (Geographic Information Systems).....	31
3.3.4 Training.....	38
3.3.5 Policy and regulatory aspects.....	38
3.3.6 Data acquisition/reduction.....	38
3.3.7 Report Preparation.....	39

3.4 Science Components.....	39
3.4.1 Emission Modeling System (EMS).....	40
3.4.2 Meteorological Modeling System (MMS)	42
3.4.2.1 Current Use of Meteorological Models.....	42
3.4.2.2 Prognostic Meteor. Models and 4DDA.....	43
3.4.3 Air Quality Modeling System (AQMS).....	44
3.4.3.1 Transport and Diffusion.....	44
3.4.3.2 Atmospheric Chemistry	44
3.4.3.3 Aerosol Description.....	45
3.4.3.4 Deposition.....	46
3.4.4 Statistical Methods.....	46
3.4.5 Air Pollution Effects	46
3.4.5.1 Atmospheric Visibility	46
3.4.5.2 Exposure simulation	47
3.4.6 Numerical methods	47
3.5 Design Issues.....	49
3.5.1 System architecture.....	49
3.5.2 Database management.....	50
3.5.3 Networking.....	52
3.5.4 Parallel and distributed computing.....	53
3.5.5 Prototyping.....	54
3.6 Development Issues for CMS modules.....	55
3.6.1 Coding Standards.....	55
3.6.2 Software Repository	56
3.6.3 Tools for Collaborative Development.....	57
3.6.4 Module Evaluation.....	57
4.0 Historical Perspectives.....	58
4.1 Existing Systems.....	58
4.1.1 EPA Guidelines Models	58
4.1.2 ROM/UAM.....	59
4.2 Pre-CMS Systems.....	59
4.2.1 SARMAP	60
4.2.2 LMOS	60
4.2.3 UAM-V Modeling System	61
4.2.4 ARPS	61
4.2.5 GEMS	62
4.3 CMS developments.....	62
4.3.1 Models-3	62
4.3.2 CAMRAQ CMS	63
5.0 CAMRAQ's CMS Development Plan.....	65
5.1 Phase I–Framework Design	65
5.1.1 Scope of Work.....	65
5.1.2 Deliverables	66
5.1.3 Schedule.....	67

5.2 Phase II–CMS Development.....	68
5.2.1 Scope of Work.....	68
5.2.2 Deliverables.....	69
5.2.3 Schedule.....	69
5.3 Parallel Effort (PE Task).....	69
6.0 Expected Difficulties.....	71
7.0 Conclusions and Recommendations.....	74
References.....	76
Appendix A: Summary of the CMS Survey.....	A-1
Appendix B: Hardware/Software Configuration Issues for CMS.....	B-1
Appendix C: Description of Some Existing Air Pollution Modeling Prototypes.....	C-1
Appendix D: Coding Standard Example.....	D-1
Appendix E: Canadian Air Pollution Modeling Efforts.....	E-1

APPENDIX B:

A DISCUSSION OF OBJECT-ORIENTED DESIGN & PROGRAMMING

A Discussion of Object-Oriented Design & Programming

It is acknowledged that various readers of this document will be approaching the computer system design presented here with some foreboding. The introduction of the concepts of object-oriented design and object-oriented programming to the CMS design, while key to its success, can create some confusion and consternation among all but the most dedicated computer “hacker”. We will attempt to explain the key ingredients of the object-oriented approach, in hopes that its power and flexibility can be understood by the community which will come to depend on it.

We will first deal with the concept of object-oriented programming, then extend the view to the process of object-oriented design. In the popular literature, one will read that object-oriented programming as represented by the C++ system, possesses three key qualities beyond existing programming languages: “strong typing”, “encapsulation”, and “inheritance”.

B.1 Strong Typing

To explain these concepts we will return to the programming roots which many of us share, Fortran. “In the beginning”, a Fortran programmer was free to use many shortcuts which today’s environments seldom tolerate. The programmer could assume the implicit definition and type of any variables whose names began with the letters: i, j, k, l, m, or n. This was a dandy shortcut and saved a lot of typing in early programs. But if one wanted an array of floating point variables named MONSTER, the data had to be declared with the Fortran statement REAL MONSTER(1000). In short order, programs grew in size and confusion as the “implicit typing” (I-N) and the “explicit typing” (REAL, INTEGER) were mixed within programs, subprograms, and functions. The original author might be able to keep things sorted out, but others, given this code as an “inheritance” would stumble for some time before attaining comprehension of the variable definitions.

The next step in this chaotic process is when the arbitrary function and subroutine capabilities were placed into the programmer’s hands. Fortran made no restrictions on the variable passed by one routine and received by another. A common blunder in Fortran programming would be:

```
PROGRAM MAIN
  REAL THISISREAL(1000)
  CALL TESTDATA(THISISREAL, STATUS)
END
SUBROUTINE TESTDATA(IDONTKNOW, ISTATUS)
  INTEGER IDONTKNOW(10000)
  ISTATUS= 0
  DO 1000, I=1,10000
    IF( IDONTKNOW.eq.1) ISTATUS=1
```

```
1000 CONTINUE
      RETURN
      END
```

Now the above example is patently ridiculous (unless one has some clever programming scheme afoot). It will cause all sorts of mischief. If the program doesn't crash for having wandered into CPU memory that doesn't exist it will most likely return a false status. This example represents the antithesis of the concept of "strong typing". Here the programmer is attempting to pass a real array of length 1000 to a subroutine which expects a much larger array of integers. Fortran programmers were not the only culprits, of course, PASCAL and "C:" language programmers, could and did commit similar sins.

Enter here the concept of "strong typing", wherein the programming language and its compilers demand that:

1. The user must specify the type of all variables.
2. The type of parameters passed between programs or functions must agree.

A simple mechanism for accomplishing these aims was for Fortran compilers to provide the `IMPLICIT NONE` control statement and for all Fortran or C modules would use a "header file" to define the types of parameters expected by the function or subroutine being called. "C" programmers have been using "header files" from the very outset of their careers. These "header files" are alternatively called "include files" (and now every Fortran-77 and Fortran-90 compiler handles "include files"). An include file which enforces "strong typing" for the above example might appear as:

```
demo.inc

SUBROUTINE TESTDATA(INTEGER (*), INTEGER )
```

and the file would be included in the Fortran compilation of the example by an explicit statement such as:

```
INCLUDE "demo.inc"
```

If this had been the case for the example above, the compiler would have detected the error in the MAIN program which is attempting to pass the REAL array; whereas, the "header file" indicates that a REAL array is required. The employment of strong typing causes the errors to be detected and reported by the compiler rather than at some unpredictable future time. The enforcement of variable types passed between programs can be extended beyond the basic integer, real, and character variables to variable types created from the basic building blocks. Fortran-90, C, and C++ provide the opportunity for the user to create structures (for example, which can be given an arbitrary type name which can then provide "type checking" in "header files").

For example, we create a single entity in the Fortran-90 language:

```
TYPE BASIC_T
  SEQUENCE
  INTEGER A, B, C
  REAL*4 RA, RB, RC
END TYPE BASIC_T
```

This type definition may then be used in variable declarations in the main program, such as:

```
TYPE (BASIC_T) AA, BA, CA
```

and the type definition might then appear in a “header file” which specifies the interfaces for all subroutines and functions which will be called, such as:

```
INTERFACE
  INTEGER FUNCTIONA( ARG1, ARG2)
  BASIC_T ARG1, ARG2
  END FUNCTION FUNCTIONA
END INTERFACE
```

The main program might then utilize the variables and function:

```
STATUS= FUNCTIONA(AA,BA)
```

The inclusion of INTERFACE definitions make it possible for the Fortran-90 compiler to perform the same “strong type” checking which C++ supports. The existence of this facility does not, of course, guarantee that errors will not occur in parameter exchange, but it does provide a “compile time” checking for a very common source of programming effort.

B.2 Encapsulation

Once programming languages and compilers are empowered to check for and enforce policies such as “strong typing” the mechanisms are in place to further improve the processes of software development. One of the most powerful concepts arising from these modern software technologies is “encapsulation”. An entity can be created which contains all of its data and all the procedures which operate on that data. If the operations on the data within this entity are restricted to those included with the entity and no other program can touch the entity’s data, that entity could be termed “encapsulated”. As an example, a Fortran subroutine with several entry points could be contracted as an “encapsulated” module:

```
SUBROUTINE NEWMESH( SIZEX, SIZEY)
  INTEGER SIZEX, SIZEY
  REAL MAINMESH(1000,1000)
  COMMON /MINE/XDIM, YDIM, MAINMESH
  INTEGER XDIM, YDIM
```

```

INTEGER I, J
XDIM= SIZEX
YDIM= SIZEY
DO I=1, SIZEX
    DO J= 1, SIZEY
        MAINMESH(I, J)= 0.0
    END DO
END DO
RETURN
ENTRY ADDPOINT( X,Y, VALUE)
    INTEGER X,Y
    REAL VALUE
    MAINMESH(X,Y)= VALUE
RETURN

ENTRY GETPOINT(X1,Y1, RETVALUE)
    INTEGER X1,Y1
    REAL RETVALUE
    RETVALUE= MAINMESH(X1,Y1)
RETURN

END

```

When the routine `NEWMESH` is called, it fills a two dimensional matrix of `SIZEX` by `SIZEY` with a floating point zero. Note that the actual size of the array is held in the variables, `XDIM` and `YDIM`, and are retained in the named `COMMON` block. The subroutines `ADDPOINT` and `GETPOINT` place or return a floating point value in that matrix at a specified location.

If the labeled `COMMON` block "MINE" does not appear in any other module of the program, this data might be called "private", and thus, this series of routines would create and maintain the data entity "MAINMESH" in a "fully encapsulated" manner. No other process can affect the `MAINMESH` matrix.

If several different `MAINMESH` matrix sets are needed then copies of this example could be created with different names for the subroutines, `NEWMESH`, `GETPOINT`, and `ADDPOINT`, and the `COMMON` name. Each of these could be called "objects" even though they are not exactly like the C++ "object" counterparts.

In C++, the above example would appear as a definition of a "class" (perhaps called `NEWMESH`) and "objects" are created and given names using this "class" definition. The subroutine name would remain the same for all objects but the invocation of that subroutine would be "qualified" by pre-appending the object name followed by a period:

```

newmesh mesh-a(100,100);    /* create a newmesh object named mesh-a */
newmesh mesh-b(112,134);   /* create a newmesh object named mesh-b */

mesh-a.addpoint(10,20,3.1415);

```


In C++, the array MAINMESH could be a static array within the object or it may be created dynamically, depending on the needs of the applications. In C++ functions, data can be specified as "private"; thus, the function addpoint might call another function called "testpoint" which would only be available to the mesh-a object and no other. This concept of "hiding" critical data and functions, gives C++ more flexibility in the scrutiny of its objects than is possible in Fortran-90.

Once a "C++ class" or Fortran-90 subroutine has been defined in this way, the last step is to compile the functional code and add it to a binary library. This library and the "include" files, which contain the INTERFACE definitions then become the formal definition of the "object" which other programmer can incorporate into their own programs. Like the basic mathematics libraries which accompany every computer system these days, the internal workings of these functions are invisible to the user. They know only about the INTERFACE methods they are given in the include files.

This then is the process of "encapsulation", creating modules which contain data and processes which the user cannot directly access except through the explicit function calls permitted by the "header" (include) files.

B.3 Inheritance

The last major feature of object-oriented programming we will discuss is that of "inheritance". This concept is not native to Fortran-90 as it is in C++ but we can try and demonstrate how "inheritance" works in a primitive way using our Fortran example. Let us pose a situation wherein a programmer wishes to provide a more robust matrix handling capability than our simple 'get and put' example. This new module would access the NEWMESH directly to perform a collection of new functions among which might be MESHSUM:

```
SUBROUTINE NEWFUNCS(DUMMY)
.
  ENTRY MESHSUM( SUMVALUE)
    REAL MAINMESH(1000,1000)
    COMMON /MINE/XDIM, YDIM, MAINMESH
    INTEGER XDIM, YDIM
    INTEGER I,J
    SUMVALUE=0.0
    DO I=1,XDIM
      DO J=1,YDIM
        SUMVALUE= SUMVALUE + MAINMESH(I,J)
      END DO
    END DO
    RETURN
.
END
```

This set of routines could be compiled and included in the same library as the NEWMESH routines. A set of INTERFACE descriptions would then be included in the header file.

Since MAINMESH is shared by these two modules (and only by them through the unique COMMON/MINE/, it could be said that the module NEWFUNCS has “inherited” the matrix from NEWMESH. Further, the subroutine call definitions which were originally created for a NEWMESH “header” file could be incorporated with the INTERFACE definitions for the subroutine calls in NEWFUNCS resulting in a single “header” file called NEWFUNCS. The result would be an apparent “inheritance” of the NEWMESH functions as well as the data from the example we started with.

The object-oriented view of “inheritance” is much broader and more flexible, but the idea of building definitions for objects from the definitions for other objects is a primary technique in object-oriented design and programming. Think of this process as being the next step beyond the manner in which we now program, building routines out of subroutines and functions and then using these routines as building blocks for more complex routines. The major difference in object-oriented programming is that instead of normal Fortran or C routines we use as the basic building material self-contained (encapsulated) object definitions which possess the properties which we described above.

B.4 Object-Oriented Design

Once the object-oriented programming paradigm is in place, along with the compilers and other tools to improve programming productivity, the next step is to utilize these concepts as a basis for the system design process itself. Characteristically, this means approaching the design from a very high level, first defining the nature of data and processes in an almost philosophical manner and from that specifying the object definitions (or “classes”) which will best express the high-level view of the system. This process is quite orthogonal to the traditional methods used for complex Fortran and C programs in the past and takes a while to reach a point where the procedures are intuitive.

Experience with object-oriented design and C++ implementation has demonstrated that this methodology is quite powerful and results in systems which are more easily implemented, understood, and maintained. Properties such as quality and efficiency in the resulting system are vastly more easy to assure in object-oriented systems.

APPENDIX C:

DESCRIPTION OF SOME EXISTING AIR POLLUTION MODELING PROTOTYPES AND SOFTWARE PRODUCTS

Table of Contents - Appendix C

ADOM, Canada/ENSR	C-3
ADSO Models, ARIA, France	C-4
ARPS, CAPS	C-5
Canadian Air Pollution Modeling Efforts	C-6
CIT Airshed Model, Carnegie & California Institute of Technology	C-9
EUMAC Modeling System, European Modeling System, University of Cologne & others	C-13
EWB, SESCO	C-14
FAST UAM, ENVIRON International Corporation	C-17
GEMAP, Radian/Alpine Geophysics/CMU	C-18
GEMS, CMU	C-21
Guideline Models, US EPA	C-24
HYPACT, ASTER	C-25
LMOS, Lake Michigan Region	C-27
MARS/MEMO (EUMAC Zooming Models), University of Thessaloniki	C-28
MM5, NCAR/Penn State	C-29
Models-3, HPCC/US EPA	C-32
MONTECARLO, FaAA	C-34
OMEGA, SAIC	C-35
RAMS, ASTER	C-39
Regulatory Models, Trinity Corporation	C-41
ROM/UAM Modeling System, US EPA	C-42
SARMAP, SJVAQS/AUSPEX	C-43
UAM, Systems Applications International	C-55

ADOM

Canada/ENSR

A similar effort to the RADM model development was undertaken by the Canadian government with ENSR to develop a regional acid deposition model. The model that was developed is called the Acid Deposition and Oxidant Model (ADOM) (Venkatram *et al.*, 1992). It shares many of the same characteristics (e.g. basics of model formulation) as RADM, with some important differences. Two of the more notable differences is the chemical mechanism used (Atkinson and Lloyd, 1984 versus the RADM mechanism), and the treatment of cloud processes, including aqueous phase chemistry. Another difference is that the RADM photochemical model was tightly linked to the MM4 meteorological model, where as ADOM is not. Instead, ADOM has used meteorological fields developed by the Canadian Meteorological Service model, though is not tightly coupled to any single model. One of the important differences between ADOM and other regional models is its treatment of both cumulus and stratus clouds differently. ADOM (or its derivatives) has been applied in North America and Europe, looking at both acid deposition as well as ozone control. ADOM and RADM were subjected to a regional model intercomparison (EMEFS).

References

- Atkinson, R. and A.C. Lloyd (1984). "Evaluation of Kinetic and Mechanistic Data for Modeling of Photochemical Smog," *J. Phys. Chem. Ref. Data*, **13**, 315-444.
- Venkatram, A. *et al.* (1992). "The Development of the Acid Deposition and Oxidant Model (ADOM)," *Environ. Pollut.*, **75**, 189-198.

ADSO Models

ARIA (France)

As an outgrowth of work being conducted at EdF (Electricite de France) in France, a company, ARIA, was formed to further develop and apply not only the EdF air quality models, but others as well. Such models include both meteorological models (both diagnostic and prognostic) and dispersion models. The models have been engineered to be more readily coupled, particularly between the meteorological models and the dispersion models. The application package is called ADSO (Atmospheric Dispersion Simulation Objects). These models are engineered to share the same I/O structures. The GUI is basically 2D with MOTIF. NCAR Graphics is used for visualization. Recently, the models have been linked with savi3D for visualization. The visualization packages can also be used for investigating atmospheric measurements as well. More limited versions of the package for PCs have been developed, primarily serving those with limited needs or for educational purposes.

The primary meteorological models are MINERVE™, a diagnostic model, HERMES™, a regional scale prognostic model, and MERCURE™, a smaller scale CFD-type code. HERMES can use, optionally, a variety of turbulence closure models. At present, it does not have FDDA capabilities, and is hydrostatic. MERCURE is a descendent of CFD codes, and includes more detailed treatment of physical processes, such as turbulence, at smaller scales. It includes both K-L and K-e turbulence parameterizations.

Dispersion models included in the ARIA ADSO family includes DIFBOU™, TRAMES™, HERMES Disp, MERCURE and SPRAY™. DIFBOU is a Gaussian puff model using 0 or 1D meteorological data. TRAMES is a 3D Puff model. HERMES Disp is a 3D Eulerian model, but is not applicable to small scales. MERCURE is an extension of the MERCURE flow model. SPRAY is a Lagrangian Monte Carlo dispersion model.

The ADSO models have been applied in a variety of settings, and the results presented in the scientific literature. The coupling between the models, and the use of a common I/O structure would classify these models as a preliminary modeling system. However, they do not include a detailed emissions model, nor regional photochemical capabilities.

ARPS

CAPS

The Advanced Regional Predictions System (ARPS), Release 3.1, is an atmospheric modeling code developed by the Center for Analysis and Prediction of Storms (CAPS). The code is a full-physics, general-purpose mesoscale atmospheric model (CAPS, 1992). Even though its primary focus is on severe thunderstorms and tornadoes, we list ARPS in this section of pre-CMS systems because of its unique computational features.

Because of ARPS' coding style, the system has been welcomed into the computer science community as a framework for developing and testing compilers, translation tools and automated differentiation techniques. In fact, unique to ARPS is the use of structured finite difference operators, which not only preserve the formal structure in the code of the governing equations of fluid dynamics, but also expose the parallelism inherent in them (but typically masked by the solution algorithms). Thus, the code becomes naturally scaleable for general classes of parallel machines. In addition, this operator-based solution methodology has been shown to reduce development and debugging time substantially relative to more conventional methodologies.

In fact, ARPS was developed specifically for broad classes of scaleable-parallel architectures (Droegemeier *et al.*, 1993). Sabot *et al.* (1993) took ARPS and ported the entire application to a massively parallel computer, the Connection Machine system. This allowed the model to simulate well ahead of the evolving weather.

References

- CAPS (1992). *ARPS Version 3.0 User's Guide*. Center for Analysis and Prediction of Storms, University of Oklahoma, Norman, OK.
- Droegemeier, K.K. *et al.* (1993). "Design and Implementation of a Scaleable Parallel Storm-Scale Numerical Weather Prediction Model," Center for Analysis and Prediction of Storms and School of Meteorology, University of Oklahoma, Norman, OK.
- Sabot, G., S. Wholey, J. Berlin, and Oppenheimer (1993). Parallel execution of a FORTRAN-77 weather prediction model. *Proceedings, Supercomputing '93*, in press.

Canadian Air Pollution Modeling Efforts

The Canadian air pollution modeling effort is spread across government, academic, and private sectors. The following is far from a comprehensive survey of the field. Highlighted are several models and model components that could have particular significance in the development of a comprehensive modeling system for reasons that are explained in the descriptions. While Canada does have active modeling efforts in the small scale, models used in the mesoscale and up for air quality applications (not climate) are described here.

Program in Support of NO_x/VOC Management Plan

Under Canada's NO_x/VOC Management Plan, three areas have been identified for significant modeling efforts. These are the lower Fraser Valley, the Windsor/Quebec corridor and the southern Atlantic Region.

In the lower Fraser Valley, several modeling efforts are underway with involvement by a number of researchers including Dr. Steyn of UBC and Dr. Hedley of NRC. The urban airshed model (UAM) is being adapted for use with RAMS and with MC-2 (which will be discussed further below).

The Canadian Institute for Research in Atmospheric Chemistry (CIRAC) is managing a model intercomparison project to assess the performance of several models to simulate tropospheric ozone in the Windsor/Quebec corridor. Performance is being assessed based on the EMEFS data sets. The first case is in the Summer of 1988 when high values of tropospheric ozone were experienced. The second case is in the Fall of 1991 when the levels of tropospheric ozone were much lower. The models being assessed include: ALOM, ADOM, ADOM/GESIMA, and MC-2/ADOM. Since these models are being considered for use in scenarios to assess potential control measures for tropospheric ozone, which is a major Canadian issue as well as one of the issues driving the development of a comprehensive modeling system, they will be briefly discussed here.

The AES Lagrangian Oxidants Model (ALOM) is a Lagrangian model which was developed by the Atmospheric Environment Service about a decade ago (Olson *et al.*, 1979) and which has had a long and illustrious record in doing air quality work. It is most lately being applied to simulations of tropospheric ozone in the Windsor/Quebec corridor.

In the 1980s, Canadian effort focused on the development and validation of the Acid Deposition and Oxidants Model (ADOM) for assessing acid deposition. This model is a sister model to RADM and has been the subject of extensive intercomparisons with RADM and with a special data set collected in the extensive EMEFS field studies. More recently, this model has been coupled by the Ontario Ministry of the Environment and Energy to GESIMA, a German mesoscale model, for use in smaller scale applications.

Recently, AES is beginning to use numerical weather prediction models more extensively as “drivers” of air quality models. This is, of course, only possible when the model has a sophisticated boundary layer which allows realistic simulation of pollutants. The latest generation meteorological driver is represented by MC-2, which is described more fully in the following table. For the evaluation, MC-2 is being coupled to the chemistry code from ADOM.

In the southern Atlantic Region, AES is also planning to apply the MC-2 model linked to ADOM chemistry to simulate the tropospheric ozone levels in the region.

Summary of the MC-2 Model
(courtesy of R. Benoit)

The MC-2 model is a Canadian mesoscale community model which is user friendly and can be interfaced with dispersion models. It has the following characteristics.

Dynamics:

- Fully compressible Euler equations
- Horizontal Cartesian coordinates on polar stereographic projection of sphere; vertical terrain following (modified Gal-Chen) coordinate
- Helmholtz equation on perturbations of log [pressure (t+Dt)]. Adaptive ADI scheme
- Nesting of lateral/upper boundaries for all prognostic variables (sponge zone)
- Variable resolution in the vertical, 3D staggered finite differences
- time levels (t-Dt, t, t+Dt) + filter + “off-centering” (epsilon = 0.1)
- Semi-implicit time scheme that handles both acoustic and gravity oscillations. Uses an isothermal temperature reference profile
- 3D semi-Lagrangian scheme. Truncated tricubic spline
- Horizontal and vertical staggering with second order finite differencing
- Horizontal diffusion (-) time-implicit on horizontal and vertical velocity, temperature, pressure, and humidity

Physics:

- Essentially the same as in RFE, SEF (operation models), and GEF
- Plus more recent options (MC-2 = numerical laboratory for physics)
- Surface physics options:
 - (land: force-restore/ocean: fixed SST + Charnock variable drag)
 - Canadian Land Surface Scheme (CLASS): soil, vegetation, and snow layers
- Prognostic turbulent kinetic energy (TKE)
- Shallow moist convection; orographic gravity wave drag
- Convective scheme options:
 - Kuo scheme
 - moist convective adjustment
 - Frisch-Chappel mesoscale convective scheme including downdrafts

- Explicit condensation options:
 - large-scale supersaturation removal
 - prognostic cloud water scheme (Sundqvist, 1989) with integrated convection
 - warm and cold processes cloud scale microphysics
- Advanced radiative computation (solar + IR)

Program in Support of Emergency Response

The Canadian Meteorological Center in Dorval, Quebec, is one of the WMO Emergency Response Centers globally. The model that is used is based either on objectively analyzed data or on the standard weather prediction models used by the weather service coupled with a semi-Lagrangian tracer code. In the ATMES study which evaluated such models based on the Chernobyl data, an earlier version placed 5th worldwide (Pudykiewicz, 1991).

More recently, there has been research to move such emergency prediction to the mesoscale. Two approaches are being pursued; reducing the operational scale of the weather prediction models to 25 km and developing a simplified boundary layer model, mesoscale Boundary Layer Forecast Model (BLFMESO), for specific application to emergency response (Daggupaty *et al.*, 1994).

Program in Support of Air Toxics

A similar modeling system to that being used for emergency response is being used to simulate the global transport of air toxics. Simulations of organochlorine transport to the Arctic are underway. One major problem with the modeling of persistent organic pollutants is the lack of credible emission inventories. In order to make some progress internationally on the establishment of such inventories, Canada has established the Canadian Global Emissions Inventory Center (CGEIC). This center has collaborative programs in place with several other countries to produce emission inventories suitable for modeling.

References

- Daggupaty, S.M., R.S. Tangirala, and H. Sohota (1994). "BLFMESO - A 3-Dimensional Mesoscale Meteorological Model for Microcomputers," Boundary Layer Met., 71, 81-107.
- Olson, M.P., E.C. Voldner, K.K. Oikawa, and A. MacAfee (1979). "A Concentration/Deposition Model Applied to the Canadian Long-Range Transport of Air Pollutants Project - A Technical Description," AES Rpt. No. LRTAP 79-5, 30 pp.
- Pudykiewicz, J. (1991). "A Numerical Simulation of Contaminant Transport to the Arctic," Numerical Modeling of Air Pollution, H. VanDop, editor, Reidel Publishing Co.
- Sundqvist, H., E. Berge, and J.E. Kristjansson (1989). "Condensation and Cloud Parameterization Studies with a Mesoscale Numerical Weather Prediction Model," Mon. Wea. Rev., 117, 1641-1657.

CIT Airshed Model

Carnegie Mellon University & California Institute of Technology

The CIT model is a descendent of the Caltech gas-phase photochemical air quality model developed, originally, at the California Institute of Technology in the late-1970s and early-1980s. Later, the ability to track the formation and transport of aerosol nitrate was added. Due to its use as a research tool, other changes to components of the model were updated to reflect our increase in knowledge about the processes governing the dynamics of pollutants, such as adding the chemistry of methanol and other species to the original Caltech mechanism, use of an alternative chemical mechanism (e.g., the LCC chemistry), and use of a resistance-based, land-use dependent deposition scheme. Each of these improvements strengthened its scientific foundation, and kept it at the forefront of model capabilities.

Photochemical models simulate the physical and chemical processes affecting the dynamics of compounds in the atmosphere. Mathematically, the pollutant dynamics are described by the species conservation equation:

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (\mathbf{u}c_i) = D\nabla^2 c_i + R_i(c_1, c_2, \dots, c_n, T) = S_i(t)$$

where c_i is the concentration of species i , \mathbf{u} is the velocity vector, D is the molecular diffusivity of species i in air, R_i is the chemical production, T is temperature and S_i is the (elevated) source rate of i . Boundary conditions govern the concentrations and pollutant flux into the region of interest. In practice, an approximation to the above set of continuous, differential equations is solved using finite techniques solving for spatially averaged concentrations. Likewise, temporal averaging is used to allow the calculation of the fluid dynamic transport to be computationally tractable since the flowfield is highly turbulent. Temporal averaging of the advective flux leads to parameterization of the transport into a term describing bulk motion of the air (e.g., mean winds) and a diffusive, turbulent flux (e.g., K-theory approximation):

$$\overline{(\mathbf{u}c_i)} = \overline{\mathbf{u}}\langle c_i \rangle + \overline{\mathbf{u}'c_i'} \approx \overline{\mathbf{u}}\langle c_i \rangle - K\nabla\langle c_i \rangle$$

where $\overline{\mathbf{u}}$ is the temporally averaged velocity vector, \mathbf{u}' is the turbulent fluctuating velocity, c_i is the temporally averaged concentration of species i , c_i' is the instantaneous fluctuating concentration and K is the second order diffusivity tensor. This is a turbulence closure parameterization. In the CIT model, K is taken to be diagonal (see discussion below). Similarly, spatial discretization, and the chaotic nature of turbulence, lead to the introduction of a spatially averaged concentration that would be found over a number of realization of the nearly same experiment (i.e., where all the mean variables such as wind velocity are the same), which is called the ensemble average, $\langle c \rangle$. The resulting final equation, called the Atmospheric Diffusion Equation (ADE) is a statement of the conservation of species for an ensemble, grid averaged, field.

As implemented in the CIT model, the horizontal boundary conditions are:

$$\begin{aligned} \left[\bar{u} \langle c_i \rangle - K \cdot \nabla \langle c_i \rangle \right] \cdot \hat{n} &= \bar{u} \langle c_i \rangle^b \cdot \hat{n} & \bar{u} \cdot \hat{n} \leq 0 \\ -\nabla \langle c_i \rangle \cdot \hat{n} &= 0 & \bar{u} \cdot \hat{n} > 0 \end{aligned}$$

where $\langle c_i \rangle^b$ is a specified boundary concentration.

The vertical boundary conditions are

$$-K \cdot \frac{\partial \langle c_i \rangle}{\partial z} - v_{g,i} = E_i \quad z = 0$$

and

$$\frac{\partial \langle c_i \rangle}{\partial z} = 0 \quad z = H$$

where $v_{g,i}$ is the deposition velocity of species i , E_i is the emission rate, and H is the top of the modeling region. Unlike other models, the choice here has been to set a no-gradient top boundary condition. In essence, this is derived from the view that the best estimate of the local concentration just above the modeling region is the current prediction at the top of the modeling region. This makes it insensitive to setting an arbitrary, often very high, upper boundary condition. Instead, the top of the modeling region must be set well above the top of the mixed layer, so the concentration gradients should be small and the absolute concentrations of most species should also be relatively small. This mode of operation is particularly useful for control strategy calculations because an arbitrary choice of boundary conditions does not impact the results severely.

Depending on the topographic relief of the modeling region, numerical implementation of the ADE can be complicated. Solution is simplified by using a terrain following coordinate system. Amongst other aspects of the transformation is the addition of off-diagonal terms in the eddy diffusivity tensor, K . It has been implicitly assumed, and can be shown for all but very rugged terrain, that those off-diagonal terms are small for urban modeling.

Solution of the atmospheric diffusion equation for systems such as urban smog and acid deposition is computationally demanding. Standard techniques will generally be inefficient, and often inaccurate. The CIT model has adopted and refined peer-reviewed numerical techniques that are both accurate and fast (economical).

Given the wide variation in time scales of the various physical and chemical processes, and the structure of the ADE, the CIT model employs operator splitting. These methods are ideally suited to deal with this problem because they allow specialized techniques to

treat individual components of the original equation. The basic elements of the splitting process are the use of operators, L , that describe horizontal transport:

$$\frac{\partial c}{\partial t} = -\nabla_H \cdot \mathbf{u}c + \nabla_H \cdot K\nabla_H c = L_H c$$

vertical transport:

$$\frac{\partial c}{\partial t} = -\frac{\partial}{\partial z} \cdot \mathbf{u}_z c + \frac{\partial}{\partial z} K \frac{\partial}{\partial z} = L_z c$$

and chemistry and emissions:

$$\frac{\partial c}{\partial t} = R + S = L_c c$$

The horizontal transport operator can be further decomposed into operators treating transport in the x and y direction. If A_x , A_y , A_z , and A_c are numerical approximations to the transport and chemical operators, then a solution can be obtained from the sequence:

$$C^{n+1} = A_x A_y A_z A_c (2Dt) A_z A_y A_x C^n$$

where n is the time level and Dt the numerical time step.

In the original implementation of the Caltech model, the vertical transport operator was independent of the others, as shown above. However, the time scale for the vertical diffusive transport is very short during the day in the mixed layer, and is comparable to the chemical time scales of many components. Also, the solution of a diffusively-dominated process (such as vertical transport in the mixed layer) has a similar exponential structure like chemical decay. Thus, in later implementations of the Caltech model, the vertical transport operator was combined with the chemical operator, leaving:

$$C^{n+1} = A_x A_y A_{cz} (2Dt) A_y A_x C^n$$

This structure allows very fast and accurate solution techniques to be applied to solve the advectively-dominated horizontal transport, and a separate technique for chemical production and decay.

Advective dominance in the horizontal transport leads to the solution having hyperbolic characteristics. Numerical solution leads to the production of spurious waves (dispersion) and loss of detail (diffusion). This is a classic problem in computational fluid dynamics, and a number of specialized techniques have been developed. Of those tested, the Chapeau function-based finite element technique was shown to be both

accurate and fast, and is used in the CIT model. More recent tests have further verified its characteristics.

Most of the computer time involved in solving the ADE is consumed by integrating the chemical production and loss of each species. This is due, in large part, to the stiffness of the system of coupled differential equations. Tests showed that the implicit, hybrid, asymptotic, exponential solver of Young and Boris provided significant efficiency advantages over other techniques for solving stiff systems of equations.

An indication of the computational effectiveness of the techniques chosen is that they have been adopted by much more recent models. As specified above, the "old" CIT model proved to be a reliable and accurate tool for investigating and elucidating the dynamics of pollutants in the atmosphere.

EUMAC Modeling System

European Modeling System, University of Cologne and others

The centerpiece model of the European modeling system development effort is the EUMAC project. It is a combination of a photochemical transport model, a meteorological model and an emissions model, linked together to facilitate regional modeling. The photochemical model is a descendent of RADM. The meteorological model is a descendent of MM4/5, and has both FDDA and non-hydrostatic capabilities. MM5 and RADM are described in more detailed elsewhere in this appendix. Both of the models, as used in the EURAD system, include updates, primarily in the treatment of the physics. The emissions model, and emissions inventory preparation procedures in general, are not as detailed as comparable systems in the United States (e.g., in comparison with GEMAP or the large number of source classes and estimation procedures used for inventory development here). This is primarily a reflection of the longer term, historically more intense regulatory climate in the US. A number of recent studies in Europe are aimed at developing more detailed inventories, which in turn will drive the emissions model development.

A key feature of the EURAD system is that it is a linked system, such that when one changes the domain or grid size used, it is consistently and readily treated by all three system components. Thus, there is less problem with making sure that all changes are made consistently throughout the model application, and significantly less effort is required for new applications.

The EUMAC model has been applied to a number of regional domains in Europe, primarily for scientific study. It has been used in a one-way nesting mode, though it currently does not have two-way nesting capabilities for the photochemical modeling portion. At present, it does not have a plume-in-grid capability as well.

EWB

SSESCO

The Environmental WorkBench product (EWB) from SSESCO is an extremely easy to use visualization and analysis application targeted at environmental data. It is aimed at putting sophisticated interactive visualization capabilities directly in the hands of the scientist or engineer working with the data, without the need for a computer support person. It is distinguished from data flow products like AVS in that it provides a ready to use interface without any network programming required. The EWB currently has numerous users in the fields of meteorological research, air quality work, and groundwater remediation. Besides allowing the user to interactively explore their data sets, it also provides a highly effective means for communicating their insights to others. It has been used with great success in a courtroom setting for a Superfund case to communicate complex 3D information about plume structure and movement. The EWB also serves as an integration platform for models and additional analysis facilities. SSESCO is currently installing real-time meteorological modeling systems based on the EWB at two sites, one for emergency response purposes at a DOE laboratory and one used by an electrical utility for current routing calculations.

Perhaps the most important attribute of the EWB is its user interface. By adopting an object oriented approach, it achieves a synthesis of sometimes contrary attributes: easy to learn, efficient for the experienced user, and logically extensible. This is accomplished in part by using a very "flat" user interface. That is, a large part of the functionality is always readily accessible to the user without "diving" into menus and searching. Customers frequently report back at the ease and quickness in getting new users up to speed.

The EWB is targeted specifically at environmental applications. By staying focused on this market, development efforts are tailored to its specific needs. Examples of this include unsurpassed support for handling of nested grid models, and the optional use of meteorological station model style wind arrows. Another example is the support for discrete observational data integrated with model generated data such as gridded or particle model outputs. The EWBs frame manager makes it easy to view together multiple data sets from different sources with asynchronous, non-uniform time steps. For example, one could pull together topography and land use data with outputs (and inputs) from emissions, meteorological, and photochemical models, along with sets of real data from meteorological and air quality monitoring systems. The time stepping control system allows the user to step through time in an interactive rendering mode which employs a memory management scheme so that there is no requirement that all data over time fit into memory. They can also save sequences of images for use in third party animation tools.

Systems currently supported by the EWB include: IBM RS/6000, SUN, and Hewlett Packard workstations, and OS/2 on the PC.

System features include:

- Random access file structure using the netCDF based public domain MeRAF file system with support for gridded, discrete (non-grid-based observation), and particle types
- Support for geo-referenced or Cartesian coordinate systems
- Object oriented graphical user interface (GUI) that is very easy to use
- Tools for converting model and observational data sets and data writers to netCDF
- Interactive rotation/translation of scenes in 3D space
- Time sequencing controls to step forward/backward, animate sequentially, or go to a chosen time step; including multiple asynchronous or non-uniform time steps
- Interactive slicers to select cross sections through 3D data sets
 - geo-referenced slicer that gives feedback in geographic units dependent on the current coordinate system
 - ijk slicer based on the indexing information of the 3D grid
 - arbitrary slicer to slice along non-orthogonal angles and at varying resolutions
- Display operators available on the slices
 - contour lines with selectable contour levels
 - color shading by data value with variable transparency level
 - arrow and streamline representation for vector quantities
 - positional reference lines at user selected intervals
 - color coded shapes at each grid node
- Multiple 3D iso-surfaces at selected parameters and values with variable transparency
- Display of particle positions with coloring by type, height, and source
- Display of discrete data using colored spheres and labels for scalar data and arrows for vectors (with arrowheads or meteorological style)
- Multiple user definable color maps to which iso-surface and colored field shading may be separately assigned
- On screen annotation for generation of report ready figures

- Image export in any of the common image formats (gif, tiff, encapsulated postscript, etc.)
- Graceful handling of missing or bad data values by all the graphics rendering routines
- Automatic data synchronization to allow automatic screen updating as new data arrives in real time from a model or set of sensors

FAST UAM

ENVIRON International Corporation

In photochemical grid models, typically 80 percent of the CPU time is spent numerically integrating the time evolution of (solving) the chemistry. The desire to increase the complexity of photochemical mechanisms will tend to push this figure higher. Whilst improvements in computer performance continue at an impressive pace, the relief provided by the arrival of each new generation of computers seems only temporary because the gains are soon eroded by expanded modeling domains, nested-grids, higher resolution, etc. Thus, computer time seems always to be at a premium and the need for significantly more efficient chemistry solvers seems likely to remain for the foreseeable future.

Chemistry can be speeded by using parallel computers if the solver/model is developed to exploit the parallel architecture. A potential drawback to this approach is that the resulting code may be tailored to a specific computer architecture and only run efficiently on a limited number of rather expensive computing platforms. Development of a fundamentally more efficient chemistry solver is a more attractive approach since it yields benefits on all computing platforms. ENVIRON has developed a highly-efficient chemistry solver that is based on an "adaptive-hybrid" approach. Relative to the standard chemistry solver in UAM-IV and UAM-V, our approach results in about a ten-fold speedup in the chemistry and an overall speedup in the model of 3 or 4 times. An attractive feature of this solver is that its efficiency relative to other solvers should improve further for more detailed photochemical mechanisms. Model performance with the ENVIRON fast solver is very similar to the standard version of UAM-IV and its use does not change any conclusions that would be drawn about model performance relative to observations or the effectiveness of control strategies.

GEMAP

Radian/Alpine Geophysics/CMU

The Geocoded Emissions Modeling and Projections (GEMAP) system was developed by Radian Corporation under the joint funding of SARMAP and LADCO. It prepares gridded, hourly, and speciated emission estimates for use by photochemical models. The GEMAP system may be used to prepare emissions estimates for State Implementation Plans (SIPs), urban- and regional-scale air quality models, and air quality management plans. GEMAP has been built around SAS[®] and ARC/INFO[®] which are state-of-the-science software packages for manipulating data. Also included in GEMAP are MOBILE (US EPA's Fortran model for computing mobile source emissions factors), EMFAC (California Air Resources Board's Fortran model for computing mobile source emissions factors), and BIOLCM (US EPA's model for estimating adjustment factors due to leaf canopy effects).

GEMAP consists of eight separate models for computing emissions estimates. Models for point, area, motor vehicle, and biogenic sources are used to produce baseline emission estimates for these four source types. The Grid Definition Model prepares the domain modeling grid for spatial allocation of emissions estimates that are generated by other GEMAP models. The Speciation Model applies split factors specific to a VOC chemical mechanism to speciate the criteria emissions estimates that are computed by the five, GEMAP emissions estimates models (Point Source Emissions Estimates Model, Area Source Emissions Estimates Model, Crude Oil Storage Tank Emissions Estimates Model [COST], Biogenics Emissions Estimates Model [BIOEM], and the Motor Vehicle Emissions Estimates Model [MoVEM]) into the lumped chemical groupings that are required by an air quality model. The Projections Model is used to project baseline emissions estimates to future years. Finally, the Uncertainty Model is used to modify emissions estimates to reflect uncertainty in the emission estimates.

The GEMAP Grid Definition Model generates the grid system cell structure; that is, the emissions modeling domain. The GEMAP Grid Definition Model is built around ARC/INFO, a geographical information system (GIS). Use of ARC/INFO enables the user to generate the emissions modeling domain grid structure or other relevant geographical data as *layers* or *coverages* of information. Spatial data are represented as features (points, lines, or polygons), with both locational and thematic (attribute) components – attribute data may include population and housing counts, survey data, plant species, etc. The locational data pertain to the geographical location of a feature on the Earth's surface and are associated with an acceptable coordinate system, such as the Universal Transverse Mercator (UTM) coordinate system. Thus, lengths, perimeters, or surface areas of a feature can be calculated while keeping the feature-associated data in spatial relation with other geographical data.

The GEMAP Point Source Model uses state-supplied annual-average, average-day, and/or day-specific emission estimates to compute gridded, hourly adjusted, pollutant-specific point source emission estimates using a combination of SAS- and ARC/INFO-based processors. Please note that the GEMAP Point Source Model is not a true emissions estimates model. It utilizes emissions estimates prepared by an external

entity to generate gridded, hourly, criteria emissions estimates. During the emissions estimation process, the GEMAP Point Source Model can account for day-specific emissions events, regulatory control programs (control efficiency, capture efficiency, and rule-effectiveness), and source specific operating parameters and schedules.

The GEMAP Area Source Model uses state-supplied, county-wide annual-average, average-day, and/or day-specific emission estimates to compute gridded, hourly adjusted, pollutant-specific area source emission estimates using a combination of SAS- and ARC/INFO-based processors. Please note that like the GEMAP Point Source Model, the GEMAP Area Source Model is not a true emissions estimates model. It utilizes emissions estimates prepared by an external entity to generate gridded, hourly, criteria emissions estimates. During the emissions estimation process, the GEMAP Area Source Model can account for day-specific emissions events, regulatory control programs (control efficiency, capture efficiency, rule penetration, and rule-effectiveness), and area source specific operating parameters and schedules. Furthermore, the GEMAP Area Source Model can aid in the development of the spatial surrogates that are used to allocate area source emissions estimates from a county-wide basis to a cell-by-cell basis.

The GEMAP's Motor Vehicle Emissions Estimates Model (MoVEM) calculates gridded, hourly, link-specific emission estimates. MoVEM uses user-supplied vehicle activity data and US EPA emission factors (generated by MOBILE, or for California, generated by EMFAC) to estimate emissions. The activity data and emission factors are combined to obtain the final emission estimates. MoVEM contains the MOBILE model to calculate the necessary hour-specific emission factors for all states but California. A separate model, Cal-MoVEM, is under development, and it will be specifically used to estimate mobile source emissions for the state of California. A variety of activity data are needed from the user and include:

- Vehicle miles traveled (VMT) for a given road or region;
- The percentage of VMT accumulated by each of the MOBILE vehicle classes; and,
- Average vehicle speed over the road or in the region.

One of the principal design features of MoVEM is its ability to combine different types of vehicle activity data in order to estimate emissions. MoVEM can accept the following types of vehicle activity data:

- Urban-scale transportation model output;
- Regional or statewide transportation model output;
- Small-scale polygons of VMT, such as public land survey quarter section; and

- Countywide estimates of VMT.

Many urban regions included in modeling inventories have detailed link-specific VMT and speed estimates available from existing transportation models. The estimates are either daily, hourly, or peak period estimates. MoVEM assigns the activity data to the geocoded network and calculates the link-specific emission estimates, which are aggregated into gridded, hourly estimates.

However, the emissions modeling domain often extends beyond the urban boundary and beyond the limits of urban transportation models. In areas beyond the urban transportation model, activity data are usually available either from regional or statewide transportation models or from countywide estimates of VMT. Often, the countywide estimates are disaggregated by area type (urban, rural, etc.) and road functional class (interstate, arterial, collector, local road, etc.). In other cases, rural VMT is partially accounted for by regional or statewide transportation models that account for VMT on the higher road functional classes. MoVEM is designed to combine such regional VMT estimates with detailed geocoded urban or statewide transportation networks to calculate gridded, hourly emission estimates. The final output of MoVEM is a data set of emission estimates by grid cell, hour, pollutant, process, and technology type.

The GEMAP Biogenic Emissions Estimates Model (BIOME) computes gridded, hourly adjusted, pollutant-specific biogenic emission estimates using a combination of SAS- and ARC/INFO-based processors. BIOME utilizes satellite imagery-based or other land use/land cover data and species-specific biomass and emission factors to calculate emission rates at standard conditions (30°C and 800 $\mu\text{E}/\text{m}^2/\text{hr}$). BIOME also spatially allocates and temporally resolves the normalized emissions estimates for each hour of an episode day using gridded meteorological data. BIOME generates hourly-specific biogenic adjustment factors through the application of BIOLCM, an US EPA model that computes biogenic adjustment factors due to leaf canopy effects, temperature, humidity, solar intensity, and wind speed.

The GEMAP Speciation Model computes the chemical mechanism-specific emissions estimates based on the criteria emissions estimates supplied to it by the Point Source Model, the AREAS Source Model, BIOME, and MoVEM. The GEMAP Speciation Model splits the TOG component into the lumped model species classes that the air quality model uses. It also splits NO_x into NO and NO_2 , and SO_x into SO_2 . CO and PM are simply converted from mass-based values to mole-based values. The GEMAP Speciation Model currently supports the CB-IV and SAPRC chemical mechanisms. Finally, the GEMAP Speciation Model reformats the resulting speciated emissions estimates into a form that can be used by the air quality model. It currently supports the UAM, RADM, and SAQM air quality models.

GEMS

CMU

The GEMS (Geographic Environmental Modeling System) prototype consists of 75,000 lines of C++ code implementing a total of 150 classes. The development of the system was greatly aided by the use of the OMT modeling methodology for analysis and design and by the ability to reuse components from existing systems and class libraries. The GEMS user interface component and the visualization routines are implemented using X and the OSF/Motif tool kit. Motif widgets are accessed from C++.

GEMS was developed in a constant dialog with end users and domain experts in a prototyping process similar to the one we envision for the development of the CMS system. The development of the GEMS User Interface, for example, proceeded in several stages, each of which produced a prototype version built on the knowledge gained from the previous prototypes. The first iteration of the User Interface consisted of a set of sketches done by the graphical designer working directly with the client. The designer created a set of sketches (storyboards) illustrating a walk-through of what the eventual user of the GEMS system would see.

The storyboards were presented by the designer as part of the first prototype and the response of the clients was very positive. Both the other members of the development team and the clients were very impressed with the quality of the design produced by a professional designer as compared with their experiences in previous software development projects involving GUIs built only by the computer scientists.

The problem was to get the conceptual design, which existed only as a series of sketches (artists interpretation), implemented. The chosen platform for the final system was X11 and the OSF/ Motif tool kit. The development team had little expertise in GUI development, and X/Motif represented a sharp learning curve. It was decided that the graphical designer would continue making his ideas more concrete by using an animation tool, MacroMind Director, with which he was familiar and which was supported by the Design Department. The second prototype of the User Interface presented to the client(s) consisted of an animation running under the MacroMind Director program which allowed navigation through the entire User Interface but without any functionality behind the scenes.

In an ideal world, it should be possible to generate the code for the final user interface directly from the MacroMind animations; however, since MacroMind Director is available only on the Apple Macintosh and the rest of the GEMS system was implemented on X11 Unix machines, there was a conversion problem. In fact, MacroMind director does not create any executable code at all, and for this reason the animations became a set of executable requirements for the GEMS user interface – the user interface would be built to match the look and feel of the MacroMind animations, but would otherwise start from scratch.

Since we could not directly make use of the MacroMind Director work, we tried to ease the pain by making use of a high-level interface building tool kit. We spent considerable

development effort attempting to make use of the Interviews system, developed at Stanford University, to develop the GEMS interface. We ran into several problems with the InterViews system, including the lack of consistent documentation, disparities between the documentation and the implementation, lack of certain important pieces of functionality that we required, and compatibility problems between the InterViews code and the C++ compiler used for the rest of the system. We did implement a prototype user interface using the InterViews tool kit but finally decided to take the lessons learned from this experience and move to using the Motif tool kit directly instead of through InterViews.

Recent trends in the area of user interface development systems have been toward tools that allow the layout and the look-and-feel of a user interface to be created in a graphic package and then transformed directly into executable code and used in the final system. These developments can greatly simplify the process of user interface building, and will allow the effort currently spent on implementing the user interface to be redirected toward the creative aspects of making an interface pleasing and consistent. We feel that the problems we encountered with InterViews were due mostly to the immaturity of the concept and the product itself. We definitely feel that user interface builders and libraries of interface classes will become more prevalent and useful as the technology progresses.

Figure 3-3 in Chapter 3 of the CP shows the GEMS user interface, which is divided into four main windows: the Toolbox, the Main Map Window, the Overview Window, and the Graph Window. The Toolbox is where most of the interaction between the user and system takes place. It consists of a set of *tools* that provide a way to perform various kinds of analysis of the data available to the system. The Map Window is the central part of the display system. The data that makes up the map is divided into a series of layers. The data in each layer can have its own coordinate system and each layer provides conversions between the different coordinate systems in use. In this case the TIGER (Topologically Integrated Geographic Encoding and Referencing system) data is stored in Latitude/Longitude coordinates while the CIT Airshed Model grids are registered in UTM (Universal Transverse Mercator). The User Interface performs the necessary transformations between the coordinate systems used by the different layers and the coordinate system of the display. The map view tool contains a series of toggle buttons that specify whether a given layer is visible or not.

The first thing a user must do after starting up the system is choose a scenario from the list of available scenarios (the ones the system has data available for). In this case, the choice is by city. After the user has chosen the city to work with, the TIGER data for that city is loaded into memory and displayed on the Main Map Window. The initial TIGER display shows the chosen area, divided by county boundaries (as defined by the US Census Department). Clicking with the mouse button in the Main Map Window will select a county and display its name in the Status Line at the top of the window.

The Overview Window provides a way to zoom in on different portions of the map that may be of interest. Since the TIGER data is stored in vector format, the system can

magnify the data to any desired zoom level. The Find tool provides an Information button which displays a short description of the object(s) that is currently selected on the Map. Both the Name and the Information fields are common to all objects that make up the system, and the descriptions that each provides is set up by the database (in this case the TIGER database) when the object is first created.

Another tool from the Toolbox, the chemical lab tool, provides access to data about different Air Quality and Meteorology measurements (in this case those that make up the inputs to and outputs from the CIT Airshed Model). The information in this database is registered to a grid that overlays the area in question (Los Angeles in this case) and forms another layer on the Map. The user can now choose a portion of the Map (either by selecting an object from the TIGER layer(s) or dragging over a series of grid squares) and display Air Quality information for the chosen region. On the chemical lab tool, the user must select a date for which he wants the data displayed and must choose the chemical species he is interested in from the list of those species available from the database. The system takes this information and creates what is called a Map Overlay, another layer that sits on top of the TIGER and Airshed Grid layers. This layer consists of a series of Computation Grid objects, each of which contain values for a certain data element over a certain period of time. Each Grid is assigned a color according to the scale displayed at the right of the Overview Window, with blue representing low concentrations and red being high concentrations. To get a different view on this same data, the user might choose the Graph option under Output in the chemical lab tool and would be presented with a plot in the Graph Window showing the concentration of the chosen chemical (averaged over the chosen set of Computation Grids) throughout the chosen day. The chemical lab tool and the underlying data representation is designed to be very extensible so that it would be possible to incorporate a visualization system such as PV-Wave or AVS, and (given sufficiently powerful graphics capabilities) be able to create a 3D display of perhaps an ozone cloud over this same region of Los Angeles.

Another important aspect of the system is the capability to save any Plot or Map generated from the chemical lab, either for later review inside the GEMS system or for use outside the system. Plots can be saved as Postscript images or in the formats of several popular plotting packages. Maps can also be saved as Color Postscript or in any of several graphics formats. Since this system is intended for use in a regulatory environment where there is a need to be able to defend and reproduce a given piece of analysis, each output also contains a short report as to how the analysis was generated. For example, a plot might contain information as to which CIT Airshed Model files were used (including modification times and locations), how the plot was computed, and what assumptions were made in the computation. For more complex types of analysis, this report might also contain the relevant code that performed the calculation so that the work could be reproduced and verified using a different system.

Guideline Models

US EPA

The most widely used air quality models in the US are the EPA-guideline Gaussian plume dispersion models. These models consist of screening models (e.g., SCREEN2), flat terrain models (e.g., ISC), and models that deal with simple complex terrain conditions (e.g., RTDM, CTDM, COMPLEX I). All of the EPA-guideline plume models, along with their documentation, are available from EPA's SCRAM bulletin board.

The formulation of Gaussian plume models is very weak, assuming steady-state meteorological conditions for each hour of the year. Consequently, the scope of application for the models is very limited and they are only applicable to within a few 10s of km from the source over flat or rolling terrain. Not surprisingly, model performance evaluations of the Gaussian plume models indicate that they can not reproduce the observed concentration distribution for a given hour but do a better job in reproducing the annual frequency distribution of hourly observed concentrations with a conservative bias (tending toward overestimation).

The input parameters for regulatory applications of guideline plume models are set by EPA's guidance. There are very few decisions for the user to make in their application, mainly just the location of the receptors. Several user interfaces have been developed for the EPA plume models, with the user-friendly BREEZE system developed by Trinity Consultants probably the best known. Although the plume modeling systems are easy to use with user-friendly front-ends, the scientific formulation of the models is very poor with very simple description of transport and diffusion, almost no deposition, and virtually no treatment of atmospheric chemistry.

HYPACT

ASTER

HYPACT – the Hybrid Particle and Concentration Model – represents a state-of-the-art methodology for predicting the dispersion of air pollutants in 3D, mesoscale, time-dependent wind fields. HYPACT allows assessment of the impact of one or multiple sources emitted into highly complex local weather regimes, including mountain/valley and complex terrain flows, land/sea breezes, urban areas, and other situations in which the traditional Gaussian plume-based models are known to fail.

HYPACT, developed by ASTER, represents the next generation of dispersion modeling systems. It combines the best features of grid-based Eulerian dispersion methodologies with Lagrangian Particle Dispersion Modeling (LPDM).

HYPACT is a new generation model suitable for predicting the dispersion of pollutants in complex mesoscale, time dependent meteorological regimes. Species can include gases, radionuclides, and a spectrum of aerosol sizes. The 2D or 3D wind and turbulence field is provided by RAMS (for forecasts) or an observational network (for diagnostic application). HYPACT is ideal for regimes in which the assumptions underlying Gaussian plume-based models are violated, such as highly sheared flows, recirculating coastal and mountain/valley wind systems, urban heat islands, plume fumigation and bifurcation. Sources can be single or multiple, instantaneous (explosive), continuous, or time varying. Source geometry can include point, area, mobile, and line sources of various orientations. The model domain can extend from an area as small as an industrial plant site to hundreds of kilometers. The number of particles released is limited only by available memory and can exceed tens of thousands. Outputs displayed on interactive graphics terminals can include instantaneous and maximum surface concentrations and dosages, trajectories, time histories at arbitrary receptors, plume “centerline” values, streaklines, total path length pollutant burden, animated plume visualizations and source/receptor statistics.

Although the RAMS code can directly compute the dispersion of any number of “tracers” in an Eulerian framework, HYPACT has certain advantages because it combines in one code the best features of both the Lagrangian and Eulerian dispersion estimating methodologies. The advantage is greatest near a source region for tracers when the source is small and unresolvable on the Eulerian grid. A comparable Eulerian treatment would necessarily represent the source by a volume no smaller than one grid cell, and would immediately begin diffusing the tracer in adjacent cells. A Lagrangian approach, on the other hand, is fully capable of representing a source of any size, and of maintaining a concentrated, narrow plume downwind of the source until atmospheric dispersion dictates that it should broaden. In contrast, at large distances from the source, where the tracer plume is typically broad and well mixed, representation of the plume by Lagrangian particles can become inefficient due to the large number of particles required to achieve a smooth characterization of the plume. The hybrid Lagrangian and Eulerian approach used in HYPACT represents a tracer by Lagrangian particles near the source, but converts particles to Eulerian concentrations where appropriate at large distances downwind.

HYPACT is a modular code with new features being regularly added. The turbulent velocity components can be represented by either a first order Markov chain scheme or a fully random walk scheme. The initial features include a level 2.5 turbulence closure scheme. The dispersion and size sorting of heavy particles (typically diameters > 1 micrometer) are treated separately from gases and submicron aerosols. Liquid aerosol evaporation, linear chemical transformation and radiological decay can be specified. Dry deposition of gases and heavy particles are treated. Plume rise from buoyant and/or momentum sources can be taken into account using a variety of approaches. Concentrations are calculated using either a simple averaging over specified sampling volumes or, alternately, by using a more computationally efficient kernel density approach.

LMOS

Lake Michigan Region

The Lake Michigan Ozone Study (LMOS) is a multi-year effort to assist in the design of cost-effective and defensible emission control strategies for the four-state region surrounding Lake Michigan.

In spite of the previous and rather stringent emission controls, especially during hot summers, numerous exceedances of the 120 ppb ozone standard continue to occur. These violations are particularly frequent in the immediate vicinity of the lake itself, an indication of the role played by the lake breeze circulation. Both sources and receptors (population) are concentrated in a the narrow coastal zone experiencing the highest values.

Given the complex nature of the lake-induced effects, a state-of-the-art prognostic meteorological model (RAMS) was employed. An emissions model was developed collaboratively with SARMAP. Both these models provided input into an advanced version of the Urban Airshed Model (UAM-V). Moreover, a large-scale field observation program collected special weather and chemistry data at the surface and aloft. The overall budget is in excess of \$10 million.

These are some of the problems encountered in LMOS:

- Three different modeling groups had to communicate via phone and fax. Visuals (almost all black & white) were transmitted on paper, data on tapes.
- Each participating groups used different computer systems.
- In spite of best intentions, a common base map (or display format) was never achieved.
- All three groups used entirely different, and incompatible, visualizations.
- Each group was very familiar with its own module but knew little of the strengths and weaknesses of the others.
- Acquisition of input data (and data for evaluation) was extremely tedious and consumed a very large fraction of the overall budget.
- Even at the end of LMOS, there are still basically three software systems that are quite independent of each other.

MARS/MEMO (EUMAC Zooming Model) University of Thessaloniki

While the EUMAC model is generally considered and used as a regional model, a second modeling system has been developed to operate at more of the regional scale. This system, called either the EUMAC Zooming Model (EZM) or MARS/MEMO, was developed primarily by Moussiopoulos (1995), and includes a non-hydrostatic meteorological model (MEMO) linked to a photochemical model (MARS).

MEMO is a prognostic, non-hydrostatic model that solves the conservation of mass, momentum and scalar quantities (potential temperature, kinetic energy and humidity). It includes an inelastic approximation to filter sound waves. Terrain following coordinates are used, and varying grid spacing can be used in each direction. Turbulent diffusion is followed using a one equation model. Soil temperature is solved using a 1D heat equation for the soil. MEMO allows multiple nests.

MARS is a 3D photochemical model. It's general formulation is similar to other photochemical models, and like the CIT model, it treats vertical diffusion and chemistry together (i.e., those two processes are not split). At present, MARS uses the KOREM mechanism.

The EZM has been applied in a variety of domains including Athens and Thessaloniki, Greece, Barcelona and Lisbon. More information about the EZM and its applications can be obtained from Prof. N. Moussiopoulos at Aristotle University, Thessaloniki, Greece.

Reference

Moussiopoulos, N. (1995). "Air Pollution Models as Tools to Integrate Scientific Results in Environmental Policy," *Air Pollution Theory and Simulation*, H. Power, N. Moussiopoulos, and C.A. Brebbia (eds), Computational Mechanics Publications, 10-18.

MM5**PSU/NCAR**

The non-hydrostatic Penn State University/National Center for Atmospheric Research (PSU/NCAR) mesoscale model, now widely known as MM5, is a 3D nested-grid primitive-equation meteorological model designed for a wide variety of atmospheric applications including air-quality studies. It is described in detail by Dudhia (1993) and Grell *et al.* (1994). The model is written in a terrain-following sigma (non-dimensionalized pressure) vertical coordinate and uses a split semi-implicit temporal integration scheme. Prognostic equations predict the 3D wind components (u , v , and w), temperature (T), water vapor mixing ratio (qv), and the perturbation pressure (p'). The perturbation pressure is the departure from a temporally invariant reference-state pressure described by Dudhia (1993). Use of a reference state plus perturbations to describe the total pressure is important for reducing numerical pressure-gradient errors that occur in almost all terrain-following coordinate systems, so that the MM5 solutions remain accurate in regions of steep topography.

The physical parameterizations in the model provide many options to the user. The model contains the multi-layer Blackadar planetary boundary layer (PBL) and Gayno-Seaman 1.5-order closure PBL schemes to represent turbulent fluxes of heat, moisture and momentum (Zhang and Anthes 1982; Gayno 1994). A surface energy budget equation predicts the ground temperature (T_g) and includes the effects of insulation, atmospheric path length, water vapor, cloud cover and longwave radiation. The surface physical properties of albedo, roughness length, moisture availability, emissivity and thermal inertia are defined as a function of land use for 13 categories via a look-up table. A column radiation model also is included in the MM5 (Dudhia, 1989).

Precipitation options include several deep convective parameterizations. These are the Anthes-Kuo scheme (Anthes 1977; Anthes *et al.* (1987), the Betts-Miller scheme (Betts and Miller 1986), the Grell Scheme (Grell 1993) and the Kain-Fritsch scheme (Kain and Fritsch 1993). Each of the four convective schemes normally is applied in hybrid form along with a resolved-scale precipitation scheme that explicitly predicts cloud water and rain water concentrations (Hsie and Anthes 1984) with simple ice physics included (Dudhia 1989). For economy, the MM5 model may also be run in a dry mode without precipitation physics.

The model may be configured with nested grids having either one-way interactive or two-way interactive mesh interfaces and up to ten separate domains. The nested grids have a mesh ratio of three to one. Pre-processing software allows the generation of detailed initial and lateral boundary conditions based on background analyses from the National Meteorological Center, plus observations. The model's top level can be set arbitrarily at any pressure level, although 100 mb is the most commonly used top level to ensure that vertically propagating wave energy is damped in the stable stratosphere. Horizontal and vertical resolution, and the number of grid points, is completely arbitrary. Typical applications may provide greater vertical resolution in the lowest 1-2 kilometers where surface heating and orography have the greatest influence on mesoscale flows.

The Gayno-Seaman 1.5-order PBL scheme is designed to allow accurate predictions with the lowest model calculations performed at 10 m AGL.

The MM5 also includes a multi-scale 4D data assimilation (4DDA) technique based on Newtonian relaxation, or nudging. The 4DDA scheme is a continuous assimilation method that relaxes the model state toward the observed state by adding to one or more of the prognostic equations artificial tendency terms based on the difference between the two states. It is said to be a form of continuous data assimilation because the nudging term is applied at every time step, thereby minimizing “shock” to the model solutions that may occur in intermittent assimilation schemes. The multi-scale 4DDA technique was developed by Stauffer and Seaman (1994) and includes simultaneous use of two approaches outlined in Stauffer and Seaman (1990) and Stauffer *et al.* (1991): (1) nudging toward gridded analyses interpolated to the model's current time step, and (2) nudging directly toward individual observations. These two approaches are referred to as “analysis nudging” and “obs nudging”. Analysis nudging is ideal for assimilating synoptic-scale data that cover most or all of a model domain. Obs nudging does not require gridded analyses of observations and is better suited for assimilating high-frequency asynoptic data that may be distributed non-uniformly in space and time. Nudged variables normally include winds, temperatures and water vapor. This versatile 4DDA system is very valuable for supporting a range of air-quality studies.

References

- Anthes, R.A. (1977). A cumulus parameterization scheme utilizing a one-dimensional cloud model. *Mon. Wea. Rev.*, **105**, 270-286.
- Anthes, R.A., E.-Y. Hsie, and Y.-H. Kuo (1987). Description of the Penn State/NCAR Mesoscale Model Version 4 (MM4). NCAR Tech. Note, NCAR/TN-282+STR, 66 pp.
- Betts, A.K. and M.J. Miller (1986). A new convective adjustment scheme. Part II: Single column tests using GATE wave, BOMEX, ATEX and Arctic air-mass data sets. *Quart. J. Roy. Meteor. Soc.*, **112**, 693-709.
- Dudhia, J. (1989). “Numerical study of convection observed during the Winter Monsoon Experiment using a mesoscale two-dimensional model,” *J. Atmos. Sci.*, **46**, 3077-3107.
- Dudhia, J. (1993). A Nonhydrostatic Version of the Penn State-NCAR Mesoscale Model: Validation Tests and Simulation of an Atlantic Cyclone and Cold Front. *Mon. Wea. Rev.*, **121**, 1493-1513.
- Gayno, G.A. (1994). Development of a Higher-Order, Fog-Producing Boundary Layer Model Suitable for Use in Numerical Weather Prediction. M.S. Thesis, The Pennsylvania State Univ., 104 pp.
- Grell, G.A. (1993). Prognostic evaluation of assumptions used by cumulus Command: parameterizations. *Mon. Wea. Rev.*, **121**, 764-787.
- Grell, G.A., J. Dudhia, and D.R. Stauffer (1994). A Description of the fifth generation Penn State/NCAR mesoscale model (MM5). NCAR Technical Note, NCAR/TN-398+STR, 138 pp.
- Hsie, E.-Y. and R.A. Anthes (1984). Simulation of frontogenesis in a moist atmosphere using an alternative parameterizations of condensation and precipitation. *J. Atmos. Sci.*, **41**, 2701-2716.

- Kain, J.S. and J.M. Fritsch (1993). Convective parameterization for mesoscale models: The Kain-Fritsch scheme. The representation of cumulus convection in numerical models, K.A. Emanuel and D.J. Raymond (eds), *Amer. Meteor. Soc.*, 246 pp.
- Stauffer, D.R. and N.L. Seaman (1990). Use of Four-Dimensional Data Assimilation in a Limited-Area Mesoscale Model. Part I: Experiments with Synoptic Data. *Mon. Wea. Rev.*, **118**, 1250-1277.
- Stauffer, D.R., N.L. Seaman, and F.S. Binkowski (1991). Use of Four-Dimensional Data Assimilation in a Limited-Area Mesoscale Model. Part II: Effects of Data Assimilation Within the Planetary Boundary Layer. *Mon. Wea. Rev.*, **119**, 734-754.
- Stauffer, D.R. and N.L. Seaman (1994). Multiscale Four-Dimensional Data Assimilation. *J. Appl. Meteor.*, **33**, 416-434.
- Zhang, D.L. and R.A. Anthes (1982). A High-Resolution Model of the Planetary Boundary Layer - Sensitivity Tests and Comparisons with SESAME-79 Data. *J. Appl. Meteor.*, **21**, 1594-1609.

Models-3

HPCC/US EPA

The Models-3 development effort being pursued under the High-performance Computing and Communications (HPCC) program at the U.S. Environmental Protection Agency (EPA) has goals very similar to those of CAMRAQ. This is not surprising, since both efforts were outgrowths of EPRI-organized workshops in late 1990 and early 1991 that brought atmospheric and computer researchers together to formulate the need for and desired characteristics of a "Complete Modeling System" for atmospheric research and policy analyses. As EPA's primary effort under the HPCC initiative, development of Models-3 has enjoyed a considerably higher level of funding than that for CAMRAQ's CMS framework design. This is reflected in the disproportionate number of design and assessment documents, relative to those from CAMRAQ, that have been produced as part of Models-3 development under a variety of grants, cooperative agreements, and contracts. Unfortunately, most of the documents are not publicly available. The major goals, as listed in their Concept Paper document (Novak *et al.*, 1995), are these:

- To provide more effective solutions for multipollutant and multimedia assessment efforts
- To provide key federal, state, and industrial users with a computational and decision support environment that is easy to use and responsive to environmental problem-solving needs
- To advance the capability of environmental assessment tools by adapting them to a heterogeneous, distributed computing environment that includes massively parallel architectures

The second goal above includes the development of a framework that allows the users of the system to utilize it in their daily activities. The description of this framework as provided in the Models-3 documentation is similar in many respects to the requirements listed in the CP for the CMS.

The CMS effort and the Models-3 work complement each other. The CMS and Models-3 development groups are collaborating to prevent a duplication of effort and make most efficient use of the resources and expertise available to both groups. This interaction will continue throughout the development of the CMS and will be valuable to both groups.

Both groups see that it is necessary to build a community consensus on some of the basic standards for communications and module development that can be used by both the Models-3 and CMS efforts. A close interaction between the groups and an open exchange of ideas will allow members of the environmental modeling community to make use of the modules from both systems that are most appropriate to their particular problem and link them together seamlessly.

Reference

Novak, J.H., R.L. Dennis, D.W. Byun, *et al.* (1995). EPA Third Generation Air Quality Modeling System, Volume 1: Concept. EPA/600/R95/084, U.S. EPA, Research Triangle Park, NC.

MONTECARLO

FaAA

MONTECARLO is a Lagrangian particle model for the simulation of transport and fate of pollutants in the atmosphere. MONTECARLO is written in Fortran-77 and has been successfully run on workstations, Power PCs, and Pentium-based PCs. MONTECARLO was developed as an improvement of the MC-LAGPAR III model (Zannetti, 1986; Graziani and Mosca, 1992). It accepts complex terrain, full 3D meteorological input, and "zooming" regions with higher terrain resolution. The model provides unique, high-resolution simulations of continuous or instantaneous releases under time-varying, non-homogeneous atmospheric conditions. Point, segment, area, and volume emissions can be used.

The model includes space-dependent, time-dependent linear chemistry, e.g., for SO₂-to-SO₄ conversion inside a plume. Similarly, dry and wet deposition are accounted for by using a linear function with space-dependent, time-dependent rates.

MONTECARLO calculates concentrations by either superimposing a concentration grid and counting the particles in each cell, or by using the kernel method.

A user's manual of the model is available (IDEA, Inc., 1995).

References

- Graziani, G. and S. Mosca (1992). "Different Schemes for Turbulence in a Lagrangian Particle Model," *Computer Techniques in Environmental Studies IV*, P. Zannetti, ed., Elsevier Applied Sciences, 61-72.
- IDEA, Inc. (1995). "PDM: Particle Dispersion Model – User's Manual," prepared for Environment Institute under Contract 10379-94-08 F1ED ISP I.
- Zannetti, P. (1986). "Monte-Carlo Simulation of Auto and Cross-Correlated Turbulent Velocity Fluctuations (MC-LAGPAR II Model)," *Environmental Software*, 1, 26-30.

Operational Multiscale Environment model with Grid Adaptivity (OMEGA)

SAIC

SAIC's Applied Physics Operation (APO) has developed a revolutionary atmospheric simulation tool for aerosol transport. The new model, the Operational Multiscale Environment model with Grid Adaptivity (OMEGA), uses a novel unstructured, adaptive grid approach which is designed to resolve the local atmospheric effects of terrain and other surface features while maintaining a simulation of the larger scale weather features.

OMEGA represents a significant advance in the field of weather prediction. Operational atmospheric simulation systems in common use today do not provide the resolution necessary to incorporate the effects of the underlying surface. For this reason, a local forecaster in the United States must interpolate the local forecast from National Meteorological Center forecasts guidance (grid point data at 90 km intervals). The forecaster also incorporates his experience and knowledge of local effects to arrive at the final forecast. Incorporating this "local knowledge" is not feasible in remote areas of the world when time is a factor. The simulation must incorporate these local features. Operational forecast models in current use are scale specific; their fixed rectangular grid structure limits the resolution of both the input boundary conditions and the resulting atmospheric simulation. The major advantages of OMEGA over the current state-of-the-art include the ability to resolve the surface terrain down to scales of 1 km and along with that the local perturbations on the larger scale wind field. This local wind field perturbation is of extreme importance in determining the trajectory of an aerosol release. In order to calculate this local perturbation, however, it is important to include all of the physical parameters and processes which affect the local flow.

OMEGA forecasts wind, temperature, moisture, cloud elements, and precipitation by solving a system of non-hydrostatic equations on the unstructured grid. The structure of the grid may be adapted to topographic features such as terrain and land/water boundaries or it may be adapted to atmospheric phenomena such as frontal regions or areas of vertical instability that are favorable to storm development. The OMEGA grid resolution can range from 100 km to 1 km in the horizontal and a few tens of meters to 1 km in the vertical. The simulation of the surface in the model includes a complete surface energy budget and a soil hydrology model. These modules include the effects of topography, land use, land/water composition, vegetation, soil moisture, and snow cover. The inclusion of this physics in a multiscale model represents an additional advance in the state-of-the-art. SAIC scientists have also incorporated an integral aerosol transport model into the design of OMEGA to provide subgrid transport simulation.

The left side of Figure 1 shows the OMEGA grid over the Florida peninsula as depicted within the SAIC X-window analysis interface developed for OMEGA. Notice the enhanced grid resolution along the coastline and around Lake Okeechobee. The enhanced resolution allows for a more accurate representation of the sea breeze circulation. The right side of Figure 1 shows a constant Southwest flow of 6 m/s which

was used for this idealized simulation. Figure 2 depicts the simulated near-surface winds for 1000, 1300, 1600, and 1900 LST (3, 6, 9, and 12 hours, respectively, after the model initiation). Figure 2 also shows the tracers from three simulated continuous plumes.

OMEGA represents a revolutionary atmospheric simulation tool for multiscale applications. While obviously aiding the sciences of numerical weather prediction, the model will also have valuable applications in numerous aerosol transport problems, ranging from urban air quality and compliance with US Environmental Protection Agency regulations to emergency response to toxic releases. Figure 3 shows an OMEGA grid adapting to pollution sources.

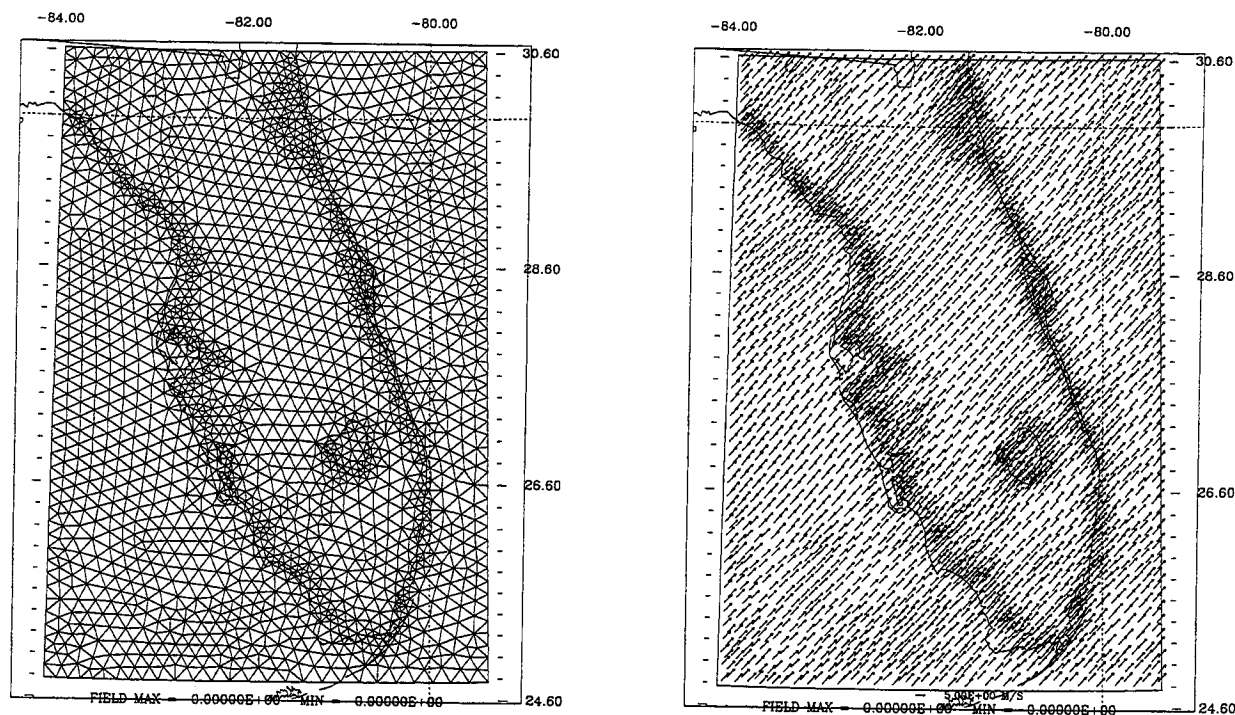


Figure 1. Simulation of aerosol transport and diffusion in an idealized Florida sea breeze circulation using OMEGA. The left side of the figure shows the initial grid while the right side shows the initial wind field for this simulation. Initialization occurred at 7 AM local time.

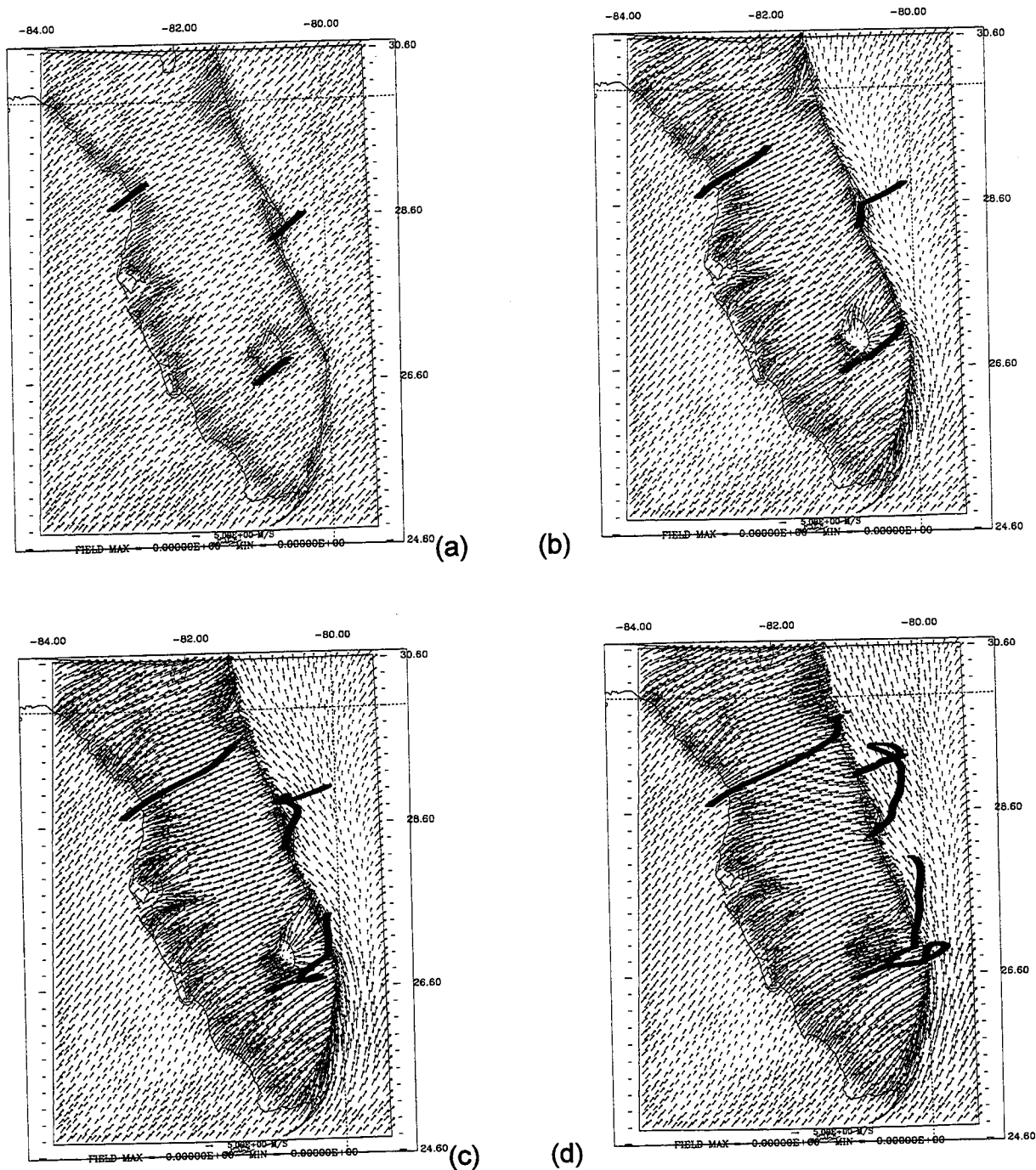


Figure 2. Simulation of aerosol transport and diffusion in an idealized Florida sea breeze circulation using OMEGA. The location of the centroids of the puff releases is shown after 3, 6, 9, and 12 hours of simulation.

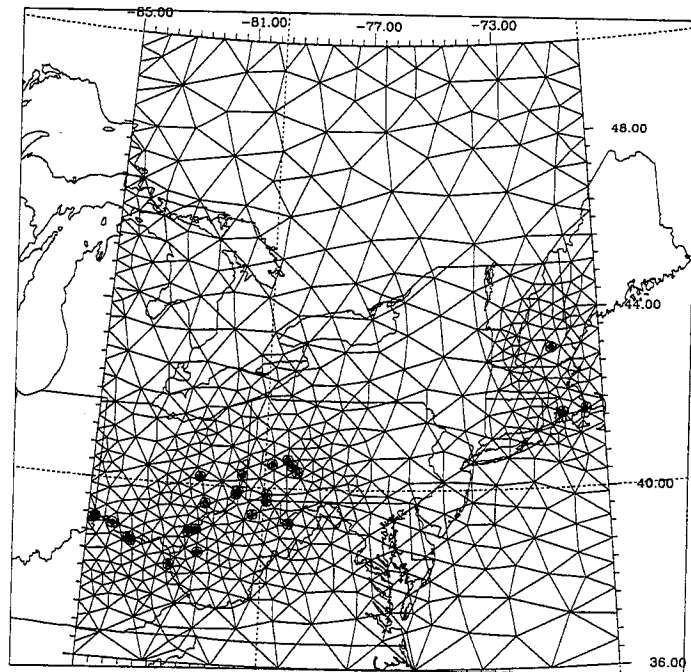


Figure 3. OMEGA grid adapting to the 1990 Regional Interim Emissions Inventory. Higher resolution in the vicinity of emissions sources producing more than 25,000 TPY of SO₂.

RAMS

MRC/ASTER

RAMS – the Regional Atmospheric Modeling System – represents the state-of-the-art in general purpose regional, mesoscale, and cloud-scale atmospheric simulation and prediction software. RAMS resulted from the merger of the cloud-scale, non-hydrostatic model developed by Dr. William R. Cotton with the mesoscale primitive equation code developed by Dr. Roger A. Pielke. Mission Research Corporation (MRC)/ASTER Division holds the exclusive marketing license from Colorado State University for the RAMS code.

RAMS is in fact a series of interconnected modules that allow simulation/prediction of atmospheric phenomena in domains ranging from global to cumulus cloud scale. Only those options required for a specific purpose need be employed. Either a full featured or user-customized configuration of RAMS along with extensive training, code maintenance and user support are available. ASTER provides the latest versions of RAMS which have been under development for over 20 years, and which will continue to advance as new physical modeling and numerical techniques are developed and tested. RAMS has been widely used, tested and documented in over 500 technical publications, including studies of land/sea breezes, convective storms, weather modification, soil/vegetation/atmospheric interactions, air pollution dispersion and emergency response, mesoscale temperature forecasts, large eddy simulations and air flows over complex terrain. RAMS currently also serves as an operational regional weather prediction system.

Initializing data can be as minimal as a single local rawinsonde or profiler. Alternately, RAMS initialization can use an entire mesoscale database or, more commonly, nest within grid points fields provided by NMC or other global models providing non-homogeneous initial and non-stationary boundary conditions. 4D data assimilation (4DDA) is available.

Model configurations include 1D, 2D, or 3D, numerous (> 40 if needed) layers in the vertical, detailed planetary boundary layer representations, two-way multiple nested interactive grids (in both vertical and horizontal). Horizontal grid sizes can range from < 2 m (simulating flow around buildings) to > 100 km (global circulation modeling). RAMS has hydrostatic or non-hydrostatic modes, and can employ uniform or variable land use, topography, roughness, soil moisture and water temperature. There are selectable options for turbulence closure, finite difference schemes, geographic coordinate systems, upper and lateral boundary conditions and more.

RAMS generates basic atmospheric variables (wind, temperature, pressure, moisture) at each model grid point and time step. From these a wide variety of parameters can be derived, including turbulence, vorticity, stability indices, sound propagation, air density, refractive indices, cloud liquid water, precipitation rate, etc.

RAMS is designed to serve as a front end to a variety of application modules. These include predicting the environmental impact of gaseous and aerosol pollutants, the

RAMS is designed to serve as a front end to a variety of application modules. These include predicting the environmental impact of gaseous and aerosol pollutants, the transport of radionuclides in complex atmospheric flows, the environmental component of electrical energy consumption, convective storm development, etc. An interactive graphic system can be used to display and animate RAMS generated fields in a variety of formats.

The diversity of RAMS users include National Laboratories, military test ranges, nuclear safety offices, university research teams and commercial weather forecasting services.

Regulatory Models

Trinity Corporation

Trinity, in many ways, is one of the most successful developers of a preliminary modeling system. They have taken a suite of the more basic models, e.g., Gaussian Plume Models, that are used for permitting and similar applications, and bundled them together in a fashion that is more readily accessible than their standard counterparts. In many ways, the CMS envisioned here is a similar, but much more involved, process. In the case of the Trinity Modeling System, the various models are made readily accessible using engineered user interfaces. Data is more standardized. Model results can be readily visualized, and report generation is facilitated.

Access to the Trinity modeling suite, or similar capabilities would be a desirable aspect of a regional CMS to increase the potential uses. Also, the capabilities provided by the CMS can be utilized by the more basic models, e.g., development of meteorological and emissions fields. Such fields, with an appropriate I/O API could be used relatively seamlessly to drive the Gaussian plume models, e.g., for permitting and SIP preparation.

ROM/UAM Modeling System

US EPA

The current EPA regulatory version of the Urban Airshed Model (UAM) along with a set of pre- and postprocessors and documentation was delivered to the EPA in 1990. The UAM modeling system can be easily obtained by downloading it from the EPA SCRAM bulletin board. This system includes Version 2.0 of the Emissions Processing System (EPS 2.0), the Diagnostic Wind Model (DWM), and the ROM (Regional Oxidant Model)/UAM Interface. Along with the software and documentation, EPA has also published guidelines for the regulatory application of the UAM that delineate the procedures to be used when applying the UAM for regulatory decision making (e.g., a SIP), including procedures for a comprehensive model performance evaluation.

When available, the EPA guidelines recommended use of the ROM/UAM interface system for developing the UAM initial concentrations and boundary condition (IC/BC) inputs. This is accomplished through use of the ROM/UAM interface system software. The portions of the ROM inputs and outputs needed for the ROM/UAM interface system are available from the Gridded Model Information Support System (GMISS) on EPA's IBM mainframe computer system. The ROM model itself has only recently been made available by the EPA. However, the science and numerics in the ROM/UAM modeling are 5 to 10 years old. Furthermore, the ROM model has not yet been subjected to the level of comprehensive model performance evaluation as required for the regulatory models (e.g., UAM).

During the initial stages in the development of the UAM modeling system there was a concerted effort to develop a set of software programs that easily linked with each other. However, later on the focus was to bring newer technology into the UAM applications (e.g., use of the DWM) with no resources allocated to integrate the new modules into the UAM system. The resources allocated to the delivery of the UAM modeling system in 1990 were sufficient solely for documentation. Very few resources were available for streamlining the system or for providing user-friendly interfaces. The exception to this is for the EPS and GMISS where the EPA developed graphical user interfaces (GUI). Unfortunately, these GUIs were developed in SAS, the GMISS GUI is limited to EPA's IBM mainframe, and the EPS GUI use is also very limited.

SARMAP Air Quality Modeling System**SJVAQS/AUSPEX**

As part of the San Joaquin Valley Air Quality Study (SJVAQS)/Atmospheric Utilities Signatures: Predictions and Experiments (AUSPEX) Regional Model Adaptation Project, RADM and MM4 were extended to provide additional capabilities (the names of the updated models are SAQM [SARMAP Air Quality Model] and MM5, respectively). In addition, GEMAP (Geocoded Emissions Modeling and Projections,) was developed jointly with the Lake Michigan Air Directors' Consortium (LADCO) and applied in the modeling region to obtain modeling emissions inventories. Both MM5 and GEMAP are described elsewhere in this appendix. In developing SAQM, a number of improvements or changes with respect to RADM were made:

- A fixed vertical coordinate system was incorporated into the model to make its vertical grid structure compatible with the vertical grid structure of MM5.
- Its mass conservation module was updated to make it compatible with the non-hydrostatic meteorological input.
- The advection scheme was updated from Smolarkiewicz's (1983) scheme to Bott's (1989a, 1989b) scheme to reduce numerical diffusion and increase numerical accuracy.
- Its dry deposition module was updated based on findings from the 1991 San Joaquin Valley Deposition Study (Massman et al., 1994)
- Its chemical mechanism was changed from the RADM2 mechanism to a choice of either of two mechanisms: the Carbon Bond Mechanism, version 4 (CBM-IV) and the Statewide Air Pollution Research Center (SAPRC) mechanism (Carter, 1990).

The following description of SAQM is adapted from the draft final report to the SARMAP sponsors prepared by Julius Chang of the State University of New York at Albany.

SAQM is a three-dimensional regional scale nonhydrostatic air quality model (Jin and Chang, 1994). It is based on the modeling framework of the Regional Acid Deposition Model (RADM) (Chang et al., 1987) with some fundamental changes to allow the model to have much wider applications rather than to be limited to the condition of hydrostatic balance. For each atmospheric trace species considered in the model, its concentration is governed by atmospheric transport, source emissions, deposition removal, and chemical transformation. The corresponding constituent mass conservation equation is:

$$\frac{\partial C}{\partial t} = -\nabla \cdot (\bar{V}C) + \nabla \cdot (K_e \nabla C) + P_{chem} - L_{chem} + E + \left(\frac{\partial C}{\partial t} \right)_{cloud} + \left(\frac{\partial C}{\partial t} \right)_{dry} \quad (1)$$

where C is the chemical trace species concentration, \bar{V} is the three-dimensional velocity vector at each grid point in the model domain, K_e is the eddy diffusivity used to parameterize the subscale turbulent fluxes of trace species, P_{chem} and L_{chem} are the production and loss rates due to chemical reactions, E is the source emission rate,

$(\partial C / \partial t)_{cloud}$ is the time rate of change of concentration due to cloud effects, and $(\partial C / \partial t)_{dry}$ is the rate of change due to dry deposition.

The Non-hydrostatic Eulerian Transport Equation

SAQM's horizontal coordinate system uses the Lambert conformal projection. Vertically, SAQM uses a terrain-following non-hydrostatic s coordinate system. The vertical coordinate σ is defined as a function of a selected reference state of the atmosphere which does not change with time and has the following form:

$$\sigma = \frac{P_0 - P_{T0}}{P_{S0} - P_{T0}} \quad (2)$$

where P_{T0} is the reference pressure at the top of the model and is set to 100 mb, P_0 is the reference pressure at a given point (x, y, z) , and P_{S0} is the reference pressure at the surface at position (x, y) .

The transport equation under the nonhydrostatic s coordinate system is derived through the continuity equation and has the form of

$$\begin{aligned} \frac{\partial C^*}{\partial t} = & -m^2 \left[\frac{\partial}{\partial x} \left(\frac{u C^*}{m} \right) + \frac{\partial}{\partial y} \left(\frac{v C^*}{m} \right) \right] - \frac{\partial}{\partial \sigma} (W C^*) \\ & + m^2 \frac{P^*}{\rho_0} \left[\frac{\partial}{\partial x} \left(\rho K_h \frac{\partial}{\partial x} \left(\frac{C}{\rho} \right) \right) + \frac{\partial}{\partial y} \left(\rho K_h \frac{\partial}{\partial y} \left(\frac{C}{\rho} \right) \right) \right] \\ & + \left(\frac{g}{P^*} \right)^2 \frac{\partial}{\partial \sigma} \left(\rho_0 \rho K_z \frac{\partial}{\partial \sigma} \left(\frac{C}{\rho} \right) \right) \end{aligned} \quad (3)$$

where $C^* = (P^* / \rho_0) C$; C is the trace species concentration in density units, such as molecules/cm³; ρ_0 is the air density of the reference state; $P^* = P_{S0} - P_{T0}$; m is the ratio of the transformed distance on the Lambert conformal map projection to the true distance; g is the gravitational acceleration; ρ is the density of air; K_h and K_z are the horizontal and vertical eddy diffusivities; u and v are the horizontal wind velocity components in x and y directions, and W is the vertical velocity defined in the non-hydrostatic s coordinate system and has the form of

$$W = -\frac{\rho_0 g}{P^*} w - \frac{\sigma m}{P^*} \left(u \frac{\partial P^*}{\partial x} + v \frac{\partial P^*}{\partial y} \right). \quad (4)$$

It should be noted that variable C/ρ has been used for eddy diffusion calculations in equation (3), instead of density C . These forms of eddy diffusion can be easily derived through first order K-theory approximation of turbulent fluxes by neglecting the smaller density perturbation term [Stull, 1988; Venkatram, 1993]. The use of mixing ratio for the eddy diffusion terms guarantees the conservation of mixing ratio and ensures that tracers are being mixed properly with its carrier. In theory, using tracer density C for eddy diffusion calculations would be fine if the tracer carrier, the air, is also being

mixed in the same manner both in formulation and in numerical computation. But this has not been the case in most of the existing air quality models. Air density is normally given by the meteorological model or interpolated from observations. When that is coupled to air quality model eddy mixing algorithm one can no longer guarantee that an inert tracer at equilibrium has constant mixing ratio vertically.

The vertical velocity W defined in equation (4) is a function of all three-dimensional wind components u , v , and w . The three-dimensional wind and mass fields used by the air quality model are generally hourly data from a mesoscale meteorological model or observations. When these hourly wind fields are interpolated to the time step of advection calculation, the three wind components, u , v , and w , are no longer guaranteed to be consistent among themselves. Thus the calculated vertical velocity, W , becomes inconsistent with the horizontal wind components, u and v , as well. This inconsistent wind fields will create numerical oscillations in trace species concentrations when advection calculations are carried out in the model.

Mass-inconsistent problem can be caused either by spatial and time interpolations of the meteorological fields or by using different numerical schemes in solving the transport equations between the meteorological model and the air quality model. Considerable efforts have been made since the first recognition of the mass-inconsistent problem between wind and mass fields [Sasaki, 1958]. To date, the widely used approach for solving the mass-inconsistency problem is to directly adjust the wind fields. In all the current mass-consistent wind field adjustment models, it is generally assumed that air is incompressible so that only hourly adjustment to the wind fields is needed. When the incompressibility of air or wind field is no longer valid, a step by step adjustment to the wind fields becomes necessary. This is computationally very expensive. In fact, it is virtually impossible for an air quality model to carry such a complex and time consuming wind field adjustment model economically.

Jin et al. [1993] developed a new mass-inconsistency correction scheme for three dimensional air quality models based on the work of Kitada [1987]. By concurrently computing the tracer species' transport and medium transport, the proposed correction scheme is able to eliminate numerical oscillations generated by the mass-inconsistent wind fields. This methodology has been employed in SAQM. Experimental studies with uniform tracer mixing ratio in SAQM show less than 0.01% changes in all the model grids after 24 hours of transport calculation. With this error correction scheme SAQM can use meteorological data from diverse sources including operational weather forecasts. Further, this also allows for grid refinements so that SAQM can be used for urban air quality study when meteorological data on grid dimensions of a few kilometers must be interpolated from lower resolution data base.

Chemical Mechanisms and Emissions

SAQM has three different gas-phase chemical mechanisms, CBM-IV (Whitten et al., 1980), SAPRC (Lurmann et al., 1986) and RADM2 (Stockwell et al., 1990). All other

components of these different versions of SAQM are identical except for those details associated with the differences in chemical speciation. Among these mechanisms the inorganics are very much alike with only minor differences. It is the representation of the organics that differentiates these mechanisms. At each grid point of the simulation domain, SAQM predicts time-varying concentrations of 40 to 60 trace gases depending on the version of the model. It is interesting to note that each of these mechanism has been tested against almost all the smog chamber data and all have been judged as acceptable (Dodge, 1989). Therefore any differences among these models must be considered as the uncertainty of current knowledge. Indeed, recent calculations have shown that while ozone amount at many locations as predicted by SAQM/CBM-IV and SAQM/SAPRC are very similar, the amount of H₂O₂ as predicted by these two models can be very different. While it is tedious to present a listing of all three mechanisms, the following table gives an overview of the differences in addition to the difference in philosophy for lumping organics species. Our experience indicates that the smaller organics speciation of the CBM-IV mechanism does allow that version of SAQM to be almost a factor of two faster than the SAPRC version and the RADM2 version is about the same speed as the SAPRC version.

Table 1. Summary of three gas-phase chemical mechanisms

	SAPRC	CBM-IV	RADM2
Number of reactions	135	87	157
Number of photolytic reactions	16	11	21
Number of reactive species	50	36	58
Organics			
alkanes	3	2	5
alkenes	5	3	4
aromatics	3	3	3
aldehydes	3	2	2
other carbonyls	3	2	4

Emissions supplied to SAQM include mobile sources, point sources, area sources, and biogenics from agriculture and forests with grid resolution of 4 km. The emissions are processed separately for SAPRC and CBM-IV chemical mechanisms according to the mechanism specific organic lumping procedures. Unlike NAPAP emissions generated for RADM, SARMAP emissions are classified into source categories before they are combined for model input. These separate emission source categories allow changes of individual emission sources, such as mobile sources, and made model sensitivity studies much more convenient and meaningful. In addition to the emission category classification, the exact geographical locations of stacks are provided instead of the center points of emission grids where the stacks belong. These accurate stack coordinates allow us to use model grids smaller than emission resolutions to study the effect of individual plumes.

Numerical Algorithms for Transport Calculations

The atmospheric transport processes as represented in equation (3) consist of two parts: advection and eddy diffusion. For the advective transport calculation, there are many modern schemes available for use, such as the simple positive definite advection scheme of Smolarkiewicz as used in RADM [Smolarkiewicz, 1983], the Bott's high order polynomial scheme [Bott, 1989a, 1989b], the Prather second-order moment scheme [Prather, 1986], and the semi-Lagrangian approach [Smolarkiewicz and Pudykiewicz, 1992]. The choice of a numerical advection scheme for a specific application involves the consideration of numerical accuracy, mass conservation, computational efficiency, and the overall computer memory requirement. In SAQM, Bott's advection scheme has been implemented based on its overall merits as compared to the other schemes. In the horizontal direction, operator splitting technique with Bott's fourth order scheme for uniform grid-size is used. Because of the variable vertical grid-size a new second scheme for nonuniform grids has been

derived. It was found to be more accurate than what was used in RADM in maintaining sharp vertical gradients in tests of trace species distributions.

The vertical eddy diffusion calculation is based on K-theory for non-convective atmospheric conditions as described in Chang et al. [1987]. Under convective conditions, the non-local closure asymmetrical mixing scheme developed by Pleim and Chang [1992] has been implemented. The atmospheric stability is determined by the calculated Richardson number according to temperature and wind profiles provided by MM5.

Horizontal eddy diffusion has been considered in SAQM simulation. With the highly accurate Bott's scheme there is very little numerical diffusion hence it is important to include the physical diffusion. In SAQM, a value of $50 \text{ m}^2 \text{ s}^{-1}$ is being used for the horizontal diffusion calculations, according to a number of sensitivity studies with the model. For horizontal grid-size smaller than $12 \times 12 \text{ km}^2$ this value must be reduced appropriately to be compatible with the grid size.

Other Model components

Dry deposition fluxes are computed for all the relevant species (up to 13 species) by multiplying trace species concentrations in the lowest model level by species-specific deposition velocities, which are parameterized in terms of atmospheric stability, land type, season, insolation, and surface wetness.

Effects of clouds, including subgrid-scale vertical redistribution, aqueous chemistry, and trace gas and particle scavenging, are parameterized using a simple one-dimensional cloud model (Walcek and Taylor, 1986) and a local equilibrium aqueous chemistry and scavenging submodel. The presence and characteristics of clouds are parameterized in terms of precipitation rate and vertical profiles of temperature and moisture.

Nested SAQM

A two-way nesting technique coupling models with different geographic scales was developed. In this manner, the small scale model can provide critically needed high resolution corrections to the larger scale model and the larger scale model can provide the time dependent boundary conditions back to the smaller scale model so that consistent information is used by both models (Chang et al, 1993; Chang et al., 1994). Since this technique is not yet published some technical details are provided here.

Consider the numerical solution of a set of pollutant transport equations over a given geographical domain. First, a base uniform grid system is selected and over this grid system a suitable finite difference approximation is applied to this set of equations. This solution, solved over a predetermined uniform grid, is called the base model or

the coarse grid model. Over a subdomain of this domain a finer grid system is superposed using a three-to-one reduction in grid size. Because in most air quality models chemical species mixing ratios are specified at grid center, an odd number scaling ratio insures that every third grid column in the nested grids shares the same physical location with one of the base model grid columns. The relevant solution over this nested grid system is the nested model or fine grid model. At any given time step the base model will provide the appropriate boundary conditions for all variables to the nested model. Each model will be solved independently but simultaneously. Because of Courant condition, to maintain relative accuracy the nested model must take several small time steps to reach the same time instant as one step of the base model. At this juncture, a variant of the Richardson extrapolation algorithm is applied at all grid points that have solutions in both grid systems to provide coupling between results from these two models.

Without loss of generality, a solution of a one-dimensional advection equation for a scalar ϕ is,

$$\frac{\partial \phi}{\partial t} + u \frac{\partial \phi}{\partial x} = 0 \quad (5)$$

where u is the velocity. For any given finite difference solution scheme of this transport equation included in an air quality model, adopt the convention that lower case variables denote those values associated with the nested model and upper case variables denote the corresponding values associated with the base model. For example, the scalar ϕ_j^{n+1} represents the value at grid point j after $n+1$ time steps in the nested domain. For a one-step (in time) advection scheme that is p th-order accurate in space, with constant grid spacing Δx and time increment Δt over the nested domain, the analytical solution $(\phi_j^{n+1})^*$ is approximated as

$$(\phi_j^{n+1})^* = \phi_j^{n+1} + C \left[\Delta t \cdot (\Delta x)^p \cdot (u\phi)_j^{(p+1)}(\xi) \right], \quad (6)$$

where $(u\phi)_j^{(p+1)}$ is the values of $p+1$ th-order spatial derivative of $(u\phi)$ at grid point j after $n+1$ time steps, and C is some constant. ξ represents some mean value in the neighborhood of $((n+1)\Delta t, j\Delta x)$ and exists by virtue of the Mean Value Theorem. Similarly, in the coarse grid domain the coarse grid scalar Φ at grid J after $N+1$ time steps is set as (Φ_J^{N+1}) . Grid spacing and time increment are denoted ΔX and ΔT respectively. Then the analytical value $(\Phi_J^{N+1})^*$ is

$$(\Phi_J^{N+1})^* = \Phi_J^{N+1} + C \left[\Delta T \cdot (\Delta X)^p \cdot (U\Phi)_J^{(p+1)}(\eta) \right], \quad (7)$$

where U represents wind speed on the coarse grid and η is some mean value in the neighborhood of $((N+1)\Delta T, J\Delta X)$. If grid j and grid J are at the same physical location in the modeling domain and $(n+1)\Delta t$, and $(N+1)\Delta T$, are at the same time, then the analytical solutions should be equal,

$$(\phi_j^{n+1})^* = (\Phi_J^{N+1})^* . \quad (8)$$

If we further assume the solutions to be sufficiently smooth over the domain of simulation then the unquantified part of the error terms of (6) and (7) are approximately equal to each other,

$$(u\phi)_j^{(p+1)}(\xi) \equiv (U\Phi)_J^{(p+1)}(\eta). \quad (9)$$

Therefore, after rearrangement, truncation errors of (6) and (7) can be estimated. Equations (6) and (7) can be rewritten as

$$(\phi_j^{n+1})^* = \phi_j^{n+1} - \frac{\Delta t \cdot (\Delta x)^p}{[\Delta T \cdot (\Delta X)^p - \Delta t \cdot (\Delta x)^p]} (\Phi_J^{N+1} - \phi_j^{n+1}), \quad (10)$$

and

$$(\Phi_J^{N+1})^* = \Phi_J^{N+1} - \frac{\Delta T \cdot (\Delta X)^p}{[\Delta T \cdot (\Delta X)^p - \Delta t \cdot (\Delta x)^p]} (\Phi_J^{N+1} - \phi_j^{n+1}). \quad (11)$$

Equations (10) and (11) provide corrections to the solutions on the corresponding fine grid and coarse grid points, respectively. The corrected solutions are identical at those common points up to the next order of the truncation error for the given difference scheme. It is clear that at all points of collocation, solutions on both grid systems are identical (except for grid average scaling) and information from the nested model has been integrated into the base model. Furthermore, at those points of the corrected solutions are usually more accurate than either solution individually.

This two-way nesting scheme has been successfully implemented in SAQM (Chang et al, 1994). For a model validation case of August 1990 over central California the two-way nested modeling system was able to bring the coarse grid solution of oxidant concentration to near the level of the fine grid solution. In fact very early in this five-day episode the coarse grid was not able to predict that a certain rural region was exceeding the US ozone standard. But with the help of the nested model the exceedances were simulated in the fine grid model. This then fed back to the coarse grid model for the next several hours. For every subdomain tested with the two-way nesting technique, better resolution was achieved in the corresponding grid location in the coarse grid model. At the FSD site the over prediction of ozone peak on the last day of the simulation was reduced by 50% with the 4x4 km² nested model while all other peaks remained the same. This change was traced to the fluctuations in wind fields around this site in 12x12 km² SAQM and the nested model with much smaller grid size. The important finding is that when prediction improvement is made through nesting, the previous good results remains the same. This adds significantly to confidence in the model as a prognostic tool.

Surface layer submodel

In the application of SAQM and similarly with other Eulerian air quality models it was found that comparison of calculated trace gas concentrations in the lowest level of the model with measured ground level concentrations often disagree. This is particularly true for VOCs and NOx. For primary pollutants SAQM usually underestimates the concentrations in source regions. In the same regions, SAQM also frequently overestimates ozone minima at night.

This problem can be attributed to the fixed 60m thickness of the lowest level of SAQM. At night, if the boundary layer is very stable, in 5-8 hours pollutants emitted from the surface may not mix higher than 20m or so. But SAQM with a 60m-thick first level will mix all sources uniformly in this layer. Therefore independently of all other factors one can expect an underestimate of source gases at night by factors of 3 or more. At the same time, surface ozone in the source region is removed by interaction with NO and through dry deposition. If the first level is 60m thick then we have a 60m volume of ozone to remove which is much more than the 20m or so occurring in real life. Further the underestimated NO concentration will not be so efficient in converting ozone to NO₂ which then adds to the underestimation.

An obvious solution to this problem is to add more vertical levels near the surface keeping a thin first level, e. g. 10m or 20m thick. This was tested with SAQM with interpolated wind fields. The surprising finding was that SAQM execution speed degraded significantly. The comparison with observation did indeed improve but not as expected. The increase in execution time was due to the need for a much smaller time step in solving the vertical diffusion equation during daytime period. If the time-step was increased, the results were either poor or the solution could actually become unstable. The sharp change in layer thickness between the first and second layer caused significant loss in numerical accuracy unless much smaller time steps were used. The resulting increase in computation cost by factors of two or more was not acceptable in application.

The following surface layer submodel (sls) was developed to resolve the first level of the original SAQM grid system with a number of new levels, e. g. level-a, level-b and level-c (see figure on next page). The diffusion coefficient K_2 is given by MM5 or SAQM as before. The diffusion coefficients K_a , K_b , K_c are estimated at the resolved levels using MM5 meteorological conditions. At the present, levels a, b and c have thicknesses 10 m, 20 m and 30 m, respectively. K_1 is an average of K_a , K_b and K_c . Wind fields for each of these levels are also interpolated and the mass-consistency error correction technique of SAQM is used in the sls submodel in computing advections.

In SAQM with sls, all surface emissions will go into level a. Those point sources that originally went into level 1 of SAQM will be reprocessed to go into levels b or c of the sls submodel as appropriate. Dry deposition applies only to level a in the sls submodel.

Original SAQM
Grid

level 2, K2	Surface-layer Submodel Grid	
	Kc	level c
level 1, K1	Kb	level b
	Ka	level a

At each time step the exchange between levels a, b and c is computed directly using all the MM5 meteorological variables as if the original equations were solved. Level 2 is the upper boundary condition for the sls submodel. The numerical solution of the sls submodel is efficient because it has only three levels. Explicit methods with small time steps can be used.

In the sls submodel all trace chemicals can also undergo horizontal transport taking into account the effect of resolved wind fields from MM5 and the impact of sources and dry depositions into thin surface layers. For each variable its values are averaged from levels a, b and c and the corresponding variable in level 1 is set at this value. Now the original vertical exchange is solved with the original scheme using levels 1, 2, 3,... At this stage a correction is imposed in that changes in level 1 concentrations from this step are then proportionally distributed into levels a, b and c in the sls submodel.

Using the sls, it is clear that while maintaining the same daytime accuracy in ozone predictions, night-time ozone levels improved dramatically. What is even more significant is that simulated NOx time series at these sites agreed much better with observations.

There remained overshoot in NOx time series associated with singularities in emission modeling which will be corrected in the future. The ability to predict extremely high NOx levels such as over 100 ppb as is actually observed is a significant advance in urban ozone modeling.

As of this writing, SAQM is in use by about half-a-dozen groups outside the staff at its home repository at the California Air Resources Board. As mentioned above, it has been modularized at the North Carolina Supercomputing Center. The modules include alternative treatments of advection, diffusion, chemistry, and numerical

solution algorithms and are user selectable. Simulations using this modularized system in a SAQM-emulation configuration agree almost exactly with those performed with the original version.

References

- Bott, A., A positive definite advection scheme obtained by nonlinear renormalization of the advective fluxes, *Mon. Wea. Rev.*, 117, 1006-1015, 1989a.
- Bott, A., Reply, *Mon. Wea. Rev.*, 117, 2633-2636, 1989b.
- Carter, W.P. "A detailed mechanism for the gas-phase atmospheric reactions of organic compounds." *Atmos. Environ.* **24A**, 481-518, 1990.
- Chang, J. S., R. A. Brost, I. S. A. Isaksen, S. Madronich, P. Middleton, W.R. Stockwell, and C. J. Walcek, A three-dimensional Eulerian acid deposition model: Physical concepts and formulation, *J. Geophys. Res.*, 92, 14,681-14700, 1987.
- Chang, J. S., F. S. Binkowski, N. L. Seaman, J. N. McHenry, P. J. Samson, W.R. Stockwell, C. J. Walcek, S. Madronich, P. B. Middleton, J. E. Pleim, and H. H. Lansford, The Regional Acid Deposition Model and Engineering Model, NAPAP Report 4, 1990.
- Chang, K.-H., F. T. Jeng, and Julius S. Chang, A two-way nesting technique for air quality modeling, 1993 (preprint).
- Chang, Julius S., Shengxin Jin, and Ken-Hui Chang, Two-way and one-way nested SARMAP air quality models, 1994 (preprint).
- Demerjian, K. L., K. L. Schere, and J. T. Peterson, Theoretical estimates of actinic spherically integrated flux and photolytic rate constants of atmospheric species in the lower troposphere. *Adv. Envir. Sci. Technol.*, 10, 369-459, 1980.
- Dodge, M. C., A Comparison of Three Photochemical Oxidant Mechanisms, *J. Geophys. Res.*, v. 94, D4 5121-5136, 1989.
- Grell, G. A., J. Dudhia, and D. R. Stauffer. A Description of the Fifth-Generation Penn State/NCAR Mesoscale Model (MM5). NCAR Technical Note, NCAR/TN-398+STR, 1994.
- Haagenson, P. L., J. Dudhia, D. R. Stauffer, and G. A. Grell. The Penn State/NCAR Mesoscale Model (MM5) Source Code Documentation. NCAR Technical Note, NCAR/TN-392+STR, 1994.
- Jin, S. and J. S. Chang. A Three-dimensional Nonhydrostatic Regional Scale SARMAP Air Quality Model (SAQM), 1994, preprint submitted to JGR.
- Kitada, T., Effect of non-zero divergence wind fields on atmospheric transport calculations, *Atmos. Environ.*, 21, 785-788, 1987.
- Lurmann, F. W., A. C. Lloyd, and R. Atkinson, A chemical mechanism for use in long-range transport/acid deposition computer modeling, *J. Geophys. Res.*, v. 91, 10905-10936, 1986.

Appendix C: Air Pollution Models/Systems

Massman, W.J., et al. "An evaluation of the Regional Acid Deposition Model surface module for ozone uptake at three sites in the San Joaquin Valley of California." *J. Geophys. Res.*, 99, 8281-8294, 1994.

Pleim, J. E., and J. S. Chang, A non-local closure model for vertical mixing in the convective boundary layer, *Atmos. Environ.*, 26A, 965-981, 1992.

Prather, M.J., Numerical advection by conservation of second-order moments, *J. of Geophys. Res.*, 91, 6671-6681, 1986.

Sasaki, Y., An objective analysis based on the variational method, *J. Meteor. Soc. Japan*, 36, 77-88, 1958.

Smolarkiewicz, P.K., A simple positive definite advection scheme with small implicit diffusion, *Mon. Wea. Rev.*, 111, 479-486, 1983.

Smolarkiewicz, P. K., and J. A. Pudykiewicz, A class of Semi-Lagrangian Approximations for Fluids, *J. Atmos. Sci.*, 49, 2082-2096, 1992.

Stockwell, William R., Paulette Middleton, Julius S. Chang, and Xiaoyan Tang, The second generation regional acid deposition model chemical mechanism for regional air quality modeling, *J. Geophys. Res.*, v. 95, 16343-16367, 1990.

Walcek, C.J., and G.R. Taylor, A theoretical method for computing vertical distributions of acidity and sulfate production within cumulus clouds, *J. Atmos.Sci.*, 43, 339-355, 1986.

Whitten, G. Z., H. Hogo, and J. P. Killus, The carbon-bond mechanism: A condensed kinetic mechanism for photochemical smog, *Environ. Sci. Technol.*, v. 18, 280-287, 1980.

UAM

Systems Applications International

The Urban Airshed Model (UAM) developed and maintained by Systems Applications International (SAI) is the most widely used photochemical air quality model in the world today. Its roots extend back to the earliest attempts at photochemical air quality modeling in the early-1970s pioneered by SAI. Since that date, the model has nearly undergone a continuous process of application, comprehensive performance evaluation, update, extension, and improvement. The UAM has been applied to and evaluated in over 34 cities, 26 in the United States, and 8 abroad including Athens, Turin, Taipei, Kaohsiung, Tokyo, Mexico City, and Melbourne. It has been successfully applied and evaluated in areas with meteorological conditions that range from simple to complex, coastal and inland locations, mild-to-severe ozone problems, emissions configurations ranging from low-to-high spatial variability, and databases that range from sparse to rich.

There are two main versions of the UAM in use today: the standard EPA version of the UAM (UAM-IV), which is the EPA recommended model for performing regulatory analysis and making policy decisions with regard to ozone abatement, and the new variable-grid version of the UAM (UAM-V), which incorporates many scientific improvements over the UAM-IV. The UAM and UAM-V are both supported by comprehensive modeling systems for developing the inputs required by 3D photochemical models. These modeling systems are described next, followed by a more detailed description of the nested-grid UAM-V.

Overview of the UAM-IV Modeling System

The current UAM-IV modeling systems was released by the EPA in 1990, and is fully documented in the UAM user's guide. The UAM-IV modeling system includes the following components:

- o The UAM-IV 3D photochemical grid model for estimating ozone and precursor concentrations (Morris and Myers, 1990);
- o The Diagnostic Wind Model (DWM) for generating physically realistic 3D wind fields (Douglas, Kessler, and Carr, 1990);
- o Version 2.0 of the Emissions Processing System (EPS 2.0) which generates hourly, gridded, speciated emissions data required by a photochemical grid model (EPA, 1992);

- o The ROM/UAM Interface System for generating UAM-IV initial concentrations and boundary condition inputs from output from EPA's Regional Oxidant Model (ROM) (Tang *et al.*, 1990); and,
- o Assorted additional pre-processing programs for generating additional UAM-IV inputs (Morris *et al.*, 1990).

A UAM Postprocessing System (UPS) has also been developed for displaying the UAM inputs and outputs. In addition, user-friendly GUIs have been developed for the UAM-IV, such as the UAMGUIDES, which provide easy access to the operation and visualization of the inputs and outputs of the UAM-IV.

Overview of the UAM-V Modeling System

There are currently two modeling systems centered around the UAM-V: the Lake Michigan Ozone Study (LMOS) Photochemical Modeling System, and the UAM-V Modeling System. Both these modeling systems use prognostic meteorological models with 4D data assimilation (4DDA) for developing dynamically consistent 3D meteorological inputs for the UAM-V: the LMOS system uses the CALRAMS, and the UAM-V system uses the SAIMM prognostic meteorological models. The EMS-95 and EPS 2.0 emissions modeling systems are used, respectively, with the LMOS and UAM-V modeling systems. These systems are currently being used in the Lake Michigan, Gulf of Mexico, Atlanta, Northeast, and other regions.

Description of the UAM-V

The most current operational version of the UAM, UAM-V, contains two-way grid nesting allowing regional-scale precursor transport and several imbedded urban areas to be treated in a single modeling domain. In addition, UAM-V allows variable vertical layer number and spacing, specification of 3D meteorological variables, and explicit treatment of subgrid-scale photochemical plumes (i.e., plume-in-grid). The UAM-V software has been completely rewritten to be modular in form, and includes updated deposition, plume rise, solar flux, and chemical kinetics modules.

Conceptual Overview of the Model

Version V of the Urban Airshed Model (UAM-V) is a 3D photochemical grid model designed to calculate the concentrations of both inert and chemically reactive pollutants by simulating the physical and chemical processes in the atmosphere that affect pollutant concentrations. The basis for the UAM-V is the atmospheric diffusion or species continuity equation. This equation represents a mass balance in which all of the relevant emissions, transport, diffusion, chemical reactions, and removal processes are expressed in mathematical terms. The model is usually applied to an 48- to 120-hour period during which adverse meteorological conditions result in elevated pollutant concentrations of the chemical species of interest.

The major factors that affect photochemical air quality include:

- o The spatial and temporal distribution of emissions of NO_x and volatile organic compound (VOC) (both anthropogenic and biogenic);
- o The composition of the emitted VOC and NO_x;
- o The spatial and temporal variations in the wind fields;
- o The dynamics of the boundary layer, including stability and the level of mixing;
- o The chemical reactions involving VOC, NO_x, and other important species;
- o The diurnal variations of solar insolation and temperature;
- o The loss of ozone and ozone precursors by dry and wet deposition; and,
- o The ambient background of VOC, NO_x, and other species in, immediately upwind, and above the region of study.

The UAM-V simulates these processes when it is used to calculate ozone concentrations. It can also be used to simulate carbon monoxide concentrations in an urban area, a simulation that involves no chemical reactions. In the model, the species continuity equation is solved using the method of fractional steps in which the atmospheric diffusion equation is solved separately in the following order: emissions are injected; horizontal advection/diffusion is solved; vertical advection/diffusion and deposition is solved; and chemical transformations are performed for reactive pollutants. The UAM-V performs this four-step solution procedure during each time step. The maximum time step is a function of the grid size and the maximum wind velocity and diffusion coefficient. Typical time steps for coarse (10-20 km) grid spacing is 10-15 minutes, whereas, time steps for fine grid spacing (1-2 km) will be on the order of minutes.

Because the UAM-V accounts for spatial and temporal variations, as well as differences in the reactivity (speciation) of emissions, it is ideally suited for evaluating the effects of emission control scenarios on urban air quality. This is accomplished by first replicating a historical ozone episode to establish a base case simulation. Model inputs are prepared from observed meteorological, emission, and air quality data for a particular day or days using prognostic meteorological modeling and/or diagnostic and interpolative modeling techniques. The model is then applied with these inputs and the results are evaluated to determine its performance. Once the model results have been evaluated and determined to perform within prescribed levels, the same meteorological

inputs and a projected emission inventory can be used to simulate possible future emission scenarios. That is, the model will calculate hourly ozone patterns likely to occur under the same meteorological conditions as the base case.

The new UAM-V has the following additional features over the UAM-IV:

1. Structured modular computer code: The UAM-V computer code has been rewritten to be more modular for ease of inclusion of new modules and to take advantage of modern computers.
2. Vertical grid structure: The vertical layer structure in the UAM-V can be arbitrarily defined by the user and is no longer defined from the diffusion break (mixing height). This allows for higher-resolution vertical layers near the surface and better matching with output from prognostic meteorological models, which usually use a terrain-following coordinate system.
3. 3D inputs: Several meteorological variables that were considered spatially constant in the UAM CB-IV (i.e., in the METSCALARS input file) now vary temporally and spatially (e.g., temperature, water vapor, pressure, and photolysis rates). Furthermore, the horizontal diffusivities and vertical turbulent exchange coefficients are now required as input, usually calculated from a prognostic meteorological model.
4. Variable-grid resolution for chemical kinetic calculations: A chemical aggregation scheme has been implemented in the UAM-V; the chemistry calculations can be performed on a variable grid while the advection/diffusion and emissions injections are performed on a fixed grid.
5. Two-way nested grid: Finer grids can be imbedded in coarser grids for more detailed representation of advection/diffusion, chemistry, and emissions. Several levels of nesting can be accommodated.
6. Update of the CB-IV chemical mechanism: The Carbon Bond IV chemical mechanism has been updated. The XO_2-RO_2 reaction has been added along with new temperature effects for PAN reactions. In addition, aqueous-phase chemistry has been added as an option.
7. New dry deposition algorithm: The dry deposition algorithm formulated by Wesely (1989) has been implemented in the UAM-V. This algorithm is similar to that used by the Regional Acid Deposition Model (RADM).
8. Advanced prognostic meteorological model: A new advanced prognostic meteorological model was developed to support the application of the UAM-V to regional or urban domains (Douglas, Kessler, and Lester, 1991). The

prognostic model incorporates the latest 4D data assimilation, solution “nudging”, and objective combination methods to obtain physically realistic 3D meteorological fields for input into the UAM-V.

9. **True Mass Balance:** Concentrations are advected and diffused in the model using units of mass per unit volume rather than ppm. When a given amount of ppm of a pollutant is transported from one grid cell to another that has a different temperature or pressure there is a different amount of mass within the grid cell for the same ppm (ideal gas law). By expressing concentrations as mass per unit volume then true mass balance is maintained in the advection and diffusion.
10. **Plume-in-Grid:** Emissions from point sources can be treated by a subgrid-scale Lagrangian photochemical plume model. Pollutant mass is released from the subgrid-scale model to the grid model when the plume size is commensurate with a grid cell size.

Technical Formulation

The Urban Airshed Model (UAM) is a 3D grid (Eulerian) model designed to calculate the concentrations of both inert and chemically reactive pollutants by simulating the physical and chemical processes in the atmosphere that affect pollutant concentrations. The basis for the UAM is the atmospheric diffusion equation (also called the species continuity or advection/diffusion equation). This equation represents a mass balance in which all of the relevant emissions, transport, diffusion, chemical reactions, and removal processes are expressed in mathematical terms as follows:

$$\begin{array}{ccccccc}
 \frac{\partial c_i}{\partial t} & + & \frac{\partial(uc_i)}{\partial x} & + & \frac{\partial(vc_i)}{\partial y} & + & \frac{\partial(wc_i)}{\partial z} & = & \frac{\partial}{\partial x} \left(K_x \frac{\partial c_i}{\partial x} \right) & + & \frac{\partial}{\partial y} \left(K_y \frac{\partial c_i}{\partial y} \right) & + & \left(K_v \frac{\partial c_i}{\partial z} \right) \\
 \text{Time} & & \text{Advection} & & & & & & \text{Turbulent Diffusion} & & & & & \\
 \text{Dependence} & & & & & & & & & & & & & \\
 & & & & & & & & & + & R_i & + & S_i & + & D_i & + & W_i \\
 & & & & & & & & & \text{Chemical} & \text{Emission} & & \text{Dry} & & \text{Wet} \\
 & & & & & & & & & \text{Reaction} & & & \text{Deposition} & & \text{Deposition}
 \end{array}$$

where c_i represents the pollutant concentration and is a function of space (x, y, z) and time (t). The other terms in this equation are:

- u, v, w = horizontal and vertical wind speed components
- K_H, K_V = horizontal and vertical turbulent diffusion coefficients
- R_i = net rate of production of pollutant i by chemical reactions
- S_i = emission rate of pollutant i

D_i = net rate of removal of pollutant i by surface uptake processes

W_i = net rate of removal of pollutant i by wet deposition processes

Overview of Model Concepts

The UAM-V employs finite differencing numerical techniques for the solution of the advection/diffusion equation. The region to be simulated is divided up into several 3D grids covering the region of interest. The UAM-V is set up with a base coarse grid covering the entire domain which is augmented by finer nested-grids for regions in which more refined analysis is deemed necessary. The UAM-V accepts nesting in the horizontal and vertical, and allows for many levels of nesting if desired.

The vertical layer structure of the UAM-V can be defined arbitrarily by the user. Usually, the vertical layers of the UAM-V are defined to match the vertical layer structure of a prognostic meteorological model which are frequently used to define the UAM-V meteorological inputs. Note that this makes the UAM-V much more flexible than the UAM, which defined its vertical layers from the diffusion break (mixing height) input. The UAM-CB-IV grid nesting in the vertical allows for the use of high resolution in regions where it is needed (e.g., urban centers, coastal regions), and coarser resolution elsewhere.

References

- EPA (1986). "Guidelines on Air Quality Model (Revised)," U.S. Environmental Protection Agency, RTP, NC (EPA-450/2-78-027R).
- EPA (1991). "Guideline for Regulatory Application of the Urban Airshed Model," U.S. Environmental Protection Agency, Research Triangle Park, NC (EPA-450/4-91-013).
- Morris, R.E. and T.C. Myers (1990). "User's Guide for the Urban Airshed Model--Volume I: User's Manual for UAM(CB-IV)," U.S. Environmental Protection Agency, RTP, NC (EPA-450/4-90-007a).
- Morris, R.E., T.C. Myers, S.G. Douglas, and M.C. Causley (1990). "User's Guide for the Urban Airshed Model--Volume II: User's Manual for UAM Modeling System," U.S. Environmental Protection Agency, RTP, NC (EPA-450/4-90-007b).
- Morris, R.E., T.C. Myers, S.G. Douglas, M.A. Yocke, and V. Mirabella (1991). Development of a Nested-Grid Urban Airshed Model and Application to Southern California. 84th Annual Air and Waste Management Association Meeting, Vancouver, BC, June 6-21 1991.
- SAI (1993). "User's Guide to the Lake Michigan Ozone Study Photochemical Modeling System", three volumes, Systems Applications International, San Rafael, CA (SYSAPP-93/153).

APPENDIX D:

AN EXAMPLE OF FORTRAN CODING STANDARDS – FORTRAN CODING STANDARDS FOR MODELS-3

FORTRAN CODING STANDARD FOR MODELS-3

REUSE

If a library module does the task already, use it. Don't reinvent the wheel!!!

TYPOGRAPHY AND STYLE

ANSI

Code will conform to the FORTRAN ANSI 77 standards with the following exceptions:

- (1) Modules written to conform to FORTRAN ANSI 90;
- (2) Special cases where the use of extensions is necessary to achieve specific required results;
- (3) Extensions permitted by this Models-3 standard.

Compiler option(s) should be enabled to identify non-ANSI usages. Code should compile without non-ANSI messages with the above exceptions.

Source File Extensions

To satisfy the Models-3 model-builder tool use the following FORTRAN source code file name extensions:

- (1) ".F" for other than include files;
- (2) ".EXT" for include files.

SCCS

To satisfy the Models-3 model-builder tool and its use of SCCS include a comment statement at the beginning of each FORTRAN source code file as follows:

```
C %W% %P% %G% %U%
```

Comments

Comment text is preferred in mixed case (with capitalization of proper names, etc., as in normal writing). Program code should be in upper case.

Comments should be used freely, but should be concise and should not restate the obvious.

In-line exclamation-point comments documenting purpose of variables, individual steps in algorithms, and beginning and ending statements of loops or the various parts of extended IF-THEN-ELSE-END IF constructions are to be encouraged.

Revision history should be maintained in a format conforming to that of the program/subroutine/function/INCLUDE templates (Appendices 1 and 2). For the respective version of the module, it should document the nature of each revision, the name of the developer, and the date of revision.

Line Lengths, Margins, and Indentation

Production FORTRAN code must fit between columns 1 and 80 inclusive. Use column 1 for "C" or "*" comment-line indicators (be consistent with C or *). Use columns 1 through 5 for any statement labels. (You may opt to limit statement labels to columns 2 through 5.) Left

justify statement numbers. Use column 6 for continuation indicator(s) only. Use columns 7 through 72 for FORTRAN statements (comments may extend to column 80). Indent statements for clarity.

The bodies of DO-loops, IF-THEN-ELSE constructions, and loops constructed with GO TOs should be indented using increments consistent for the entirety of a module. For example:

```
      DO 200 IHOURL = 1, 24
        IF (THIS(IHOURL) .EQ. THAT) THEN
          XVAL(IHOURL) = COS(FLOAT(IHOURL))
        ELSE
          XVAL(IHOURL) = SIN(FLOAT(-IHOURL))
        END IF
100    CONTINUE      ! begin loop doing something
        CALL SOMETHING(IHOURL, WHATEVER, FLAG)
        IF (FLAG) GO TO 200
        XVAL(IHOURL) = XVAL(IHOURL) + WHATEVER
        GO TO 100    ! end loop doing something
200    CONTINUE      ! end loop on IHOURL
```

Line Numbers

FORTRAN statement labels should be in increasing order with gaps in numbering so that statement labels of future code revisions can maintain increasing number order.

White Space

<TAB>s must be removed from source code before production acceptance.

Commas in lists (e.g., declaration lists, READ/WRITE lists, argument lists, format lists) should be followed by one or more blanks.

Use blanks to pad out statements to show structural similarities and to provide better readability. For example:

```
X (IROW) = A + SIN(B * C) / D
YAB(IROW) = EE + B**2 - C**2
```

is preferable to:

```
X(IROW)=A+SIN(B*C)/D
YAB(IROW)=EE+C**2-D**2
```

Blank lines or blank comment statements should be used to set off program structures as appropriate (so that logically related parts are grouped together by the reader's eye).

NOTE: the current IBM compiler doesn't accept blank lines. However, there exist EVE procedures for making the IBM conversion automatically. Under UNIX, awk or sed can do this easily, also.

Continuations

"&" is the preferred continuation character.

Continue executable statements only after delimiters (e.g., after commas in an argument list, or before or after arithmetic operators in an arithmetical expression).

Text in a continuation should be indented beyond the current indentation level, or as appropriate to show structural similarities.

Floating-Point Constants

The use of zeros where needed on either side of a decimal point to enhance readability of floating-point constants is preferred. For example, use "1.0" and "0.007" instead of "1." and ".007".

Divides

Prevent divide-by-zero situations by testing divisors for zero prior to their intended use in divides. If zero, take an alternate course of action rather than performing the divide. (Better to program for it than to be caught by it.)

Error Conditions

It shall be possible from the text of any error messages logged by a program to uniquely identify the location in the source code at which the error occurred and the nature of the error.

DECLARATIONS AND TYPING

Variable Names

Variable names should carry a meaning relevant to the context of the usage. For example, when using a DO loop to traverse a grid, "IROW" and "JCOL" are more meaningful names than "I" and "J".

Portability may require that variable names have at most six characters.

Data Types

Inclusion of an IMPLICIT NONE statement in each FORTRAN routine is preferred (where the compiler supports it).

Declaring data types as REAL, DOUBLE PRECISION, and INTEGER is preferred over REAL*4, REAL*8, and INTEGER*4 (for code portability purposes).

Conversion of values between data types should be explicit: use FLOAT, IFIX, INT, NINT, SNGL, and DBLE as appropriate. For example, if X is a real variable and I is an integer, "X = FLOAT (I)" is much easier to maintain than "X = I".

Testing REAL or COMPLEX variables for equality risks causing numerical problems. For two REAL numbers it is better to test that the absolute value of the difference between them is less than some appropriately small number, or better still for the general case, test that the absolute value of the difference is some small percentage of the absolute value of the larger of the two numbers. For example:

```
IF (ABS(A - B) / MAX(ABS(A), ABS(B)) .LT. 1.0E5) THEN ...
```

For two complex numbers test both the real portions and the complex portions separately in this manner.

Comparisons (in IF statements, for instance) should be between like types. Some FORTRAN compilers have been known to treat "IF (X .EQ. 1)..." as a comparison of bit patterns between REAL variable X and INTEGER constant 1.

Variables should be initialized before reference (on many

systems (e.g., CRAY), variables are NOT initialized to zero by the loader).

SAVE Statements and State Variables

For portability, variables within a function or subroutine which must retain their values from one call to the next should be listed in SAVE statements.

NOTE: the current VAX and IBM FORTRAN compilers do this anyway; the CRAY FORTRAN compiler does not.

("State variables" are variables local to a subroutine, used to govern the behavior of that subroutine on the basis of its past history (e.g., to indicate whether a routine which reads lines from a file is currently reading a header line, a data line, or whatever). "Text pointers" is another term which is used, especially when dealing with files.)

State variables used to control subroutines should be fully documented by comments located where the state variable is declared in the source code. These comments should indicate:

- (1) the set of possible values;
- (2) the meaning of each value; and
- (3) the action corresponding to that value.

Common Blocks and INCLUDE Files

Do not use unnamed common blocks.

Common blocks should normally be placed in INCLUDE files. One common block or related set of blocks per file is preferred.

Dependencies in INCLUDE files (e.g., for dimensioning parameters) should be documented in header comments for the dependent INCLUDE file.

Use includes in one of two ways:

- (1) For INCLUDEs that will always be used in a FORTRAN source use:

INCLUDE 'file name'

- (2) For an INCLUDE where a selection is made from a number of possibilities (e.g., different sets of model configuration parameters) use:

INCLUDE file name (Note: no quotes)

Using the second form you can then specify in a model configuration file which include file to use.

Subroutines and Functions

Where applicable, comment text should be included that outlines the purpose of each major variable, the logic of each coded algorithm, literature reference identifying source of each published equation and of equation parameters

The use of system-dependent code must be clearly indicated in the header comment.

Where there exist subroutine preconditions, (i.e., operations which MUST be done within the program before a call to the routine can be expected to work), they should be

clearly stated within header comments.

Within the calling module, subroutine calls should have comments indicating the purposes for which major subroutines are being invoked.

Return values for functions are NOT "output arguments" and SHOULD NOT be documented as though they were. Nor are they elements of the "argument list": they should instead have a separate heading in the declaration comments.

External functions are NOT "local variables" and should not be documented as though they were--they are neither local nor variable. (In fact, FORTRAN functions are global constants.)

Whenever possible, use the generic name for intrinsic functions, rather than the type-specific name. For example, use MAX() instead of AMAX1(). (It is safer to let the compiler determine the correct type for the function than for the user to possibly specify an incorrect type.) The only time that the type-specific function should be needed is if the function itself is being used as a subroutine argument.

The Models-3 model manager provides a couple of features that make it easier to select from optional subroutines such as for different nested grid models. See the model manager documentation for code requirements relating to these features.

Locality

All specification information for a variable should be located together textually in the program source code; this includes the placement together of usage comment, declaration, dimensioning, and initialization. For example:

```
...
C  thrimble constants THRM used by the Widgert Algorithm:
      REAL THRM( 3 ) / 1.0, 2.0, 3.0 /
...
```

is greatly to be preferred over widely separating the usage comment, declaration, dimension, and data statements.

Declaration Grouping

Grouping of declarations by purpose (for arguments/parameters/external functions/local variables) is preferred. For example:

C arguments

```
      INTEGER      ICOL      ! column for calculation
      REAL          XVAL      ! x,y values for
      REAL          YVAL      ! thrimble calculation
```

C parameter

```
      REAL          GG        ! gravitational constant
      PARAMETER     ( GG = 9.80655 )
```

C external functions

```
      REAL          XATAN2    ! error-trapped arctan(y/x)
      INTEGER       GETNUM    ! prompt for integer
      INTEGER       IOCL      ! interpret VMS error status
      EXTERNAL      XATAN2, GETNUM, IOCL
```

C local variables

```
INTEGER      I, J, K ! loop counters
REAL         X, Y   ! grid coordinates
...
```

Parameters

Physical constants, conversion constants, etc., should be defined using PARAMETER statements, rather than put into variables initialized, e.g., by DATA statements.

When the same parameter values are used by more than one subroutine, they should be placed in an INCLUDE file. Where possible, values from a standard INCLUDE file should be used.

Where possible, derived parameter values should be defined in terms of the primitive parameter values. For example:

```
PARAMETER (IROW = 34, JCOL = 25, IZ = 23)
PARAMETER (ISIZE = IROW * JCOL * IZ)
```

CONTROL STATEMENTS

Arithmetic IF, Alternate RETURN, PAUSE, and Assigned GOTO statements are to be avoided. They will probably be completely forbidden in the next release of the ANSI FORTRAN standard.

CONTINUE statements should be used as the targets of GO TO statements.

Computed GO TO statements should have comments documenting the possible values and corresponding meanings of the control variable.

Terminate DO-loops with (numbered) CONTINUE statements. There should be exactly one closing CONTINUE for each DO loop (i.e., rather than using a single label to terminate multiple nested loops). This is REQUIRED by various parallel FORTRAN compilers and suggested strongly in the upcoming ANSI FORTRAN draft standard.

Use type INTEGER variables (rather than type REAL) for FORTRAN DO-loop counters.

The closing CONTINUE for a DO loop must not be the destination of a GO TO from outside the body of the loop. (From INSIDE the body, a GO TO the end of the loop is OK.) Likewise, the END IF closing an IF-THEN-ELSE construction must not be the destination of a GO TO statement. This standard is strongly suggested by the upcoming ANSI FORTRAN draft standard.

Where feasible, DO loops are preferred over loops constructed with IFs and GO TOs. Likewise, IF-THEN...ELSE IF-THEN...ELSE...END IF constructions are preferred over extended sequences of IFs and GO TOs.

Loops and block constructions (IF-THEN...ELSE IF...ELSE...END IF, DO loops, etc.) with either deep nesting or long blocks of code in them should have documentation (by in-line exclamation point ("!...") comments) for block beginning, alternatives (ELSE blocks), and ending, so that their beginnings, alternatives, and endings can be matched up easily.

Loop beginning and loop ending for loops constructed with IFs and GO TOs must be documented by comments.

It is preferred that "ELSE IF", "END IF" and "GO TO" be two

words each, rather than one word each (i.e., instead of "ELSEIF", "ENDIF", or "GOTO").

INPUT AND OUTPUT

File OPEN statements should be used, instead of ASSIGNS to FOR00x.

READs from standard input and WRITEs to standard output (screen or log) should refer to unit *, rather than to units 5,6.

For programs which receive control inputs from stdin, there should be a prompt for each such input.

For programs which may run in batch mode, each control parameter (frequently taken from standard input, although not necessarily) and its description should be echoed to standard output (i.e., the program log).

Input which controls the time period for processing in a program should be requested in the form "starting date, starting time, duration", rather than some version of "starting date, starting time, ending date, ending time."

Interactive programs should check the validity of all user responses and allow the user to correct erroneous inputs. Case-sensitive input should be required only where absolutely necessary. Where possible, the program should provide "smart" default responses.

Interactive programs should test data availability before reading from an input file, rather than allowing the program to crash from, e.g., an end-of-file.

NOTE: This implies putting an ".IOSTAT =" clause in READ statements, together with appropriate error-handling to deal with I/O errors.

Use mixed-case output to a log: the human-factors specialists say that WHILE ALL CAPS GETS THE READER'S ATTENTION, IT IS HARDER TO READ TEXT THAT IS GIVEN IN ALL CAPS.

As appropriate, each step processed by a Models-3 program should be logged. Each file written to, or otherwise altered, during program execution should be logged.

It is preferred that the log file be formatted to fit within 80 columns, so that it may be viewed from a terminal.

Format numbering and placement should follow a scheme consistent throughout the entirety of a module. The following are two alternative suggestions:

- (1) Place format statements at the end of the program/subroutine/function body. Use line numbers for FORMAT statements following the numbering scheme provided in the template in Appendix 1:

- 91xxx: error and warning messages
- 92xxx: informational (log) messages
- 93xxx: file I/O
- 94xxx: internal buffering
- 95xxx: other (miscellaneous)

- (2) Place format statements immediately following their first uses in READ or WRITE statements.

APPENDIX 1: Template for Programs/subroutines/functions

Program/subroutine/function statement

```
C*****
C
C FUNCTION:
C
C PRECONDITIONS REQUIRED:
C
C RETURN VALUE(S):
C
C KEY SUBROUTINES AND FUNCTIONS CALLED:
C
C REVISION HISTORY:
C*****
```

IMPLICIT NONE

C..... INCLUDES:

C..... ARGUMENTS and their descriptions:

C..... PARAMETERS and their descriptions:

C..... EXTERNAL FUNCTIONS and their descriptions:

C..... SAVED LOCAL VARIABLES and their descriptions:
C..... NOTE: the ANSI standard requires use of SAVE statements
C..... for variables which must retain their values from call
C..... to call.

C..... SCRATCH LOCAL VARIABLES and their descriptions:

C*****

C begin body of subroutine

RETURN

C***** FORMAT STATEMENTS *****

C..... Error and warning message formats..... 91xxx

```
91000 FORMAT ( //5X , '*** ERROR ABORT in subroutine dummy ***',
&           /5X , A ,
&           // ) ! generic error message format
```

C..... Informational (LOG) message formats... 92xxx

C..... Formatted file I/O formats..... 93xxx

C..... Internal buffering formats..... 94xxx

C..... Miscellaneous formats..... 95xxx
95000 FORMAT (/5X , A , \$) ! generic prompt format.

END

APPENDIX 2: Template for INCLUDE Files

C.....
C INCLUDE FILE ZZZZ.EXT
C
C CONTAINS:
C
C DEPENDENT UPON:
C
C REVISION HISTORY:
C.....

C..... end ZZZZ.EXT

LANGUAGE-INDEPENDENT CODING STANDARD FOR MODELS-3

REUSE

If a library module does the task already, use it. Don't reinvent the wheel!!!

TYPOGRAPHY AND STYLE

Source File Extensions

To satisfy the Models-3 model-builder tool use appropriate source code file name extensions. For example:

- (1) ".F" for FORTRAN source code files;
- (2) ".EXT" for include files;
- (3) ".c" for C source code files;
- (4) ".h" for C include files;
- (5) ".cc" for C++ source code files;
- (6) ".h" and ".hh" for C++ include files.

SCCS

To satisfy the Models-3 model-builder tool and its use of SCCS include a comment statement at the beginning of each source code file. For example:

- (1) for FORTRAN:

```
C %W% %P% %G% %U%
```

- (2) for C:

```
char <file name>_vers[] = "%W% %P% %G% %U%";
```

Comments

Revision history should be maintained in a format conforming to the language used. It should document the nature of the revision, the name of the developer, and the date of revision.

Comments should be used freely, but should be concise and should not restate the obvious.

Where applicable, comment text should be included that outlines the purpose of each major variable, the logic of each coded algorithm, literature reference identifying source of each published equation and of equation parameters.

The use of system-dependent code must be clearly indicated in the header comment.

Where there exist routine preconditions, (i.e., operations which MUST be done within one routine before a call to another routine can be expected to work), they should be clearly stated within header comments.

Dependencies in INCLUDE files (e.g., for dimensioning parameters) should be documented in header comments for the dependent INCLUDE file.

State variables used to control subroutines should be fully documented by comments located where the state variable is declared in the source code. These comments should indicate:

- (1) the set of possible values;
- (2) the meaning of each value; and
- (3) the action corresponding to that value.

("State variables" are variables local to a routine, used to govern the behavior of that routine on the basis of its past history (e.g., to indicate whether a routine which reads lines from a file is currently reading a header line, a data line, or whatever). "Text pointers" is another term which is used, especially when dealing with files.)

Loops and block constructions (IF-THEN...ELSE IF...ELSE... END IF, DO loops, etc.) with either deep nesting or long blocks of code in them should have documentation comments for block beginning, alternatives (ELSE blocks), and ending, so that their beginnings, alternatives, and endings can be matched up easily.

Line Lengths, Margins, and Indentation

The bodies of LOOPS, IF-THEN-ELSEs, and SWITCH/CASEs should be indented using increments consistent for the entirety of a module.

It is preferred that source code fit between columns 1 and 80, for readability at standard terminals.

White Space

<TAB>s must be removed from source code before production acceptance.

Commas in lists (e.g., declaration lists, READ/WRITE lists, argument lists, format lists) should be followed by one or more blanks.

Use blanks to pad out statements to show structural similarities and to provide better readability. For example:

```
X (IROW) = A + SIN(B * C) / D
YAB(IROW) = EE + B**2 - C**2
```

is preferable to:

```
X(IROW)=A+SIN(B*C)/D
YAB(IROW)=EE+C**2-D**2
```

Blank lines should be used to set off program structures as appropriate (so that logically related parts are grouped together by the reader's eye). For example, it is a good practice to set off indented blocks of code with blank lines.

Floating-Point Constants

The use of zeros where needed on either side of a decimal point to enhance readability of floating-point constants is preferred. For example, use "1.0" and "0.007" instead of ".1." and ".007".

Divides

Prevent divide-by-zero situations by testing divisors for zero prior to their intended use in divides. If zero, take an alternate course of action rather than performing the divide. (Better to program for it than to be caught by it.)

Error Conditions

It shall be possible from the text of any error messages logged by a program to uniquely identify the location in the source code at which the error occurred and the nature of the error.

DECLARATIONS AND TYPING

Variable Names

Variable names should carry a meaning relevant to the context of the usage. For example, when using a loop to traverse a grid, "IROW" and "JCOL" are more meaningful names than "I" and "J".

Data Types

Conversion of values between data types should be explicit.

Testing REAL or COMPLEX variables for equality risks causing numerical problems. For two REAL numbers it is better to test that the absolute value of the difference between them is less than some appropriately small number, or better still for the general case, test that the absolute value of the difference is some small percentage of the absolute value of the larger of the two numbers:

"X = FLOAT (I)" is much easier to maintain than "X = I".

Variables should be initialized before reference: on many systems (e.g., CRAY), variables are NOT initialized to zero by the loader.

Locality

All specification information for a variable should be located together textually in the program source code; this includes the placement together of usage comment, declaration, dimensioning, and initialization. For example:

```
...  
C thrimble constants THRM used by the Widgert Algorithm:  
  REAL THRM( 3 ) / 1.0, 2.0, 3.0 /  
...
```

is greatly to be preferred over widely separating the usage comment, declaration, dimension, and data statements.

INPUT AND OUTPUT

For programs which receive control inputs from stdin, there should be a prompt for each such input.

For programs which may run in batch mode, each control parameter (frequently taken from standard input, although not necessarily) and its description should be echoed to standard output (i.e., the program log).

Input which controls the time period for processing in a program should be requested in the form "starting date, starting time, duration", rather than some version of "starting date, starting time, ending date, ending time."

Interactive programs should check the validity of all user responses and allow the user to correct erroneous inputs.

Case-sensitive input should be required only where absolutely necessary. Where possible, the program should provide "smart" default responses.

Interactive programs should test data availability before reading from an input file, rather than allowing the program to crash from, e.g, an end-of-file.

Use mixed-case output to a log: the human-factors specialists say that WHILE ALL CAPS GETS THE READER'S ATTENTION, IT IS HARDER TO READ TEXT THAT IS GIVEN IN ALL CAPS.

As appropriate, each step processed by a Models-3 program should be logged. Each file written to, or otherwise altered, during program execution should be logged.

It is preferred that the log file be formatted to fit within 80 columns, so that it may be viewed from a terminal.

C CODING STANDARD FOR MODELS-3

REUSE

If a library module does the task already, use it. Don't reinvent the wheel!!!

TYPOGRAPHY AND STYLE

ANSI

Code will conform to ANSI C (except for limited use of extensions necessary to achieve specific desired results).

Source File Extensions

To satisfy the Models-3 model-builder tool use the following C source code file name extensions:

- (1) ".c" (.cc for C++) for other than include files;
- (2) ".h" (.hh for C++) for include files.

SCCS

To satisfy the Models-3 model-builder tool and its use of SCCS include a char statement at the beginning of each C (or C++) source code file as follows:

```
char <file name>_vers[] = "%W% %P% %G% %U%";
```

Comments

Comments should be used freely and should be concise. They should not be used to restate the obvious.

Comments documenting the following (where they add to readability/understanding) are to be encouraged:

- the purpose of variables,
- individual steps in algorithms,
- at the beginning and ending of loops, structure definitions, typedefs, and the various parts of "if" and "switch" statements.

Revision history should be maintained in a format conforming to that of the templates (Appendices 1 and 2). For the respective version of the module, it should document the nature of the revision, the name of the developer, and the date of revision. (Abbreviated version/revision history information should also be provided to SCCS when making a version/revision change to SCCS controlled code.)

Include Files

Common definitions used across programs should be put in 'include' files.

Line Lengths, Margins, and Indentation

It is preferred that source code fit between columns 1 and 80, for readability at standard terminals.

The bodies of loops, if-then-else constructions, and switch statements should be indented using increments consistent

for the entirety of a module (by 4 columns is suggested). (Doing so makes it easier to visualize control structures.) It is further suggested that the bodies of such constructions be enclosed in braces {}, even if they are single statements. (Doing so makes it easier to add to or to change the code later.) For example:

```

for ( icol = 0 , cnt = 0 ; icol < SIZE ; icol++ )
{
    if ( 0 < a[ icol ] )
    {
        b[ icol ] = a[ icol ] ;
    } /* END IF-CLAUSE: a[icol] > 0 */
    else{
        b[ icol ] = -a[ icol ] ;
        list[ cnt++ ] = icol ;
    } /* END ELSE-CLAUSE: not ( a[icol] > 0 ) */
    switch ( cnt )
    {
        case 0:
            printf ( "a[] still nonpositive at %d", icol ) ;
            break ; /* 0 */
        case 1:
            printf ( "a[] turns POSITIVE at %d", icol ) ;
            break ; /* 1 */
        default:
            printf ( "%d positive a[]'s at %d", cnt, icol ) ;
            break ; /* default */
    } /* END SWITCH on cnt */
} /* END FOR-LOOP on icol */

```

White Space

<TAB>s must be removed from source code before production acceptance.

Commas in lists (e.g., declaration lists, argument lists, format lists) should be followed by blanks.

Assignment operators ("=", "+=", etc.) should be set off by blanks, to avoid confusion with obsolete "+=" (UNIX V6 and before) versions of these operators.

Use blanks to pad out statements to show structural similarities and to provide better readability. For example:

```

x [row] = a + sin(b * (float) c) / d;
yab[row] = ee + b / 2 - (float) c;

```

is preferable to:

```

x[row]=a+sin(b*(float)c)/d;
yab[row]=ee+b/2-(float)c;

```

In comparisons for equality with manifest constants (0, 1, etc.), it is more reliable to put the constant on the left as in "if (0 == x[i]) ...". This is safer, since the assignment operator looks so much like the equality operator, and the compiler will flag "if (0 = x[i])..." as an error, but will silently put a zero in x[i] and decide not to do the body of the if statement in the construction "if (x[i] = 0)..."

Blank lines should be used to set off program structures as appropriate (so that logically related parts are grouped together by the reader's eye).

Floating-Point Constants

The use of zeros where needed on either side of a decimal point to enhance readability of floating-point constants is preferred. For example, use "1.0" and "0.007" instead of ".1." and ".007".

Divides

Prevent divide-by-zero situations by testing divisors for zero prior to their intended use in divides. If zero, take an alternate course of action rather than performing the divide. (Better to program for it than to be caught by it.)

Error Conditions

Error conditions related to data processing (e.g. missing or bad data, or failed data resource acquisition) shall be logged using program issued error messages. It shall be possible from the text of any such error message to uniquely identify the location in the source code at which the error occurred, the nature of the error, and (if appropriate) a suggested way to recover. (The user generally has the ability to recover from these types of error conditions.)

Error conditions resulting from program bugs (e.g., a call to a missing function) should be logged identifying the error location. The program should then be halted immediately. (Use of the 'assert ()' macro is appropriate in these situations where the user has no control over resolving the condition.)

DECLARATIONS AND TYPING

Identifiers

Identifiers should generally be spelled out using an underscore between words. However, abbreviations (e.g., site-id) and acronyms that are readily recognized may be used. (Arbitrary dropping of vowels from identifiers should be avoided.)

In general, use lower case for variable names; start non-variable names (e.g., function names and types) with an upper case letter; use all upper case for acronyms.

Preprocessor definitions should be UPPER CASE, Typedefs should be Capitalized, and other C code should generally be lower case. (Upper case may be used in specialized situations such as advanced variable naming schemes where variable types are indicated in variable names.)

Variable names should carry a meaning relevant to the context of the usage. For example, when using a for-loop to traverse a grid, "row" and "col" are more meaningful names than "i" and "j".

Avoid the following:

Identifiers beginning with the underscore character (`_`)

Identifiers differing only in the case of the characters composing them, at least in circumstances where the identifiers may be visible to a linker (both the VMS and the DOS linkers change identifiers to uppercase before starting to process them).

Use 'static' and 'extern' qualifiers to ensure intended scoping as follows:

- All intentionally private entities (functions, non-routine variables, etc.) should be declared 'static' (to enforce local (file) scope and thus limit global namespace pollution).
- All intentionally public entities should be declared 'extern' in an appropriate header (.h) file for use (via #include).
- The keyword 'extern' should not be used inside a '.c' file.

Data Variables

Where well-defined conversion is not guaranteed, conversion of values between data types should use explicit type-casts. (In general the use of type-casts should be minimized and not used simply to 'silence' legitimate compiler warnings.)

Testing float or double variables for equality risks causing numerical problems. It is better to test that the absolute value of the difference between the two numbers is some small fraction of the absolute value of the larger of the two numbers.

Variables must be initialized before reference.

Functions

Function definition and declaration will use ANSI syntax. All functions called will be declared using function prototypes, in order to ensure that calls contain the correct numbers of arguments, each of the correct types, and return values of the correct types. Where appropriate, sets of declarations may be placed in include-files.

Read-only function arguments should be declared 'const' as in:

```
char *strcat (char *s1, const char *s2);
```

Where applicable, comment text should be included outlining the purpose of each major variable, the logic of each coded algorithm, literature reference identifying source of each published equation and the definitions of equation parameters.

The use of system-dependent code must be clearly indicated in the header comment.

Where there exist preconditions for a function (e.g. operations which MUST be done within the program before a call can be expected to work, or restrictions on the values of input arguments), they should be clearly stated within prologue comments. (Use of the 'assert()' macro may also be apropos to ensure that preconditions are met.)

Within the calling module, calls should have comments indicating the purposes for which major functions are being invoked.

Locality

All specification information for a variable should be located together textually in the program source code; this includes the placement together of usage comment, declaration, dimensioning, and initialization. For

example:

```
...
/* thrimble constants used by the Widgeart */
/* Algorithm: */
static int thrimble[ 3 ] = { 1, 2, 3 }
...
```

is greatly to be preferred over widely separating the usage comment, the declaration, and the initialization.

INPUT AND OUTPUT:

For programs which receive control inputs from stdin, there should be a prompt for each such input.

For programs which may run in batch mode, each control parameter (frequently taken from standard input, although not necessarily) and its description should be echoed to standard output (i.e., the program log).

Interactive programs should check the validity of all user responses and allow the user to correct erroneous inputs. Case-sensitive input should be required only where absolutely necessary. Where possible, the program should provide "smart" default responses, indicated in brackets [LIKE THIS] in the corresponding prompt.

Interactive programs should test data availability before reading from an input file, rather than allowing the program to crash from, e.g, an end-of-file condition.

Use mixed-case output to a log: the human-factors specialists say that WHILE ALL CAPS GETS THE READER'S ATTENTION, IT IS HARDER TO READ TEXT THAT IS GIVEN IN ALL CAPS THAN IT IS TO READ MIXED-CASE TEXT.

As appropriate, each step processed by a Models-3 program should be logged. Each file written to, or otherwise altered, during program execution should be logged.

It is preferred that output (log files) be formatted to fit within 80 columns, so that it may be viewed from any ANSI terminal or printer.

Input which controls the time period for processing in a program should be requested in the form "starting date, starting time, duration", rather than some version of "starting date, starting time, ending date, ending time."

APPENDIX 1: Template for Functions

```
/* (preprocessor DEFINES and INCLUDES here) */  
/* (global declarations here) */
```

```
*****  
*  
* PURPOSE:  
*  
* PRECONDITIONS:  
*  
* KEY CALLS:  
*  
* NOTES: (if any, such as post conditions, machine dependencies, etc.)  
*  
* REVISION HISTORY:  
*  
*****/
```

```
TYPE DUMMY ( /* (argument declarations here) */ )  
  
{ /* begin body of DUMMY() */  
  /* (local declarations here) */  
  /* (function body here) */  
  /* (return exit status here) */  
} /* end body of DUMMY () */
```

APPENDIX 2: Template for INCLUDE files

```
/******  
* INCLUDE FILE: dummy.h  
*  
* PURPOSE:  
*  
* INCLUDE DEPENDENCIES:  
*  
* REVISION HISTORY:  
*  
*****/  
  
#ifndef DUMMY_H /* (inclusion protection) */  
#define DUMMY_H  
  
/* (include-file body goes here) */  
  
#endif /* DUMMY_H */  
  
/****** END dummy.h *****/
```


ABOUT EPRI

Electricity is increasingly recognized as a key to societal progress throughout the world, driving economic prosperity and improving the quality of life. The Electric Power Research Institute delivers the science and technology to make the generation, delivery, and use of electricity affordable, efficient, and environmentally sound.

Created by the nation's electric utilities in 1973, EPRI is one of America's oldest and largest research consortia, with some 700 members and an annual budget of about \$500 million. Linked to a global network of technical specialists, EPRI scientists and engineers develop innovative solutions to the world's toughest energy problems while expanding opportunities for a dynamic industry.

EPRI. *Powering Progress*



Printed with soy inks on recycled paper (50% recycled fiber, including 10% postconsumer waste) in the United States of America.