

J E W E L

Joint Environmental Workspace and Emissions Laboratory

---



---

## **Project Management Plan**

# 1. Introduction

Many areas in the United States and around the world are faced with serious air quality problems. In the United States, local, state, and federal agencies are engaged in efforts to identify where air pollution exists, the causes of air pollution, and how to control air pollution. The goal of the efforts is to bring an area that has been designated *out of compliance* with an air quality regulation *into compliance* with the regulation.

Air quality models are used to identify the extent to which air quality is exceeded in an area. Furthermore, the air quality models are used to help identify measures that should be taken to abate air quality problems. The air quality model simulates conditions in the atmosphere both in time and in space.

The rules and regulations for the air quality models established by direct legislation typically set limits on air emissions to the atmosphere for various sources of emissions (power plants, dry cleaners, automobiles, etc.).

Emissions estimates over the study domains are major inputs to the air quality models. Emissions estimates must be prepared for each hour of the days that the air quality model is going to simulate. Furthermore, air quality study domains are gridded; therefore, emissions must be estimated for each grid cell in the domain. The tool that is used to prepare emissions estimates for a particular study domain is an *emissions modeling system*. Examples of emissions modeling systems currently in use include the following:

FREDS -- the Flexible Regional Emissions Data System

EPS -- the Emissions Preprocessor System; and

GEMAP -- the Geocoded Emissions Modeling and Projections System.

The JEWEL System will give an environmental planner the ability to compute emissions, visualize them, and attach them to geographic entities. It will allow monitoring of allowable emissions and notification of an enforcement agency if any violations are detected, and will permit getting up-to-date information from various sources, which would allow the end user to deal with the proposed placements of pollutant sources, such as smoke stacks, in a knowledgeable way.

The project starts on Aug 30, 1994 and finishes at the end of the semester. One of the project goals is to incorporate the JEWEL system into Andrew, the campus wide information network system installed at Carnegie Mellon University. Another goal is to make its technology and a prototype available to the clients of the project.

In section Overview we give an executive summary of the JEWEL system consisting of a description of the project, the deliverables, a plan for the evolution of the project. The organization of the project is described in Section Project Organization. The main purpose of this project is to provide a vehicle for students to get hands-on experience with the technical and managerial aspects of a complex software system. These aspects are described in Sections Managerial Process and Technical Process. Section Work Elements contains a description of the work packages and the schedule.

## 1.1 Project Overview

The initial use of the JEWEL system will likely be for an ongoing air quality study in the Northeastern United States to determine the optimal emissions limits to set throughout the Northeast. The JEWEL system will have a substantially faster run times than current systems which operate over a period of days in some cases. It should also provide a migration path from GEMAP and EPS since many agencies are currently tied to the use of one or the other systems.

The system will be able to operate on arbitrarily formatted input data sets. JEWEL will be able to not only operate on the inputs which are used to generate the emissions estimates, but the system will also be able to maintain a change history on any file that is modified. The information that is utilized in the JEWEL system will be able to be tagged by a limited hierarchy of political boundary information. In turn, the system will supply means to visualize the data as tagged to political boundary.

The schedule with project phases and milestones are shown below in Table 1.

**Table 1:**

Date	Project Phases	Project Milestones
Aug 30		Project Presentation by Clients
Aug 30 - Sep 15	Requirements Engineering	
Sep 1- Sep 15	Project Planning	
Sep 16-Oct 6	Requirements Analysis	
Oct 18-20		Analysis Review
Oct 11 - Oct 26	System Design	
Oct 27-Nov 9	Object Design	
Nov 1		Client Project Review
Nov 3 - Nov 29	Implementation & Unit Testing	
Nov 10		Object Design Review
Nov 30- Dec 7	System Integration and System Testing	
Dec 5		Internal Project Review
Dec 8		Project Acceptance by Client

## 1.2 Project Deliverables

The JEWEL System will produce a running system that works in the Andrew environment and passes the acceptance suit test as described in the requirements analysis document.

The following items will be produced by the JEWEL System:

- A user manual describing how to use the JEWEL System
- A Software Project Management Plan . defining the technical and managerial processes necessary for the development and delivery of the JEWEL system.
- A Project Agreement between client and developers, representing a contract between the client and the developers of what is going to be delivered.
- An Analysis Document describing the requirements of the system in terms of three types of object models, the object model, the functional model and the dynamic model and the global requirements. This document is used by the application domain expert as well as the system designer.
- A System Design Document describing the design goals, trade-offs made between design goals, the high level decomposition of the system, concurrency identification, hardware/software platforms, data management, global resource handling, software control implementation and boundary conditions. This document forms the basis of the object design. This document is read by the analyst as well as the object designer.
- A Object Design Document describing the system in terms of refined object models, in particular the chosen data structures and algorithms as well as full signatures for all public methods. this document results in the detailed specification of each class used by the programmers during the implementation phase.
- A TestManual completely describing the unit and system tests performed on the JEWEL system before delivery along with expected and actual results. This document is used by the developers and maintainers
- Source code for all subsystems of the JEWEL System.

The JEWEL System documentation will describe the principles of operation, but it will not contain installation instructions, training aids nor will it describe maintenance procedures. The delivery consists of a presentation of the system, a demonstration of the working system and the successful passing of the acceptance test.

The presentation of the project will be given at Wean Hall, Room 5427 on Dec 8, 1994. A demonstration of the acceptance test will be given at the same time.

### 1.3 Evolution of the Software Project Management Plan

The software project management plan is under version control. Proposed changes and new versions of the plan are announced on the Andrew bulletin board 15-413.announce and are made available to all the project members.

Changes to the previous version are marked with a vertical bar, as shown at left of this section.

### 1.4 Reference Materials

The following technical documentations are used by the project:

- James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William Lorensen, *Object Modeling and Design*, Prentice-Hall 1991.
- Grady Booch, *Object-Oriented Design with Applications*, Benjamin Cummings, 1991, Chapter 1.
- Grady Booch, *The Booch Method: Process and Pragmatics*, Rational, Inc., Santa Clara 1992.
- Ian Graham, *Object Oriented Methods*, Addison Wesley, 1991, Chapter 8
- Edward Yourdon, *Decline & Fall of the American Programmer*, Yourdon Press, 1992, Chapter 5, Chapter 10.
- IEEE Standard for Software Configuration Management Plans, ANSI/IEEE Std. 828-1990.
- IEEE Guide to Software Configuration Management ANSI/IEEE Std. 1042-1987.
- IEEE Standard for Software Project Management ANSI/IEEE Std. 1058.1-1987.
- IEEE Standard for Developing Software Life Cycle Processes, ANSI/IEEE Std. 1074-1991.
- Software Engineering Syllabus, 15-413 Fall 1994, Handout #1.
- JEWEL System: Problem Statement, 15-413 Fall 1994, Handout #2.

### 1.5 Definitions and Acronyms

SPMP	Software Project Management Plan
OOSE	Object Oriented Software Engineering
RCS	Revision Control System

## **2. Project Organization**

### **2.1 Process Model**

The project is initiated on Aug 30, 1994 and terminated with the end of the semester on Dec 14, 1993. Major milestones are the client project review on Nov 1, 1994 and the client review on Dec 8, 1994.

The project uses an object-oriented design methodology for the development of the software and is organized in several activities. At the end of each activity up to and including unit testing, each team submits documents describing the achievement of the activity. The individual documents produced by the teams are considered to be part of the software development documentation.

During the system integration activity the system is tested as a whole and the individual documents are integrated into a project binder. The system documentation will also be made available on a floppy disk. The activities and milestones are described in the next following sections.

#### **2.1.1 Project Planning**

Project planning includes description of project tasks, activities and functions, dependencies, resource requirements and a detailed schedule. This activity results in the software project management plan for the JEWEL System. Another output of the project planning phase is the project agreement, which is issued after the analysis activity is completed.

#### **2.1.2 Analysis**

The analysis phase consists of two activities, requirements analysis and robustness analysis. The requirements analysis activity takes the problem statement and reviews it in terms of consistency, completeness and feasibility. During this activity, a set of models of the proposed system is determined by interacting with the clients resulting in the requirements model. The main part of the requirements model is the use case model describing the complete functionality of the system. The requirements model serves as the starting point for the robustness analysis activity which generates another model, the analysis model, which specifies the logical structure of the system.

#### **2.1.3 Design**

The design phase consists of two activities: System design and object design. The purpose of the system design activity is to devise a system architecture that maps the analysis model to the chosen target environment. The major part of the system design phase is the design of subsystems, that is, the decomposition of the system with respect to the chosen target platform. The system design activity also refines the uses cases from the analysis model and describes in terms of interaction diagrams how the objects interact in each specific use case. Additional notation is used to show implementation decisions. The object design phase specifies in detail the type of each attributes as well as the signatures of the methods for each object.

#### **2.1.4 Analysis Review**

Review of project plan, requirements analysis and design. The meetings will take place on Oct 18 and Oct 20 from 9am-10:30 in WeH 5427. The Analysis Review will result in a set of presentation slides generated by members of the JEWEL project.

#### **2.1.5 Prototype**

This activity involves successful execution of a simple prototype of the JEWEL System using stubs. A first prototype of the JEWEL system is expected on Nov 1.

#### **2.1.6 Client Project Review**

Review of project plan, requirements analysis and design. The clients and other interested parties will be present. The meeting will take place on Nov 1 from 9am-10:30 in WeH 5427. The Client Project Review presentation slides will be made available to the client.

### **2.1.7 Implementation**

This activity will result in the completion of the coding of the individual objects.

### **2.1.8 Unit Testing**

During Unit Testing, test suites are designed and executed for team-specific objects. Unit testing enables the individual teams to proceed with their subsystem independent from the progress of the other teams. The result of this activity is part of the Test Manual describing how to operate the test suite and how to interpret the results.

### **2.1.9 System Integration**

During this activity a integration strategy is devised, that specifies the order in which the subsystems of the JEWEL system are integrated and tested with respect to the use cases defined in the analysis model. The system integration strategy and the subsystem tests are described in the Test Manual.

### **2.1.10 System Testing**

**Structural Testing:** This activity tests the major data paths in the complete JEWEL System.

**Functional Testing:** Tests the major functionality (use cases) with the complete JEWEL System. The basis for the functional testing activity is the user manual which is revised according to the results of the system testing phase.

**Alpha-test (Client Acceptance Test):** The system is tested to make sure it passes the client acceptance criteria as defined in the project agreement.

### **2.1.11 Manual Integration**

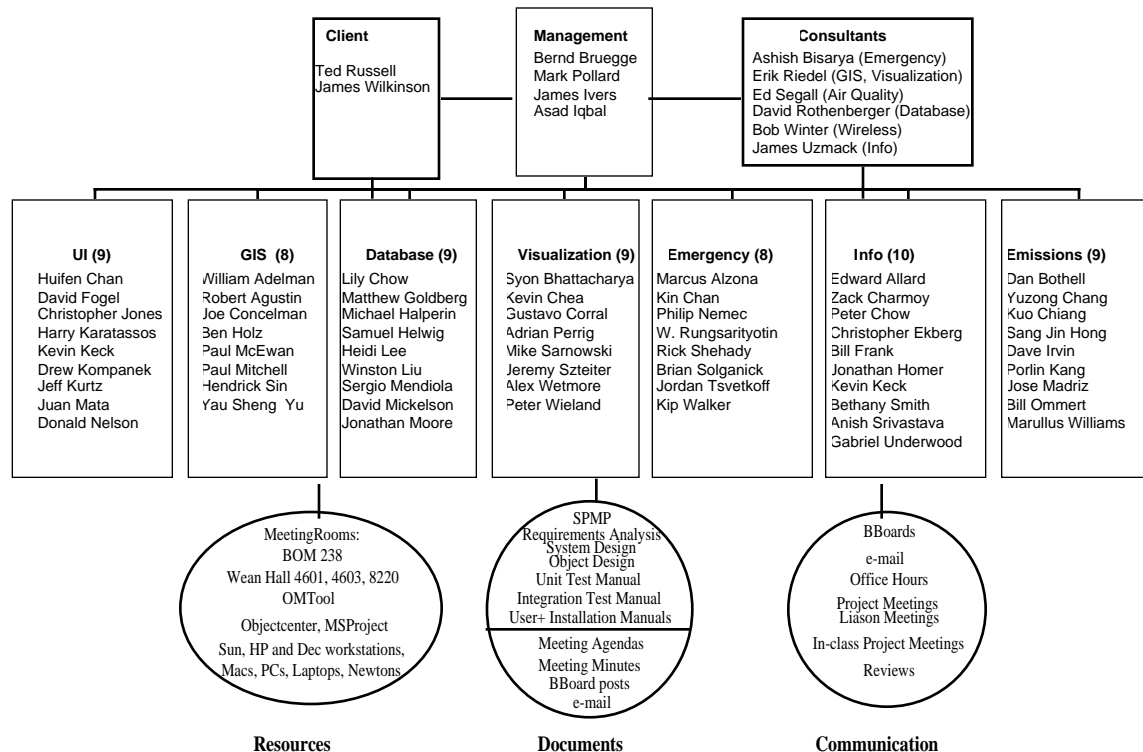
During this activity, the user manual is finalized and the project documentation is revised. As a result, a complete set of documents consisting of user manual, administrator manual, software project management plan, requirements document, analysis document, design document, test manual and source code will be created. The JEWEL System documentation will be printed on a CD and ready for client presentation. Each of the students taking the course as well as the clients will receive a CD.

### **2.1.12 Client Presentation**

At the Client Presentation, a slide presentation will be given and the client acceptance test will be executed. The client then accepts the system.

## 2.2 Organizational Structure

Below is the organizational chart of the JEWEL system indicating the position of each member in the overall structure.



The project managers can be reached as follows:

Bernd Bruegge bob@cs.cmu.edu  
James Ivers jivers@cs.cmu.edu  
Asad Iqbal ac46@andrew.cmu.edu  
Mark Pollard mpollard@cs.cmu.edu

The clients of the JEWEL System project are:

**Ted Russell**  
Department of Mechanical Engineering  
Carnegie Mellon University, Pittsburgh PA 15213  
email: russell@pollution.me.cmu.edu

**Jim Wilkinson**  
Department of Engineering and Public Policy  
Carnegie Mellon University, Pittsburgh PA 15213  
email: jw75@andrew.cmu.edu

**Michael Bookser**  
Chief of Police  
Bellevue Police Department  
email: bookser@andrew

Project-related communication with the clients has to take place via the client bboard (see section 6.3.1) not via e-mail. We expect that the client will be available for the client review to evaluate various aspects of the system design and implementation, as well as for the client acceptance test to accept the running system.

## 2.3 Organizational Boundaries and Interfaces

### 2.3.1 Electronic BBoard Communication

Several Andrew bulletin boards academic.cs. are available for the JEWEL System:

15-413.announce	Lecture and project announcements
15-413.help	Student help bboard (post only)
15-413.handin	For electronic submission of homework
15-413.client	For client-related discussions
15-513.database	Database
15-413.discuss	Group discussion
15-413.emission	Emission group
15-413.emergency	Emergency group
15-413.GIS	GIS group
15-413.info	Info group
15-413.ui	UI group
15-413.visualization	Visualization group

### 2.3.2 Meeting Times

There is a weekly project meeting for each group. The project meeting times are:

Group	Day	Time	Location
Database	Mon	3:00-4:00pm	BOM 238
Emission	Wed	4:00-5:00pm	Wean Hall 4601
Emergency	Monday	4:00-5:00pm	BOM 238
GIS	Tuesday	5:30-6:30pm	BOM 238
Info	Friday	3:30-4:30pm	BOM 238
UI	Wed	4:30-5:30pm	BOM 238
Visualization	Monday	5:00-6:00pm	BOM 238

A calendar has been created listing JEWEL group meeting times. To run the calendar tool from a Sun workstation type the following:

```
/afs/cs.cmu.edu/sun4_411/omega/usr/openwin/bin/cm &
```

To gain access to the group calendar:

- Select Multi-Browse from the Browse menu/button (use right mouse button)
- Enter mpollard@gs98.sp.cs.cmu.edu at the Calendar menu/button
- Select Add Calendar from the Calendar menu/button (use right mouse button)
- Return to the main calendar window
- Select mpollard@gs98.sp.cs.cmu.edu from the Browse menu/button (u.r.m.b.)

Please consult the calendar when scheduling further meetings. Send e-mail to mpollard@cs to

reserve the BOM-238 room. Unless otherwise indicated, all meetings take place in BOM-238.

## **2.4 Project Responsibilities**

Management of the JEWEL System is done with the following roles: project management, group leader, liaison, planner, document editor, configuration manager, developer and information management.

### **2.4.1 Project Management**

The project management function has the following responsibilities:

- Review weekly team reports
- Attend weekly team meetings
- Schedule and prepare meetings with clients
- Insist that guidelines are followed
- Assign presentations (In-class Project meetings, client review, client acceptance test) to project members.
- Resolves conflicts if they cannot be resolved otherwise
- Listening to gripes from the individual teams

### **2.4.2 Group Leader**

The group leader leads an individual team. The main responsibility of the group leader is to manage the action items of the group. In addition he or she has the following responsibilities:

- Responsible for intra-group communication
- Run the weekly project meeting
- Define, post and keep track of action items (who, what, when), i.e the agenda
- Measure progress and enforce milestones
- Deliver work packages for the tasks to the project management
- Coordinate and schedule use of resources (lab, tools,...)

The project leader is rotated on a regular basis among the team members. The change should occur after the end of each project phase.

### **2.4.3 Liaison**

The liaison interacts with the liaisons of the other teams and with the project management. Each team has a liaison. The responsibilities of the liaison are:

- Responsible for inter-group communication
- Make available public definitions of each subsystem to the other teams (ensure consistency, etc.)
- Coordinate tasks that overlap subsystems with the teams
- Responsible for team negotiations, that is, resolve technical issues spanning more than one subsystem

### **2.4.4 Editor**

The editor in each team is responsible for producing the documentation of the current project phase and:

- Collect, proofread and distribute team documentation to Info group
- Interaction with the Info group
- Collect minutes
- Manage tool task

### **2.4.5 Developer**

The developer is responsible for analysis, design, coding and testing of the individual mod-

ules. We expect each team member to be a developer.

#### **2.4.6 Configuration Manager**

The responsibilities of the configuration manager in each team are:

- Coordinate change requests
- Provide version control for group's working directory
- Coordinates configuration management issues with other groups
- Installation of group specific software and hardware

#### **2.4.7 Information Management**

The information manager maintains group information and is responsible for:

- Collecting and archiving project information
- Keeping track of the project history
- Keeping track of the design rationale
- Integrate team documents

Below are two tables describing the responsibilities for each team. Table 2 contains the group leader assignments and Table to 3 indicates the permanent team assignments

**Table 2: Group Leader Assignments**

<b>Group</b>	<b>Planning</b>	<b>Analysis</b>	<b>System Design</b>	<b>Object Design</b>	<b>Implementation &amp; Unit Testing</b>	<b>System Integration and System Testing</b>
Database	Jonathan Moore jmci	Sergio Mendiola sm9i	Lily Chow ic4u	David Mickelson dm60	Michael Halperin mh60	Heidi Lee hl1o
Emergency	Jordan Tsvetkoff jt3i	Brian Solganick bs3c	Marcus Alzona ma23	Wasinee Rungsarityotin wr2c	Jordan Tsvetkoff jt3i	Kip Walker kw27
Emission	Sang Jin Hong sh59	TBA	TBA	TBA	TBA	TBA
GIS	Ben Holz bh1s	Rob Augustin ra00	Billy Adelman wa00	David Yu dy27	Joe Concelman jc8l	Hendrick Sin
Info	Kevin Keck kk30	Bill Frank wf0i Sintha Nainggan lan sn1g	Peter Chow pc2s	Zack Charmoy zc00	Jon Homer jh85	Ed Allard ea0s Anish Srivastava as4w
UI	Juan Mata jmb3	Drew Komparek ak10	Jeff Kurtz jk78	David Fogel df2e	Chris Jones cj2f	Huifen Chanhc16
Visualization	Guastavo Corral gc3g	Guastavo Corral gc3g	Mike Sarnowski msaw	Mike Sarnowski msaw	Gabe Underwood gu02	Gabe Underwood gu02

**Table 3: Project Function Assignments**

<b>Group</b>	<b>Liaison</b>	<b>Editor</b>	<b>Configuration Manager</b>
Database	David Mickelson dm60	Lily Chow ic4u/ Matthew Goldberg mg4q	Michael Halperin mh60
Emergency	Brian Solganick bs3c	Phlip Nemec pn0q (1st half) Wasinee Rungsarityotin wr2c(2nd half)	Kip Walker kw27 Marcus Alzona ma23 (assistant)

**Table 3: Project Function Assignments**

<b>Group</b>	<b>Liaison</b>	<b>Editor</b>	<b>Configuration Manager</b>
Emission	Yuzong Chang yc0d, Dave Irvin di07	MarullusWilliams mw65, Dan Bothell db30, Bill Ommert wo07	Jose Madriz madriz@cmu, Porlin Kang pk2k
GIS	Ben Holz bh1s	Paul Mitchell pm3z	Sam Helwig sh4j, Paul McEwan pm3c
Info	Zack Charmoy zc00, Bill Frank wf0i, Chris Ekberg ce12, Jon Homer jh85	Chris Ekberg ce12	Anish Srivastava as4w
UI	TBA	Donald Nelson dn1r	Harry Karatassos hk0x
Visualization	TBA	TBA	TBA

## 3. Managerial Process

### 3.1 Management Objectives and Priorities

The philosophy of this project is to provide a vehicle for students to get hands-on experience with the technical and managerial aspects of a complex software problem. The emphasis is on team work and encouraging individual teams to work together towards the goal of implementing the JEWEL system complete.

### 3.2 Assumptions, Dependencies and Constraints

The functionality of the JEWEL System is achieved when the client acceptance test can be executed. We assume that the response time during the client acceptance test (the time between typing a command and receiving an answer) will be an order of magnitude better than the performance of the GEMAP if there is no delay in Andrew software.

We plan to implement the JEWEL System as a service in Andrew.

The software development proceeds in the following phases:

- Project planning
- Requirements Analysis
- System Design
- Implementation and Unit Testing
- System Integration and System Testing

Each phase results in one or more documents to be submitted to the project management before the deadline. Each document is reviewed at least once by the project management before it becomes a baseline document.

Each document is worth up to 10 points. We will give an A to everybody who participates in the project if the complete software system passes the client acceptance test as defined in the requirements analysis document. A different grade might be result due to individual adjustments made by the instructor. If the complete software system fails the acceptance test, an individual team can get an A, if it demonstrates that its subsystem passes its acceptance test in the testbed environment of the individual team and is ready for use by the other teams.

The JEWEL System is a project that puts emphasis on collaboration, not competition between the students. We will not accept a system that is done by one team alone.

#### 3.2.1 Assumptions

##### 3.2.1.1 Database Assumptions

- All members of the group will have unlimited access into BOM 238.
- All members of the group will have login access to the hp/unix/ andrew workstations in BOM 238.
- The system has no security access requirements.
- No bugs will be encountered in the selected DBMS, the compiler, or the class library. (They will perform correctly according to their specifications.)
- Hardware used in development has sufficient capabilities (ie. sufficient storage, reasonable response time ...)
- Data to be managed and algorithms for processing the data in GEMAP will be specified by the client and/or the Emissions group in the Requirements Analysis phase.
- The system will be developed on/for X using C++.
- A demonstration of the GEMAP system will be given by the client.

##### 3.2.1.2 Emergency Assumptions

- All chosen wireless communication packages will work according to their specifications.

- Hardware required and their specifications are always available. This includes sensing devices.
- We will always be able to select an appropriately supported communication package.
- Client will provide information about what kind of reactions are expected in response to different sensor results.

### **3.2.1.3 Emission Assumptions**

- We will reuse the existing Fortran code. Some of the code, such as Mobile Source Model, exceeds 100,000 lines. One of our goals is to design JEWEL such that the implementation of the models can be easily replaced later. Hence, we assume the correctness of the existing code.
- Fortran code will compile and work with our C++ code.
- Unix platform. We will develop the system to run on as many Unix platforms as possible. Thus, we will conform to standards such as POSIX.
- Sufficient resources(all members have access to Unix workstations). We assume enough storage (non-volatile and volatile) on the target computer.
- Visualization group will be able to display maps to verify our output.
- Users will correct their own data. We will not provide error correction capability.

### **3.2.1.4 GIS Assumptions**

- Prompt and accurate delivery of map data from the existing CD-ROM library, and prompt and accurate delivery of updates to the map information that might have been recorded during emergency sessions (building construction, disasters, location of emergency vehicles, areas of emissions, etc.)
- Maps are or can be converted into a format we can manipulate.
- Maps cover all the territory to be displayed and have no holes. 4. Assume the Visualization group will display the map data given to them by GIS.
- UI group will give appropriate information about the area of the map that the user wants to display, and the amount of information, detail, and views of the map the user wants to see. This includes three dimensional views, political boundaries, and the zoom factor of the maps. This will allow the GIS group to put the proper amount of information onto the map and properly scale the map.

### **3.2.1.5 Info Assumptions**

- In setting up the CVS we can figure out how to use CVS.
- In determining platforms that are desired for viewing the documentation there is a tool that will allow us to author to the Clients requirements.
- Once CD software/hardware is decided on, we can get it in a timely fashion
- In specifying a coding documentation standard standard will be followed and will not require supplements.
- A general assumption that we can find a documentation authoring soft/hard-ware.
- A general assumption that the directory tree structure has been established.
- A general assumption that enough and appropriate test data is available for unit testing
- A general assumption, to test CD performance, we need to have been able to create a CD.

### **3.2.1.6 User Interface Assumptions**

- The main language for the project, used at least for communication between modules will be C++. (It is possible to have a ui tool based around a non-c++ language, such as tcl/tk, but it needs to be compatible etc)

- Users of the project have a certain basic skill level in using the computer platform we develop our UI for. (they must be comfortable with, for example, switching between windows, using their input devices, etc)
- Hardware which we have access to for development will be sufficiently similar to that actually used by client/end-users. This includes things like input devices. We assume a mouse-and-keyboard-driven environment. Also, it doesn't appear that we can assume color displays, though that would be nice.
- Since one of our goals is to make a better UI than exists in GEMAP, we assume that there is the computational power/resources needed to make a better/good user interface possible.
- UI tools won't impose any unusual or catastrophic restrictions on our software project. (i.e. have big bad bugs)
- UI can communicate with other modules (not a big deal if everything is on unix, bigger deal if we use macs/pc's).
- Users will have a minimal level of knowledge about the software's usage- we don't have to teach them emissions science, or provide a separate interface for experienced vs novice users.
- Client, and a sample of end-users will be available for feedback info on UI.
- UI will have a minimum amount of processing power devoted to it such that we can count on a minimum speed-of-interaction- (that the end-user machine, if it is a multi-tasking machine, won't be running so much other software that our interface is drastically slower than we planned on, or slower than in the testing process.)
- The UI tools will not have any bugs that are sufficiently inconvenient to slow us down. (Not a trivial assumption as Galaxy had a few bugs at Xerox)
- There will be no conflicting groups (other classes) using the equipment we need in the BOM 238 lab. The ui group will therefore be able to work at all hours.
- Privileged levels of access or superuser use or access of certain data that is displayed by the user interface is not part of this lab.
- Hardware/software configuration on top of which the JEWEL system is built, is performing sufficiently well so that the network is up at least 90% of the time and computational environment mean time to failure is one year.

### **3.2.1.7 Visualization Assumptions**

- System and hardware available is sufficient for system development.
- User Interface group will give timely specifications of user interface API.
- We will have enough programmers available to complete the project.
- We will learn C++ sufficiently for completion.
- Requirements will remain the same throughout the project.
- Our communication with the Database and the UI components will be through straightforward C++ calls (message passing).
- Group will have skills necessary to complete project, such as statistics knowledge and graphics programming.
- Group members will cooperate.
- Configuration managers come up with an intergrateable build environment.
- Computing resources will be sufficient; we will have enough access to the necessary workstations.

## **3.2.2 Dependencies**

### **3.2.2.1 Database Dependencies**

- Data for the database is supplied by the Emissions and GIS groups.

- The Database group depends on the availability of class library on the chosen platform.

### **3.2.2.2 Emergency Dependencies**

- The emergency responses will have a right to suspend or cancel any modelling or computation during emergency. Thus, we depend on the Emission group to provide such a mechanism.
- All the other groups will complete their tasks on schedule and with correctness.

### **3.2.2.3 Emission Dependencies**

- GIS group must provide an accurate coordinate system.
- The GEMAP source and documentation will arrive on time so we can start our design phase.
- Manuals for SAS and ARC-INFO will be available on time so we can understand the GEMAP code.
- Database group provides us with data for our modeling module.
- Availability of the class library on the chosen platform.
- Keys for BOM 238 distributed to all group members.
- A smaller data set for rapid testing of our module.

### **3.2.2.4 GIS Dependencies**

- Emissions group provides pollution levels and areas of pollution, given in the established map coordinate system. An option is for the Visualization group to get the pollution information directly from the Emissions Group and then combine that data with map information supplied to Visualization from the GIS group. This issue must be decided by the group liasons.
- Emergency group provides the location of emergency vehicles so they can be displayed on the map.
- Database group must determine map storage and retrieval parameters, since the database group determines project data structures.
- UI Group needs to tell us what kinds of map data (such as map area and map detail) the user wants to display, and what kinds of map transformations (such as zooming) the user wants. 5. Depend on the Visualization group to draw the maps from data provided from the GIS group.
- Client is to determine exactly what kinds of map transformations the user will need the final system to do.

### **3.2.2.5 Info Dependencies**

- In determining platforms that are desired for viewing the documentation we require client feedback on viewing platforms required and what are the most important ones.
- In researching class libraries, we depend on the other groups class needs.
- In developing the first prototype we depend on the prototype specification to be able to produce the prototype. - we depend on the arrival of the hardware to be able to deliver the prototype on CD.
- We depend heavily on the UI group for generation of the prototype.
- In user documentation, when we begin writing documentation we depend on prototype and analysis describes enough functionality to allow this
- We require the backbone structure and standards have been set.
- To write user documentation we need to have decided on an authoring system.
- We heavily depend on other groups having implemented as promised.
- System testing depends on implementation and system integration.

### **3.2.2.6 User Interface Dependencies**

- Visualization and GIS groups must supply the information for screen displays.
- Client approves screen layout and requires certain features.
- Consistent communications across time and distributed environments for all issues that pertain to the integration phase.
- Project leaders (Bernd Bruegge, the TA's, and the Info group) to lead us through certain stages of the development successfully.

### **3.2.2.7 Visualization Dependencies**

- Database group provides reliable source of data.
- GEMAP Demonstration will occur.
- We will be able to get sufficient test data from database group.
- A strong and concise interface will be decided by the liasons.
- The project breakup will occur in a timely fashion.
- Valid test data will be available.

## **3.2.3 Constraints**

### **3.2.3.1 Database constraints**

- We have a limited amount of time to build the system.
- Our communication takes place distributed over time and space. Difficulties may be encountered in keeping everyone up-to-date.
- There is no budget for software purchases.
- The analysis tool OMTool is already chosen.
- It will take time to learn how to use various tools such as OMTool and the selected DBMS.
- The programming language C++ is already chosen

### **3.2.3.2 Emergency constraints**

- Wireless communication functionalities are only limited to Newtons and/or IBM-Thinkpads.
- Software packages, programming language, development platform, management tools and hardware have already been chosen.
- Our system has to comply with the EPA regulations

### **3.2.3.3 Emission constraints**

- We have a limited amount of time to build the system.
- The additional time required to train group members for C++, OMTool, SAS, and ARC-INFO.
- We may run out of computer resources during peak hours.
- No budget is allocated for this project.
- We have to allocate 2 to 3 weeks to test each prototype release. Thus, we need tight coordination between groups.
- We have chosen OMTOOL and C++

### **3.2.3.4 GIS constraints**

- Map attributes that will be required such as three-dimensional display, color, and rotation is subject to decision among the UI and Visualization group liasons. More attributes may require more detailed information from the GIS group.
- The exact functional boundaries that the GIS group must provide depends on the decisions made by the GIS, Visualization, and UI group liasons. This is to avoid

duplication of functionality in the existing system, minimize the amount of new coding required, and allow the most re-use of GEMS code as possible; this will lead to true plug and play compatibility with the existing system.

- Limited time of members to work on the project.
- Limited time to learn about the input and output format of the existing CD-ROM map library.
- The map format we will support depends on the current format of the maps in the CD-ROM library. Even if other map input formats are available, we don't have the time, resources and manpower to write a second input module.
- 6. The resolution of our maps are constrained by how detailed our given maps are.
- Maps are only accurate to the time period when the CD-ROM map library was created. New construction, disasters, and other recent changes cannot be displayed unless reported by the Emergency group to the GIS group.
- User requests for map manipulations and map display are provided by the UI group.

### **3.2.3.5 Info constraints**

- Constraints in finding the meeting time for the weekly meeting, since we have to incorporate the schedules of Bernd Bruegge, James Uzmack and the nine other group members.
- Meeting room - only one room. Conflicts with other groups.
- There are no Sun workstations in the meeting room.
- In determining platforms that are desired for viewing the documentation, we must author on Mac's, and we can not impose a software license fee on the client for the documentation viewer
- Client needs dictate the viewing platforms for documentation
- OMTool is used for object design. This must be incorporated in integrating design documentation
- We have no code at this point, so the prototype will be limited.
- Time constraints mean non-exhaustive unit testing.

### **3.2.3.6 User Interface constraints**

- Andrew (Unix) machines are the <<most>> widely available machines for development.
- Interprocess communication and concurrency will probably be necessary to coordinate the various components of system while allowing the user to interact with the interface.
- Well defined easy to use programmer's interface must be provided for emissions, GIS and visualization since these groups work is highly integrated with our own. There is the need for sophisticated communication. 4) Most students are familiar primarily with the Unix programming environment. For example, all members of the Visualization group are highly knowledgeable in Unix.
- There is a limited amount of time to complete the project. We therefore must simplify as much as possible how our group's components interact with other groups' components and re-engineer/re-use existing software when possible. Because of the time constraint, we need an interface builder and library with a sufficiently flat learning curve.
- Need to define the conceptual interfaces as soon as possible so the work can be partitioned between groups. An interface prototype will facilitate this by allowing us to visualize the problem.
- C++ is the language of choice for this class. Object oriented programming techniques should be used. We need a development environment that will allow complex communication while maintaining OOSE.

- The need for the user to interact with arbitrary user elements (for instance, a map and icons representing emissions) makes it vital that we pick a system to design our interface in with sufficient flexibility including programming our own widgets and being able to go to a lower level if necessary to program features not directly supported. Must be a convenient way for each group to manage its own interface functions.
- Need version control and other mechanisms necessary to flexibly support the simultaneous work of many people.
- The need to use re-engineering principles in order to complete the project in time necessitates the need to consider the availability of sources on the platform we choose.
- Platform choice is dependent on whether we choose Evolutionary or Revolutionary Prototyping.
- There is no/(or extremely limited?) budget for software. The hardware budget is also a scarce resource.
- Not everyone will be as knowledgeable as others when the UI tools are selected. There will be a learning curve associated with development no matter which tools are chosen.

### 3.2.3.7 Visualization constraints

- OMTool will be used for object management.
- System must run on Unix platform.
- Run on HP's, portable to other platforms.
- X-Windows will be used.
- Use of C++ and object-oriented style is required.
- System must run in 24 megabytes of memory.
- Visualization must run under gray scale displays.
- Group must meet milestones set by project management.
- CVS must be used for source management.

## 3.3 Risk Management

We assume that the hardware/software configuration on top of which the JEWEL System is built, is performing reliably.

The individual team risk management assessments are as follows:

### 3.3.1 Database risk management

- Members of Database group drop out of the course.  
Contingency: Roles are assigned to other group members. Renegotiate the interfaces with other groups to try to cut down on our own responsibilities.
- Interface between Database and another group fails.  
Contingency: Liason must renegotiate where the interface is. All groups involved work together to find problem.
- Suitable DBMS is not found.  
Contingency: Database group writes our own, but we expect that this will take too much time and may be unsuccessful.
- Database group falls behind schedule.  
Contingency: Set up more meetings. Also cut down on functionalities to make it simpler.

### 3.3.2 Emergency risk management

- The client cannot provide the guidelines to respond to sensor outputs.

- Contingency: We will have to obtain this information from local agencies.
- The emission group cannot provide info/data needed.
  - Contingency: We will still provide a wireless interface, but portions of the functionality will not be immediately available.
  - Contingency: We will have to obtain this information from local agencies.
- Other groups fail to complete their tasks on schedule.
  - Contingency: We will use stubs for interfaces to their subsystems.

### 3.3.3 Emission risk managements

- Group members drop the course.
  - Contingency: The group will assign two persons per role. Specifically for the liaison role, the emissions group will announce who these persons are to ensure that the rest of the groups are not affected by a missing liaison. Internally, the group will reassign tasks evenly among its remaining members. In the worst case of two persons with the same role assignment dropping the course, a reassignment of group roles will follow; again, if the affected key role is the liaisons, an immediate public announcement will take effect.
- Project falls behind schedule.
  - Contingency: The group will schedule extraordinary meetings. If it is due to the lack of effort of one of the group members, or due to uneven task assignments (task overload of a subgroup) then the group will consider a task reassignment which accommodates both the interest and the time constraints of all members. Also, the group will devote special attention in the requirement analysis phase to ensure that the client demands are feasible and attainable within the development time frame.
- Other groups fail to produce for us.
  - Contingency: The group will try to reschedule tasks in such a way that tasks concurrent with other groups are not interdependent. If all the group's tasks are dependent from other groups outcome, the liaisons will discuss picking up some of their work.
- Loss of data while in development process.
  - Contingency: The group will keep a backup directory and update it regularly. If data in backup directory is also lost, contact computer services for Andrew backups.
- Executable version of JEWEL runs slower than GEMAP.
  - Contingency: The group will devote a specified time period in search of the existing bottleneck. If no success was attained within this time frame, the old code and algorithms will be incorporated to the system.
- Client not available in development process.
  - Contingency: If a crucial decision is to be made and expected to be reviewed by the client, the group will come up with a collective decision involving all group members and if necessary other group's liaisons. If source availability is a problem, the group will try to reassign tasks which are independent of the missing sources.
- Current source is unreadable.
  - Contingency: The group will try to get information from outside sources, primarily the client.

### 3.3.4 GIS risk managements

- Members of the GIS group might drop the course.
  - Contingency: Arbitrate with other groups in an effort to reduce the GIS group's burden.

- The interface between the GIS system and the Visualization system does not work.  
Contingency: Talk between GIS and Visualization group liasons in an effort to discover potential problems. Find out where the problems are and work together to correct the interface problem.
- GIS system may not contain all of the functionalities of the current system(ARC/INFO).  
Contingency: Meet with the client and determine exactly what details and what aspects are absolutely necessary and what can be overlooked.
- There is not enough time to develop all of the desired funtionalities due to strict project deadlines.  
Contingency: Develop an importance heirarchy relationship plan amongst the various functions. Build the most important main functions first and leave the minute details until later. Include stubs as necessary in the GIS module to provide an interface for other groups for GIS functionalities not yet implemented.
- GIS group code is damaged or destroyed due to a computer malfunction.  
Contingency: Return to the prior regularly saved backup version made of the code and proceed from there.
- GIS Group members cannot become proficient enough in C++ to complete required coding.  
Contingency: Seek outside help from someone who knows C++ well and can either assist in the coding or assist group members in becoming proficient.

### 3.3.5 Info risk managements

- In coordinating subgroup analysis a subgroup doesn't complete analysis, or there is an inconsistency in the analysis  
Contingency: Formal liason meetings to establish better communication.
- There is overlap between subgroups in the analysis.  
Contingency: Delegate tasks to eliminate overlap
- In setting up CVS, CVS doesn't meet the project needs.  
Contingency: Find other configuration management tools for subsystems.
- We can't find authoring tool or hardware for all required platforms.  
Contingency: Drop one or more platforms and determine best option for authoring toolkit without dropping all of the clients request.  
Contingency: In addition, provide paper documentation.  
Contingency: Provide a postscript files that can be printed from the CD
- One group doesn't have something to add to the prototype.  
Contingency: We can do something ourselves (in Macromind Director or Mac-paint)
- We can't find a class library that suits the needs of all subsystems.  
Contingency: Find the closes match and alter it to fit our needs.
- We don't have hardware for pressing CDs by prototype stage.  
Contingency: Prototype delivered on floppy disks and paper documentation
- Learning curve of authoring system is too steep.  
Contingency: Restrict learning of the authoring system to a few group members and restrict the rest's activity solely to documentation.
- Subgroup falls behind in implementation  
Contingency: Add / focus manpower to increase effort  
Contingency: cut feature and document as non-implemented  
Contingency: Maintain interface to prevent blocking other subsystems.

- Radical changes occur in the subsystems (esp. the user interface).  
Contingency: Make the group that made the changes update documentation.
- Not enough time to proof-read.  
Contingency: Delegate proofreading to other groups.
- CD Performance is not good enough.  
Contingency: Modify layout of the files on the CD.  
Contingency: Generate supplemental paper documentation.  
Contingency: Condense what is presented on the CD to increase speed.
- Can't make all the copies of the CD in-house.  
Contingency: Make a master for client approval, and make a master to be sent out from 3rd party
- Can't get copies from copying company.  
Contingency: We (class members) can get them later (next semester).

### 3.3.6 User Interface risk managements

- Members that hold key roles drop the course.

Case A: The member in question is responsible for a large chunk of work, e.g. a part of the code for UI.

Contingency: Distribute his work as evenly as possible to the rest of the team.

Case B: The role of the member in question cannot be divided, e.g. he is a liaison.

Contingency: Reassign the role. Require that each liaison keep records of their discussions with their corresponding group so that time needed to get the next liaison up to speed can be minimized should the current liaison drop the course.

Case C: He plays a role that just cannot be replaced (e.g. He is the only one that knows the UI builder we have selected).

Contingency: Only select an UI builder that more than one person knows. If that is not possible, and assuming it is too late to change the builder, team members will need to work even harder since the learning curve will be steepened without a person familiar with the builder to point the team to the essentials.

- The project is falling behind schedule.

Case A: If it is due to only one or two person and it is not due to lack of effort (either the person does not have the skill or task assigned to him too large).

Contingency: The rest of the team can help out. If the rest of the team is on time but flooded with work, see case b. If it is because of lack of effort, pressure the person.

Case B: If the whole team is falling behind.

Contingency: Renegotiate the functionality of UI with the client and with other groups and at the same time, work harder so that the renegotiated functionality is delivered.

Contingency: In general, avoid making promises that we are not confident of keeping. Agree first to only basic functionality and add on to it if time permits.

- The UI group does not function properly and does not provide the functionality needed by another group.

Contingency: The liaisons of both groups get together to solve this problem.

- The customer is not available for discussing and reviewing the user interface during development.

Contingency: If the client cannot make the selected date, reschedule a time convenient for the client. Only a few members of the team need to go for the meeting if the entire team cannot make the rescheduled time. The members that met the client can the report back to the team.

- The platform of the UI is not the same as the rest of the teams.
  - Contingency: Discuss with the rest of the groups before deciding on the platform.
  - Contingency: Support communication between the different platforms used.
  - Contingency: The customer is unhappy with one or more aspects of the look and feel or design of the user interface
  - Contingency: Determine if the interface can be redesigned and implemented in the available time. If not then we determine if the change should be made, sacrificing other functionality.

### 3.3.7 Visualization risk managements

- Key members quit/drop.
  - Contingency: Get other members from other groups.
  - Contingency: Get client to simplify requirements.
- Can't get GEMAP demo early enough.
  - Contingency: Talk to client.
  - Contingency: Cut back on requirements.
  - Contingency: Detailed discussion with client about requirements.
- System, hardware available is not sufficient.
  - Contingency: Will talk to project management.
- Group cannot get up to speed on programming requirements.
  - Contingency: Redistribute coding tasks.
- Group cannot manage other skills necessary (statistics, graphics knowledge).
  - Contingency: Actively study topic.
  - Contingency: Look for other class members.
- Group is stubborn and does not cooperate
  - Contingency: Talk to project management about personell re-evaluation.

## 3.4 Monitoring and Controlling Mechanisms

For each project meeting each team produces an agenda and the minutes of the meeting. The agenda and minutes are posted on team specific bulletin boards by the documentation editors.

The baseline documents are reviewed by the project management. It is expected that each document undergoes several iterations.

## 4. Technical Process

### 4.1 Methods, Tools and Techniques

Our development methodology is based on the the OMT methodology described in Rumbaugh's et.al book "Object-Oriented Modeling and Design", Prentice Hall 1991.

The following tools are available to support the management and development of the JEWEL project:

- **MS Project:** A project management tool for planning and tracking projects
- **OMTool:** A case tool for the preparation of object models.
- **ObjectCenter, Borland C++:** A set of tools for the "back end" of software development, compilation, editing and debugging of C++ programs

### 4.2 Software Documentation

The documents to be produced by the JEWEL project are described in section Project Deliverables.

Displayed in Appendix A is the documentation plan for the JEWEL project. The documentation will be manufactured in accordance with the dates listed on this chart.

**<<To be announced by the Info Team>>**

### 4.3 Project Support Functions

The following project support functions are defined. Each of the development teams is responsible for one of these support functions.

**Table 4: Project Support Functions in the JEWEL Project**

<b>Support Function</b>	<b>Description</b>	<b>Group</b>	<b>Due Date</b>
Standards	Specify documentation and coding standards to be used by the JEWELproject.	Liaisons	Oct 1
Version Control	Specify a configuration management scheme to be used by the JEWEL project.	Configura- tion Managers	September 25
Makefiles	Define a project make file to be used during implementation, system integration and system testing.	Configura- tion Managers	September 25
Database	Survey,select and maintain a database to be used by JEWEL	Database	Oct 1
User Interface	Survey, select and maintain a user interface builder	UI	Oct 1
Communication	Survey and select a wireless communication protocol to be used by JEWEL	Emergency	Oct 1
Integration	Keep track of system integration status	Info	Nov 30- Dec 7
CD Printing	Print all project artifacts on CD and make it available to every project member	Info	Dec 8

#### 4.3.1 Environment Variables:

Two versions of the JEWEL System will be produced:

- Rapid prototype

- Acceptance test version

A well-defined master directory described by the UNIX environment variable JEWEL will be the root of the following subdirectories:

JEWEL is set to `/afs/cs/project/classes-bob/project/JEWEL/`

The `doc` subdirectory contains the documentation associated with the various project phases.

The directories `prototype` and `stable`, respectively, have the following subdirectories:

- `bin`: Command files and executables.
- `include`: Include files for the JEWEL System.
- `man`: UNIX specific man pages.
- `lib`: A library providing the functionality of the JEWEL System.
- `src`: Source code for the modules.
- `test`: Test suite for unit testing. Driver programs that test individual tasks.

To permit rapid prototyping of the JEWEL System, stub versions for the modules are created and linked into a stub library in the `prototype` directory.

For the client acceptance test the fully functional modules are placed into the `stable` directory.

The UNIX environment variable `CONFIG` has to be used to describe the current execution environment and can be set to `'stub'` or `'stable'`.

The client acceptance suite has to run in the `stable` configuration.

A default setting of `CONFIG` should be in your login file, for example when testing the client acceptance suite, the setting should be:

- `setenv CONFIG stable`

Libraries will have the name `libJEWEL$(CONFIG).a`. Two versions of the library will be maintained and kept under version control:

- `libJEWELstub.a`: Stub version of the JEWEL System.
- `libJEWEL.a`: Stable library of the JEWEL System.

Each team sets up a working directory with the same structure described above. It is recommended to set up version control schemes for these directories as well - at least for the `src` and `doc` subdirectories.

## 4.4 Work Elements, Schedule and Budget

See attached MacProject Schedule Charts, Task

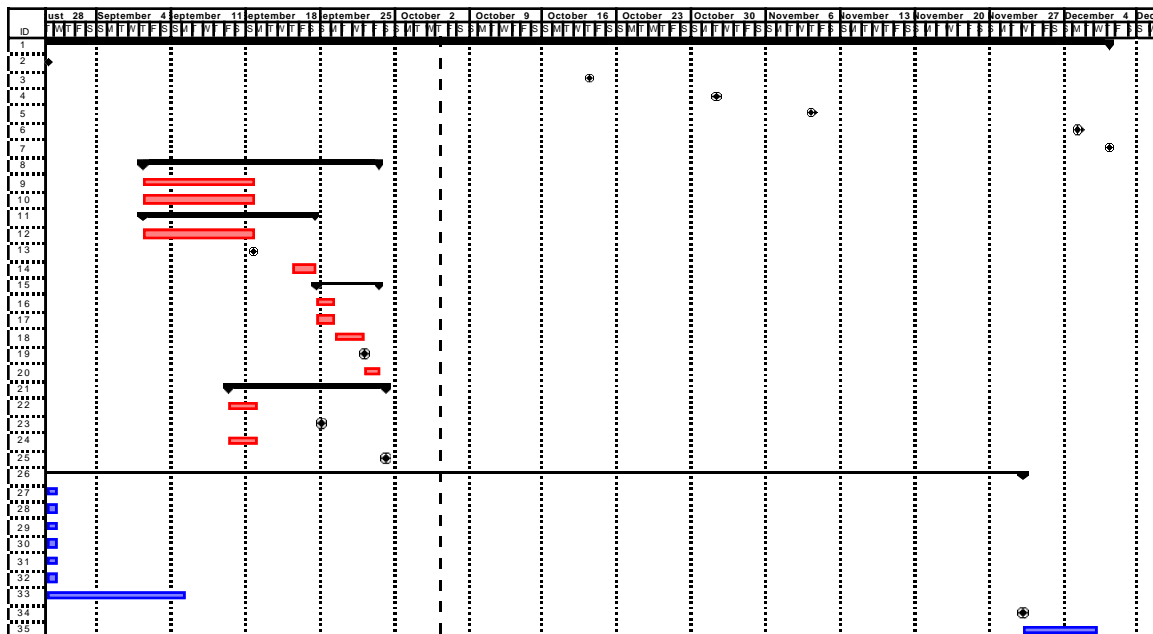
Timelines, Project Tables and Resource Tables for individual teams.

### 4.4.1 Project Schedule

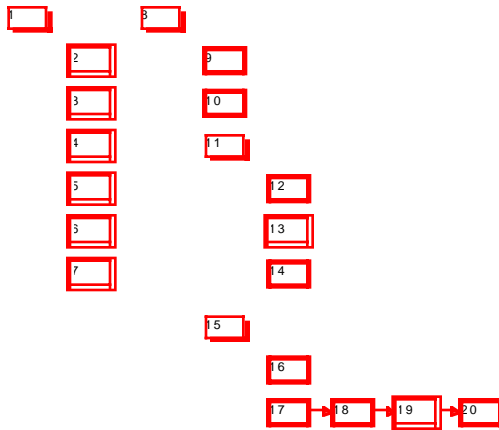
**Project Task Sheet:**

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	Project Milestones	72.19d	8/30/94 9:00am	12/8/94 10:30am	
2	Project Presentation By Clients	1.5h	8/30/94 9:00am	8/30/94 10:30am	
3	Analysis Review	2.19d	10/18/94 9:00am	10/20/94 10:30am	
4	Client Project Review	1.5h	11/1/94 9:00am	11/1/94 10:30am	
5	Object Design Review	1.5h	11/10/94 9:00am	11/10/94 10:30am	
6	Internal Project Review	1.5h	12/5/94 9:00am	12/5/94 10:30am	
7	Project Acceptance by Client	1.5h	12/8/94 9:00am	12/8/94 10:30am	
8	Project Planning	16.88d	9/8/94 9:00am	9/30/94 5:00pm	
9	Delegation of Group Roles	6.88d	9/8/94 9:00am	9/18/94 8:00pm	
10	Set Weekly Meeting Schedule	6.88d	9/8/94 9:00am	9/18/94 8:00pm	
11	Write SPMP	11.88d	9/8/94 9:00am	9/24/94 5:00pm	
12	Individual Groups Write SPMPs	6.88d	9/8/94 9:00am	9/18/94 8:00pm	
13	Group SPMP Deadline	0d	9/18/94 8:00pm	9/18/94 8:00pm	
14	Integrate Group SPMPs	1.88d	9/22/94 9:00am	9/24/94 5:00pm	
15	Write Project Files	5d	9/24/94 3:00pm	9/30/94 5:00pm	
16	Write Master Project File	0.13d	9/24/94 3:00pm	9/26/94 9:00am	
17	Write Group Project Template	0.13d	9/24/94 5:00pm	9/26/94 9:00am	
18	Write Group Project Files	3d	9/26/94 9:00am	9/29/94 9:00am	17
19	Deadline for Group Project Files	0d	9/29/94 9:00am	9/29/94 9:00am	18
20	Integrate Group Project Files	1.88d	9/29/94 9:00am	9/30/94 5:00pm	19
21	Initial Project Support	10.88d	9/16/94 9:00am	10/1/94 9:00am	
22	Set Up CVS	1d	9/16/94 9:00am	9/19/94 9:00am	
23	Configuration Support Deadline	0d	9/25/94 9:00am	9/25/94 9:00am	22
24	Find Class Library	1d	9/16/94 9:00am	9/19/94 9:00am	
25	Second Support Deadline	0d	10/1/94 9:00am	10/1/94 9:00am	24
26	System Development	66d	8/30/94 9:00am	11/30/94 9:00am	
27	Database Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
28	Emissions Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
29	Emergency Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
30	GIS Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
31	UI Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
32	Visualization Subsystem	1d	8/30/94 9:00am	8/31/94 9:00am	
33	Documentation (Info)	3d	8/30/94 9:00am	9/1/94 9:00am	
34	Unit Implementation Deadline	0d	11/30/94 9:00am	11/30/94 9:00am	27,28,29,30,31,32
35	System Integration Tests	5d	11/30/94 9:00am	12/7/94 9:00am	34

**Project Gantt Chart:**



### Project PERT Chart:

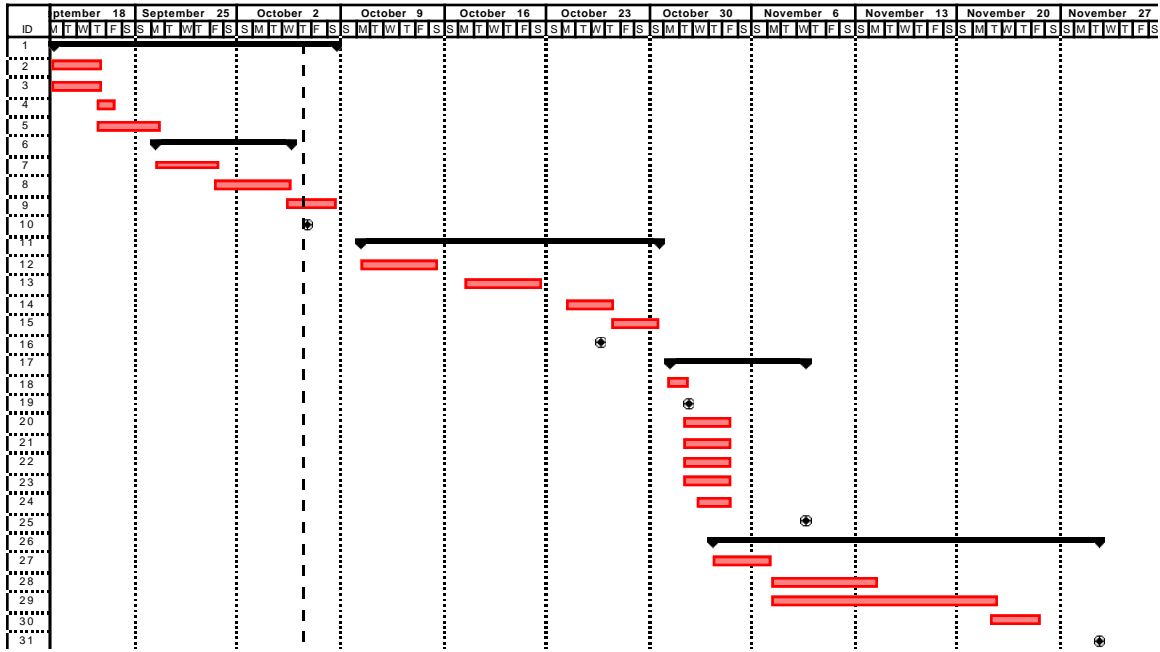


### 4.4.2 Database Schedule

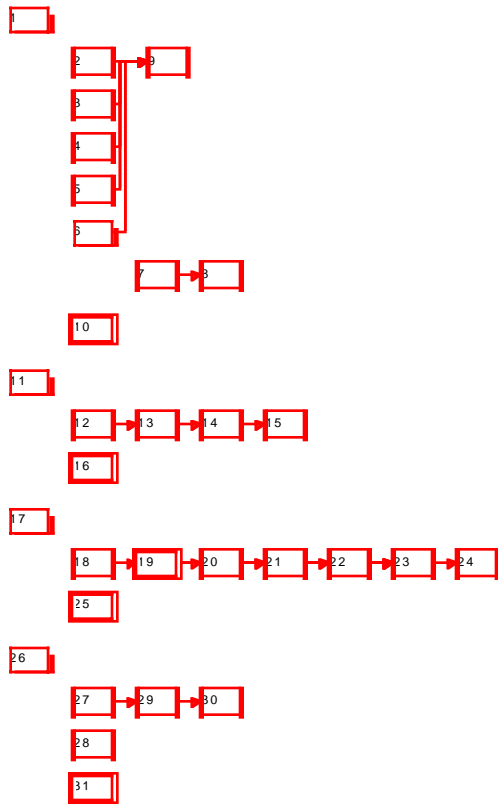
#### Database Task Sheet:

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	Analysis	15d	9/19/94 8:00am	10/8/94 5:00pm	
2	Establish number of users	4d	9/19/94 8:00am	9/22/94 5:00pm	
3	Establish system costs	4d	9/19/94 8:00am	9/22/94 5:00pm	
4	Establish end user functionality	2d	9/22/94 8:00am	9/23/94 5:00pm	
5	Establish desired data	3d	9/22/94 8:00am	9/26/94 5:00pm	
6	Intra-JEWEL requirements	8d	9/26/94 8:00am	10/5/94 5:00pm	
7	Identify necessary functionality	5d	9/26/94 8:00am	9/30/94 5:00pm	
8	Establish Intra-JEWEL data	4d	9/30/94 8:00am	10/5/94 5:00pm	
9	Write Requirements Analysis document	3d	10/5/94 8:00am	10/8/94 5:00pm	2, 3, 4, 5, 6
10	Analysis Deadline	0d	10/6/94 5:00pm	10/6/94 5:00pm	
11	System Design	15d	10/10/94 8:00am	10/30/94 5:00pm	
12	Select database type	5d	10/10/94 8:00am	10/15/94 5:00pm	
13	Define database structure	5d	10/17/94 8:00am	10/22/94 5:00pm	12
14	Define database management system functions	4d	10/24/94 8:00am	10/27/94 5:00pm	13
15	Document System Design	2d	10/27/94 8:00am	10/30/94 5:00pm	14
16	System Design Deadline	0d	10/26/94 5:00pm	10/26/94 5:00pm	
17	Object Design	8d	10/31/94 8:00am	11/9/94 5:00pm	
18	Research existing DBMS's	2d	10/31/94 8:00am	11/1/94 5:00pm	
19	Select DBMS	0d	11/1/94 5:00pm	11/1/94 5:00pm	18
20	Establish database format	4d	11/1/94 8:00am	11/4/94 5:00pm	19
21	Establish interface with user	4d	11/1/94 8:00am	11/4/94 5:00pm	19, 20
22	Establish interface with other groups	4d	11/1/94 8:00am	11/4/94 5:00pm	20, 21
23	Create Object Model	4d	11/1/94 8:00am	11/4/94 5:00pm	19, 20, 21, 22
24	Document Object Model	3d	11/2/94 8:00am	11/4/94 5:00pm	23
25	Object Design Deadline	0d	11/9/94 5:00pm	11/9/94 5:00pm	
26	Implementation & Unit Testing	18.88d	11/3/94 9:00am	11/29/94 5:00pm	
27	Create database	2d	11/3/94 9:00am	11/7/94 9:00am	
28	Acquire existing data	6d	11/7/94 8:00am	11/14/94 5:00pm	
29	Code for database access	12d	11/7/94 8:00am	11/22/94 5:00pm	27
30	Documentation	4d	11/22/94 8:00am	11/25/94 5:00pm	29
31	Implementation & Unit Testing Deadline	0d	11/29/94 5:00pm	11/29/94 5:00pm	

### Database Gantt Chart:



### Database PERT Chart:

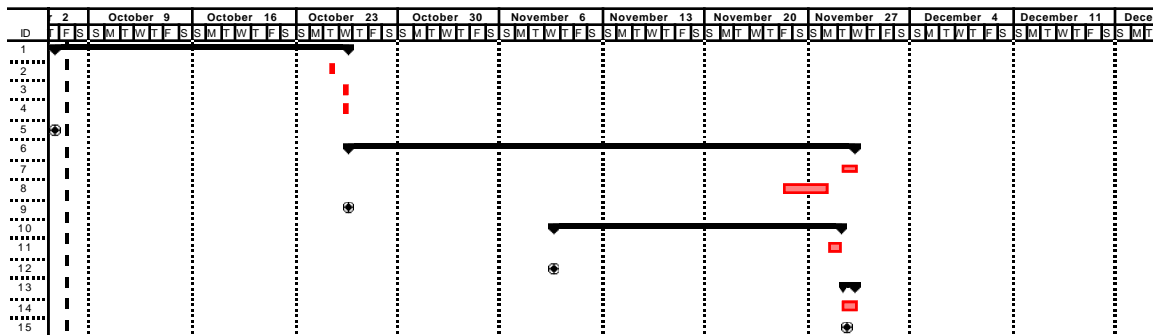


### 4.4.3 Documentation Schedule

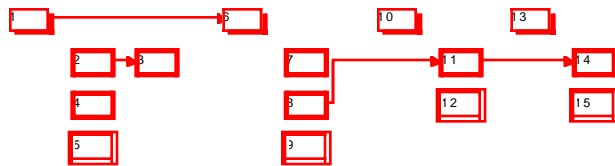
**Documentation Task Sheet:**

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	1 Analysis	14d	10/6/94 5:00pm	10/26/94 5:00pm	
2	1.1 Requirements Analysis	1d	10/25/94 8:00am	10/25/94 5:00pm	
3	1.2 Choose Authoring System	1d	10/26/94 8:00am	10/26/94 5:00pm	2
4	1.3 Specify Code Doc. Standard	1d	10/26/94 8:00am	10/26/94 5:00pm	
5	1.4 Analysis Deadline	0d	10/6/94 5:00pm	10/6/94 5:00pm	
6	2 System Design	24.13d	10/26/94 5:00pm	11/30/94 9:00am	1
7	2.1 Document First Prototype	1d	11/29/94 9:00am	11/30/94 9:00am	
8	2.2 Create Document Outline	1d	11/28/94 8:00am	11/28/94 8:00am	
9	2.3 System Design Deadline	0d	10/26/94 5:00pm	10/26/94 5:00pm	
10	3 Object Design	13.13d	11/9/94 5:00pm	11/29/94 9:00am	
11	3.1 Draft System Documentation	1d	11/28/94 9:00am	11/29/94 9:00am	8
12	3.2 Object Design Deadline	0d	11/9/94 5:00pm	11/9/94 5:00pm	
13	4 Implementation & Unit Testing	1d	11/29/94 9:00am	11/30/94 9:00am	
14	4.1 Complete System Document	1d	11/29/94 9:00am	11/30/94 9:00am	11
15	4.2 Implementation & Unit Testing Deadline	0d	11/29/94 5:00pm	11/29/94 5:00pm	

**Documentation Gantt Chart:**



**Documentation PERT Chart:**



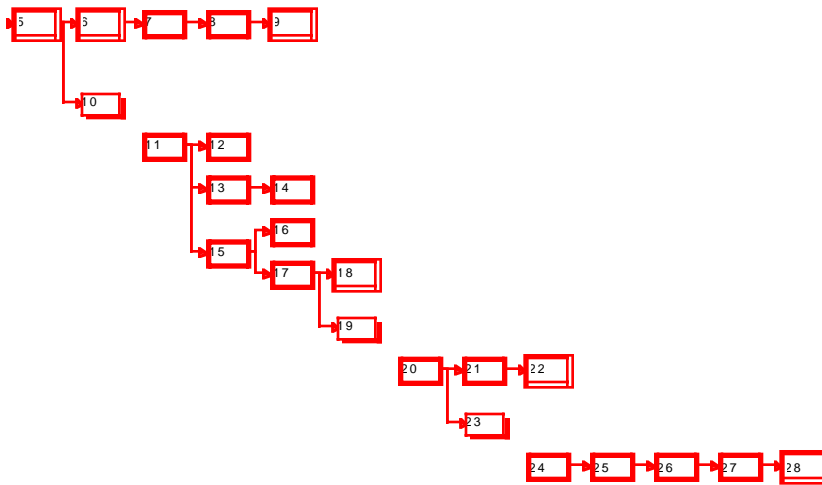
### 4.4.4 Emergency Schedule

**Emergency Task Sheet:**

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	1 Requirements Analysis	5d	Fri 9/16/94	Thu 10/6/94	
2	1.1 Obtain initial client requirements analysis	3.33d	Fri 9/16/94	Thu 9/29/94	
3	1.2 Clarify client requirements	0.33d	Fri 9/30/94	Fri 9/30/94	2
4	1.3 Negotiate with client	0.33d	Fri 9/30/94	Fri 9/30/94	2
5	1.4 Make initial models	0d	Sat 10/1/94	Sat 10/1/94	3,4
6	1.5 Revise models	0d	Sun 10/2/94	Sun 10/2/94	5
7	1.6 Verify consistency of models	0.33d	Mon 10/3/94	Mon 10/3/94	6
8	1.7 Finalize Requirements Analysis Document	1d	Tue 10/4/94	Thu 10/6/94	7
9	1.8 Analysis Deadline	0d	Thu 10/6/94	Thu 10/6/94	8
10	2 System Design	4d	Tue 10/11/94	Wed 10/26/94	9
11	2.1 Determine components	0.33d	Tue 10/11/94	Tue 10/11/94	
12	2.2 Define interrelationships	0.33d	Wed 10/12/94	Wed 10/12/94	11
13	2.3 Find dependencies on other groups	0.33d	Thu 10/13/94	Thu 10/13/94	11
14	2.4 Work with other groups	0.33d	Fri 10/14/94	Fri 10/14/94	13
15	2.5 Pick media and protocols	0.33d	Mon 10/17/94	Mon 10/17/94	11
16	2.6 Revise SPM	0.33d	Wed 10/19/94	Wed 10/19/94	15
17	2.7 Finalize System Design Document	2.33d	Tue 10/18/94	Wed 10/26/94	15
18	2.8 System Design Deadline	0d	Wed 10/26/94	Wed 10/26/94	17
19	3 Object Design	2d	Wed 11/2/94	Wed 11/9/94	17
20	3.1 Define Objects from Object Model	0.67d	Wed 11/2/94	Thu 11/3/94	
21	3.2 Finalize Object Design Document	1.33d	Fri 11/4/94	Wed 11/9/94	20
22	3.3 Object Design Deadline	0d	Wed 11/9/94	Wed 11/9/94	21
23	4 Implementation & Unit Testing	6.33d	Tue 11/29/94	Tue 11/29/94	20
24	4.1 Implement	4d	Thu 11/3/94	Sun 11/20/94	
25	4.2 Write stubs for interface with other groups	1.67d	Mon 11/21/94	Fri 11/25/94	24
26	4.3 Test with stubs	0.33d	Fri 11/25/94	Fri 11/25/94	25
27	4.4 Finalize Testing Document	0.67d	Mon 11/28/94	Tue 11/29/94	26
28	4.5 Implementation & Unit Testing Deadline	0d	Tue 11/29/94	Tue 11/29/94	27

**Emergency Gantt Chart:**

### Emergency PERT Chart:



### 4.4.5 Emission Schedule

**Emission Task Sheet:**

Unavailable at this time

**Emission Gantt Chart:**

Unavailable at this time

**Emission PERT Chart:**

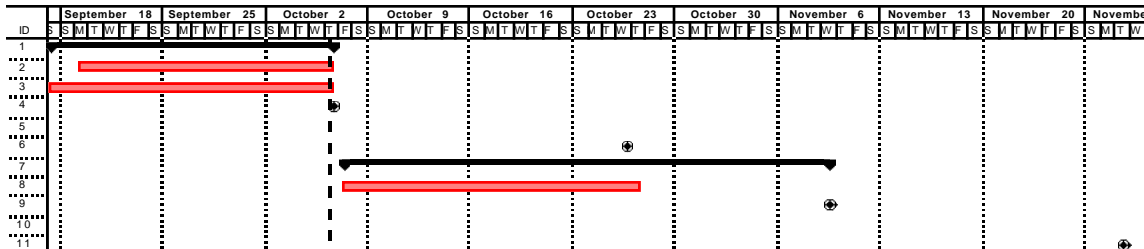
Unavailable at this time

### 4.4.6 GIS Schedule

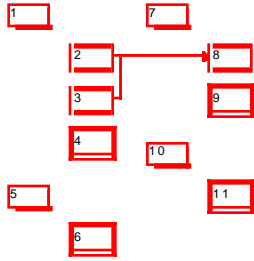
**GIS Task Sheet:**

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	1 Analysis	14d	9/17/94 8:00am	10/6/94 5:00pm	
2	1.1 Interface to Other Groups	14d	9/19/94 8:00am	10/6/94 5:00pm	
3	1.2 Determine Map I/O Format	14d	9/17/94 8:00am	10/6/94 5:00pm	
4	1.3 Analysis Deadline	0d	10/6/94 5:00pm	10/6/94 5:00pm	
5	2 System Design	0d	10/26/94 5:00pm	10/26/94 5:00pm	
6	2.1 System Design Deadline	0d	10/26/94 5:00pm	10/26/94 5:00pm	
7	3 Object Design	24d	10/7/94 8:00am	11/9/94 5:00pm	
8	3.1 Determine/Implement Map Manipulation Al	15d	10/7/94 8:00am	10/27/94 5:00pm	2,3
9	3.2 Object Design Deadline	0d	11/9/94 5:00pm	11/9/94 5:00pm	
10	4 Implementation & Unit Testing	0d	11/29/94 5:00pm	11/29/94 5:00pm	
11	4.1 Implementation & Unit Testing Deadline	0d	11/29/94 5:00pm	11/29/94 5:00pm	

**GIS Gantt Chart:**



### GIS PERT Chart:

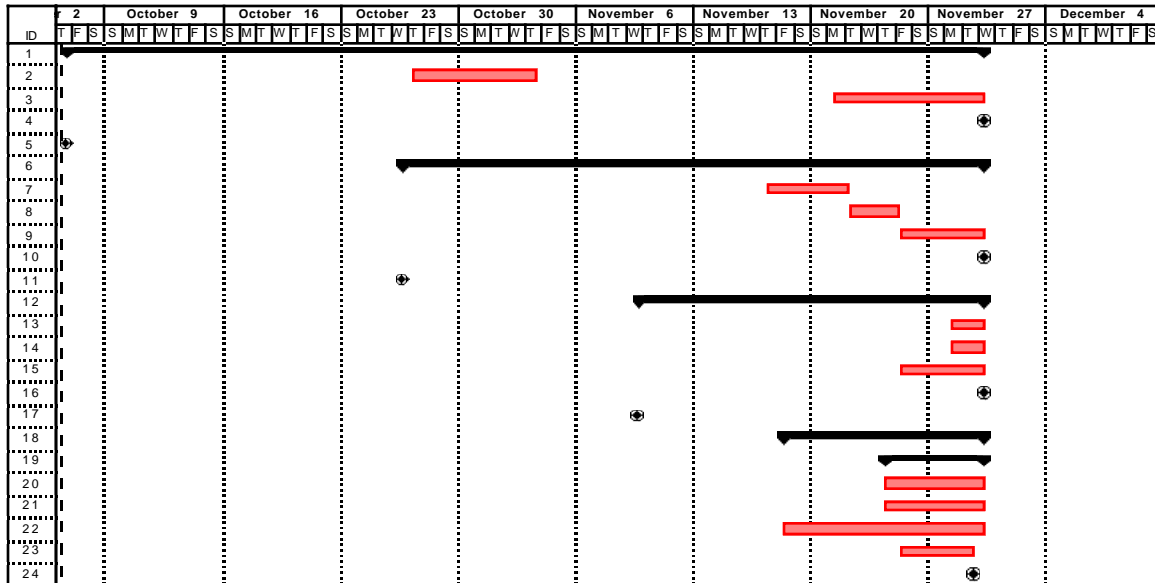


### 4.4.7 UI Schedule

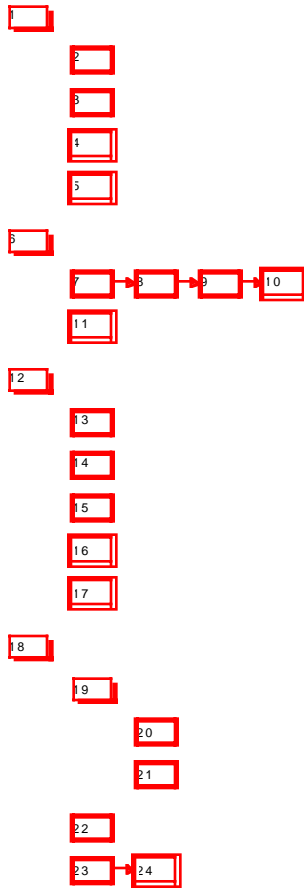
#### UI Task Sheet:

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	1 Analysis	38.13d	10/6/94 5:00pm	11/30/94 9:00am	
2	1.1 Storyboard (napkin) Interface Design	6d	10/27/94 8:00am	11/3/94 5:00pm	
3	1.2 Create preliminary object model design	7d	11/21/94 9:00am	11/30/94 9:00am	
4	1.3 Create Requirements Analysis document	0d	11/30/94 9:00am	11/30/94 9:00am	
5	1.4 Analysis Deadline	0d	10/6/94 5:00pm	10/6/94 5:00pm	
6	2 System Design	24.13d	10/26/94 5:00pm	11/30/94 9:00am	
7	2.1 Make final decision on UI builder and work	3d	11/17/94 9:00am	11/22/94 9:00am	
8	2.2 Establish expertise with the chosen tool	3d	11/22/94 9:00am	11/25/94 9:00am	7
9	2.3 Design intra-platform communications	3d	11/25/94 9:00am	11/30/94 9:00am	8
10	2.4 Document System Design	0d	11/30/94 9:00am	11/30/94 9:00am	9
11	2.5 System Design Deadline	0d	10/26/94 5:00pm	10/26/94 5:00pm	
12	3 Object Design	14.13d	11/9/94 5:00pm	11/30/94 9:00am	
13	3.1 Consolidate/organize UI relation/comm. w/	2d	11/28/94 9:00am	11/30/94 9:00am	
14	3.2 Organize communications/calls into objects	2d	11/28/94 9:00am	11/30/94 9:00am	
15	3.3 Establish a user interface in an object form	3d	11/25/94 9:00am	11/30/94 9:00am	
16	3.4 Create and document the above objects	0d	11/30/94 9:00am	11/30/94 9:00am	
17	3.5 Object Design Deadline	0d	11/9/94 5:00pm	11/9/94 5:00pm	
18	4 Implementation & Unit Testing	8d	11/18/94 9:00am	11/30/94 9:00am	
19	4.1 Construction of the UI	4d	11/24/94 9:00am	11/30/94 9:00am	
20	4.1.1 Build windows and widgets	4d	11/24/94 9:00am	11/30/94 9:00am	
21	4.1.2 Connections and Callbacks between W	4d	11/24/94 9:00am	11/30/94 9:00am	
22	4.2 Implement connections with Vis/GIS	8d	11/18/94 9:00am	11/30/94 9:00am	
23	4.3 Document the implementation	3d	11/25/94 8:00am	11/29/94 5:00pm	
24	4.4 Implementation & Unit Testing Deadline	0d	11/29/94 5:00pm	11/29/94 5:00pm	23

#### UI Gantt Chart:



## UI PERT Chart:

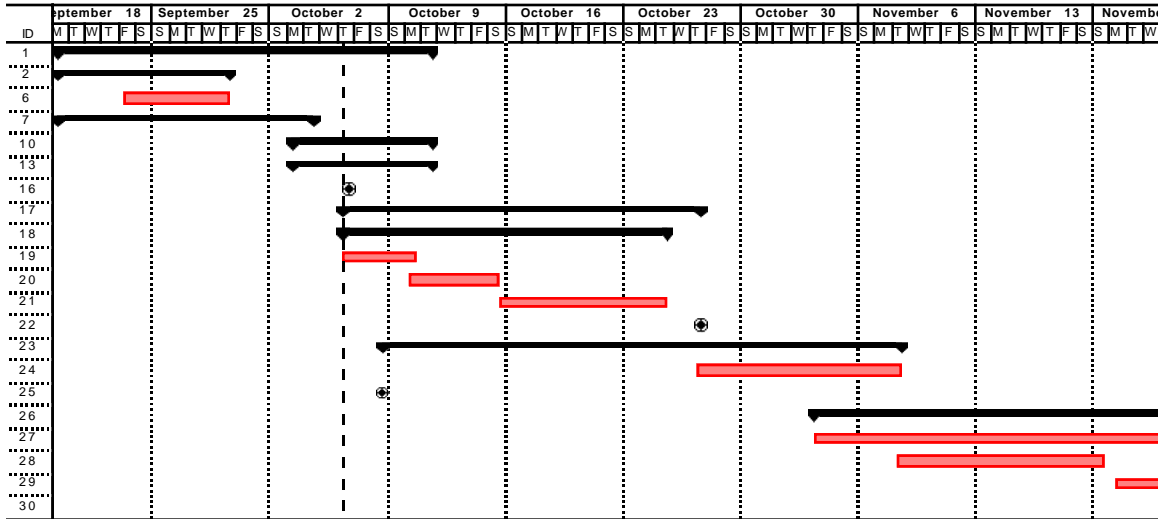


### 4.4.8 Visualization Schedule

#### Visualization Task Sheet:

ID	Name	Duration	Scheduled Start	Scheduled Finish	Predecessors
1	1 Analysis	17d	9/19/94 8:00am	10/11/94 5:00pm	
2	1.1 SBMP	9d	9/19/94 8:00am	9/28/94 5:00pm	
6	1.2 Requirements Analysis	5d	9/23/94 8:00am	9/29/94 5:00pm	
7	1.3 Object Model	12d	9/19/94 8:00am	10/4/94 5:00pm	
10	1.4 Dynamic Model	7d	10/3/94 8:00am	10/11/94 5:00pm	
13	1.5 Functionnal Model	7d	10/3/94 8:00am	10/11/94 5:00pm	
16	1.6 Analysis Deadline	0d	10/6/94 5:00pm	10/6/94 5:00pm	3
17	2 System Design	16d	10/6/94 8:00am	10/21/94 5:00pm	1
18	2.1 devise system architecture	14d	10/6/94 8:00am	10/25/94 5:00pm	
19	2.1.1 temporal plots	3d	10/6/94 8:00am	10/10/94 5:00pm	
20	2.1.2 area source plots	5d	10/10/94 8:00am	10/15/94 5:00pm	
21	2.1.3 subsystem: density plots	7d	10/15/94 5:00pm	10/25/94 5:00pm	20
22	2.2 System Design Deadline	0d	10/27/94 5:00pm	10/27/94 5:00pm	21
23	3 Object Design	22d	10/8/94 5:00pm	11/8/94 5:00pm	17
24	3.1 detail attributes and signatures	9d	10/27/94 8:00am	11/8/94 5:00pm	
25	3.2 Object Design Deadline	0d	10/8/94 5:00pm	10/8/94 5:00pm	24
26	4 Implementation & Unit Testing	19d	11/3/94 8:00am	11/30/94 8:00am	23
27	4.1 implementation	19d	11/3/94 8:00am	11/29/94 5:00pm	
28	4.2 testing	9d	11/8/94 8:00am	11/20/94 5:00pm	
29	4.3 documentation	7d	11/21/94 8:00am	11/29/94 5:00pm	28
30	4.4 Implementation & Unit Testing Deadline	0d	11/30/94 8:00am	11/30/94 8:00am	29

### Visualization Gantt Chart:



### Visualization PERT Chart:

