City of Dotcom Traffic Light Software Project

Table of Contents

Figures

Tables

1. Concept Document

1.1 Introduction

1.1.1 Description

The City of Dotcom is experiencing extraordinary growth. The basis for this concept document requirements is the growth study commissioned by the city planning committee. The city planners saw that their city was about to have a great growth spurt and consequently they commissioned a study. The study indicated a modernization project in three phases. Phase one is to be one year in length and requires the immediate installation of a traffic control system. Phase two requires road improvements in the form of bridges and overpasses, proposed changes in the zoning ordinances, and provisions for growth in communications technology. Phase two was a long term plan involving fifteen years. Phase three called for a rapid transit system patterned on Oakland California's Bay Area Rapid Transit (BART). Both phase two and three are subject to change according to growth predictions. The City Planners decided to implement phase one of the plan. This project is part of the implementation of phase one. Dotcom needs a traffic control system beginning with the bottleneck at their main intersection. Their plan is to create a central control room for the traffic control system. There will be several different algorithms for the light switching determined by a monitoring system for traffic flow. The heart of the system is the traffic sensors which provide data to the control system.

1.1.2 Purpose

The software controls the traffic by monitoring traffic flow and selecting the proper switching algorithm for the traffic light. When the system is mature it will directly control the actions of the traffic light at all times.

1.1.3 Scope

This concept document encompasses all software requirements under phase one of the prime contract. The purpose of the software is to control the traffic light in such a way as to prevent gridlock. It is being designed for use at one particular intersection, a four-way stoplight with both streets having two lanes going each direction. Though the product is intended for only one stoplight, it will have the capability of being expanded for use at multiple stoplights.

1.1.4 Objectives

The objective is for the final product to provide for economic growth by relieving traffic congestion as effectively and efficiently as possible.

1.1.5 Users

The software is used by a group of traffic controllers in a control room. Though most of the time, the software will regulate the traffic light changes on its own, these users must be able to override the software's decisions if necessary and document problems. The users are to be educated to the grade 14 level. The supervisor will be educated to the grade 16 level.

1.1.6 Capabilities

The software will be able to monitor the flow of traffic at the stoplight. It will read in data concerning the traffic flow, and based on a set of conditions within the software, and it will change the light when appropriate. Also, the software will detect the temperature in the pavement for icing conditions.

1.1.7 Assumptions

a) The software is being used on a four way stop with both streets having two lanes going each direction.
b) The sensors used to monitor the traffic flow, and the lights themselves can malfunction.
c) There will be a maturation process requiring the system to be monitored on a decreasing basis.
d) The prime contractor is providing all required hardware including traffic sensors, traffic lights, trunk lines, cabling, off the shelf software and computers.

1.1.8 Planned Future Extensions

Later the software may be used to control more traffic lights. Future extensions could include a computer operated mass transit system of electric trains.

1.2. Non-functional Characteristics

1.2.1 Performance:

The traffic controller system will provide sensor and error detection in a timely manner. Public safety is the highest consideration. Public trust in the system controlling the traffic lights in the City of Dotcom is the next consideration.

1.2.2 Interfaces:

There will be three workstations. Each workstation will use a touch screen interface for the controller operators. Workstations will interface the operator with different subsystems as follows:

a) Control System- The traffic light will be operated to optimize traffic flow. Decisions will be made based on real-time data from the traffic sensors on which lights should be green, red or yellow.
b) Signal Analyzer System- Analysis of erroneous control signals. If there are no dangerous conditions detected, the signal analyzer will allow the Control System to operate. If errors are detected the signal analyzer system will command a failsafe algorithm to run the lights until it is manually reset. It also sets off an alarm to alert the operator
c) Comparison Interface System- This system will detect hardware failures. It will look for physical defects with the lamps and sensors. The comparison Interface System is directly related to public safety.

1.2.3 Reliability

A public safety quality assurance administrator will monitor the performance of the three systems. All interactive services will be connected to an error detection database to track occurrences of unsatisfactory conditions. The benchmark for the reliability metric will be based on the number of recorded instances of errors caused by sensor malfunction (bad data), logic errors within the applications themselves, and actual human observation of traffic patterns.

1.2.4 Quality

The quality assurance evaluator who oversees system reliability for the City of Dotcom will use the metrics gathered for the reliability function to compute the level of software quality. The quality of the hardware will be guaranteed by the manufacturer since it is off the shelf equipment.

1.2.5 Security

Security will include a login id to be assigned to each certified controller. Only certified controllers will have this login id and every three months it will be changed. The data communications equipment will be separate from the internet and hardened against lightning strikes or other catastrophes.

1.2.6 Maintainability

A software administrator will be on pager support for any issues that are documented. System maintenance will include a monthly status check on all displays and their related functions. Also an automated daily self-test will be run to check all redundant paths for reliable back-up. All the cabling to the sensors will be run through conduit for easy access and inspection.

1.3. Software Environment

A Unix platform will be used for development and operational use of the system. The code will be written in assembly language to take advantage of the speed. GUI displays will be developed from within the code for the operators to have defined parameter readings.

1.4. Operational Scenarios

It is a Sunday in the city of Dotcom. A resident of the city wants to go to church that morning. He drives along the highway when he sees an intersection ahead. The light is red for his direction of traffic. Because the light is red, he knows to stop. He waits with other vehicles for the other directions of traffic to individually, on their turn with a standard timer, to have traffic lamps change from green to yellow to red. This occurs. His light now changes to green from red and his direction of traffic proceeds, while the other directions wait for their timed turn. The Traffic Control System senses the traffic with a series of magnetic sensors placed in the pavement every 200 feet in all non-residential streets. The sensors detect a vehicle passing over and send a signal to a node, which transmits traffic information back to the Control Center every 20 seconds. The main control display depicts the city streets in green, red, and yellow. Green signifying normal flow, red signifying slowed or stopped traffic, and yellow is in between.

2. <u>Project Plan</u>

2.1 Introduction

The Traffic light software project for Dotcom is limited to a budget of $xxx,xxx. This includes all software development from the beginning through acceptance testing. Computer and other hardware is being made available by the prime contractor. The software is to be completed one year from the award of the contract.

2.2 Project Organization

The development team is made up of 4 members: two programmers and a data communications specialist. One manager is also required to co-ordinate with the prime contractor, and the city planners.

2.3 Risk Analysis

The possible risks to the project are being broken down into two parts: those within our span of control and external factors. To help control these risks the users will have the ability to override the software and control the stoplight manually. The Traffic light being acquired will have a fail-safe mode consisting of a four way stop signal

2.3.1 Internal Factors

a) Operating system reliability.
b) Satisfaction of the operator with the computer interfaces.
c) Timely completion of development milestones
d) Catastrophic traffic events caused by erroneous signals from the traffic light.

2.3.2 External factors

a) Power outages
b) Traffic accidents (not caused by erroneous signals)
c) Natural disasters and acts of God.
d) Lamp burnout

2.4 Hardware/Software Resources Requirements

All necessary hardware and software needs are being provided by the prime contractor. Hardware requirements include all the mechanical components of the stoplight and a computer system running the UNIX operating System. The main software requirement is an assembler with which to write the software.

3. Project Schedule

3.1 Task Duration and Dependencies

Task	Description	Duration in Work Days	Dependant Upon
T1	System Hardware Requirements	5	None
T2	Acquisition of Light Hardware	30	T1
T3	Acquisition of Sensor Hardware	30	T1
T4	Acquisition of Data Communications Hardware	30	T1
T5	Acquisition of Control Facility Hardware	30	T1
T6	Design of Control Facility Layout	15	T1
T7	Habitation of Control Facility	50	None
T8	Laying Cable	15	T2, T3, T4
T9	Traffic Light Hardware Installation	1	T2
T10	Traffic Sensor System Installation	3	T3
T11	Test Light and Sensors After Installation	5	T2, T3, T4, T8
T12	Produce Software Development Plan, and User Manual Development Plan	15	T1
T13	Software Requirements Verification and 35% review of User Manual	10	T12
T14	Software Design Verification and 75% review of User Manual	10	T13
T15	Produce Code	50	T14
T16	Code Verification	10	T15
T17	Facility Equipment Installation	10	T5, T7
T18	Facility Software Installation	10	T15, T16, T17
T19	Software Validation and Final Review of User Manual	10	T18
T20	System Test	5	T19
T21	First Generation Software/User Manual Acceptance Test	2	T20
T22	Program Review	1	T21

3.2 Allocation of People to Tasks

Mary – Software Engineering Manager. She will take part in T1 with the prime contractor. Next she and the others will begin work on T12, T13, and T14. After completion of T14, Mary and Fred will begin work on the User Manual portion of T19. After completion of T19 Mary will be present through the final review.

Fred – Data Communications Engineer. Fred will begin on T12 while consulting with the prime contractor on matters involving Data Communication. While Bill and Renee are writing code, Fred will be working with Mary on the User Manual. After the completion of T16 he will stop working on the User Manual and assist Bill and Renee with the software installation.

Renee and Bill – Computer Programmers. The programmers will work on each task in order: T12, T13, T14, T15, T16. Then they will perform T18, After which time they will be around to perform minor support until completion of the project.

Task	Engineer
T1	Prime Contractor and Mary
T2	Prime Contractor
T3	Prime Contractor and Fred as a Consultant
T4	Prime Contractor and Fred as a Consultant
T5	Prime Contractor
T6	Prime Contractor
T7	Prime Contractor
T8	Prime Contractor
T9	Prime Contractor
T10	Prime Contractor
T11	Prime Contractor
T12	Mary, Fred, Bill, Renee
T13	Mary, Fred, Bill, Renee
T14	Mary, Fred, Bill, Renee
T15	Bill, Renee
T16	Bill, Renee
T17	Prime Contractor
T18	Bill, Renee, Fred
T19	Mary and Prime Contractor
T20	Prime Contractor and Mary
T21	Prime Contractor and Mary
T22	Prime Contractor and Mary

3.3 Activity Network and Milestone Description

3.3.1 Activity Network

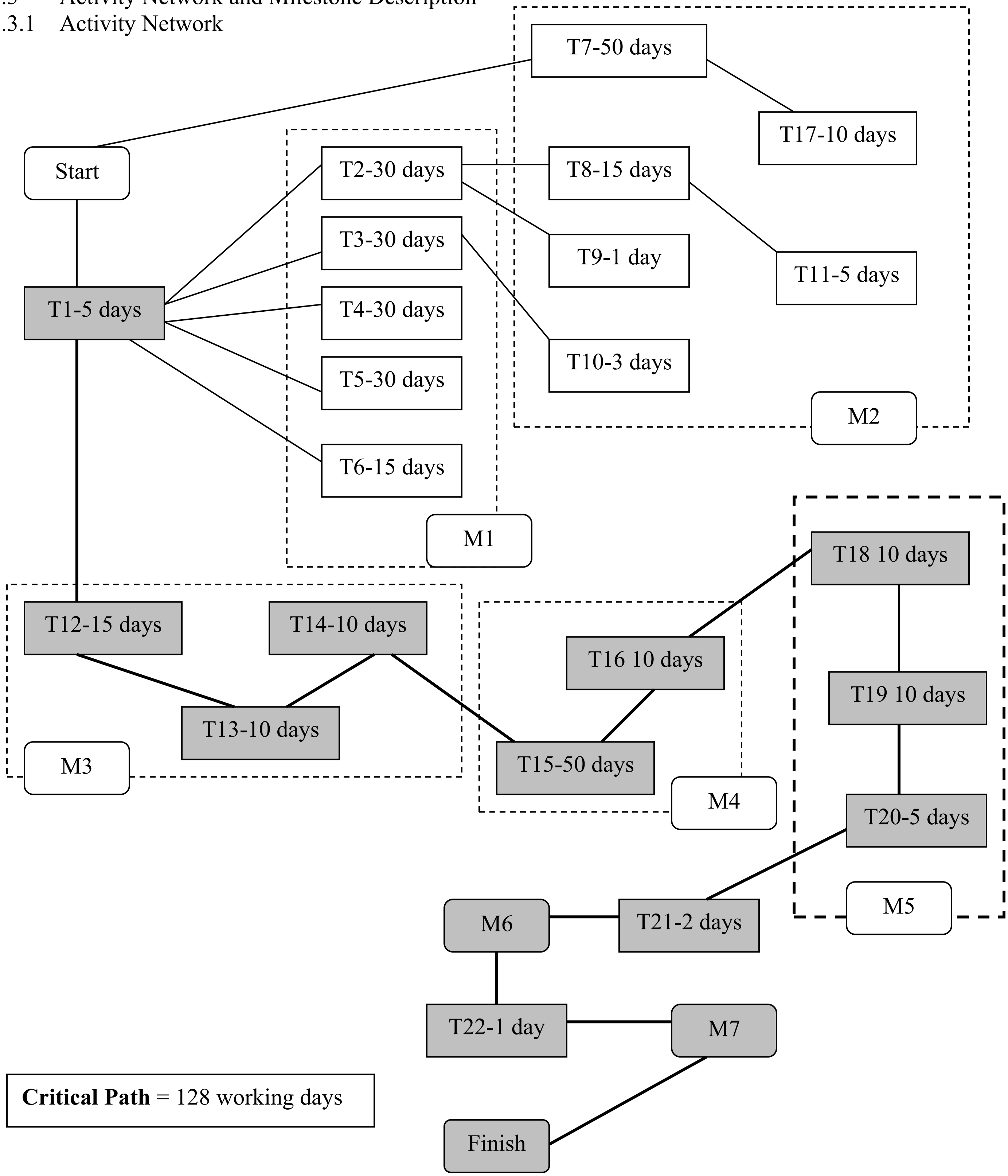


Figure 3-1 Critical Path Analysis

3.3.2 Milestone Description. Each milestone consists of a program review, and a management signoff for each task. Work-arounds must be approved by the city planning engineer. After the milestone is signed off the go ahead is given to continue to the next milestone with the following exception. Milestone M1 and Milestone M2 start concurrently after completion of task T1. Milestone M2 must be complete prior to beginning work on Milestone M5. Payments are made by the city of Dotcom with each milestone according to the contract.

3.3.2.1 Milestone M1 signifies the completion of tasks T1, T2, T3, T4, T5, and T6. All of the system hardware will be in hand when the facility is ready for habitation.

3.3.2.2 Milestone M2 signifies the completion of task T7, T8, T9, T10, T11, and T17. Task 7 is a special case. The building to be used has been selected and is owned by the city but will not be available until ten weeks after the start of the project. The start of the project is calculated with respect to the availability of the facility. Laying of cables, installation of the light system, installation of sensors, and operational checkout will all be completed at the end of Milestone M2. (The system being procured will work without software, however software is required for the special application required by the city).

3.3.2.3 Milestone M3 signifies the completion of tasks T12, T13, and T14. These tasks will begin when after T1. The hardware specifications will be used to write the software and the user manual. The completion certificates for the 35% and 75% in progress reviews of the user manual are required for this milestone.

3.3.2.4 Milestone M4 signifies the completion of task T15, and T16. Fully validated code will be delivered for initial installation into the facility.

3.3.2.5 Milestone M5 signifies the completion of task T18, T19, and T20. When milestone M5 is achieved, all labor-intensive work will be completed. Fully functional software and a users manual will be delivered at the completion of the testing.

3.3.2.6 Milestone M6 signifies the completion of task T21, the First Generation Software Acceptance Test. Task T21 will also serve as the acceptance test for the User Manual being produced, and delivered.

3.3.2.7 Milestone M7 signifies the completion of Task T22, the program review and final sign off of completion of the contract.

3.4 Activity Bar Chart

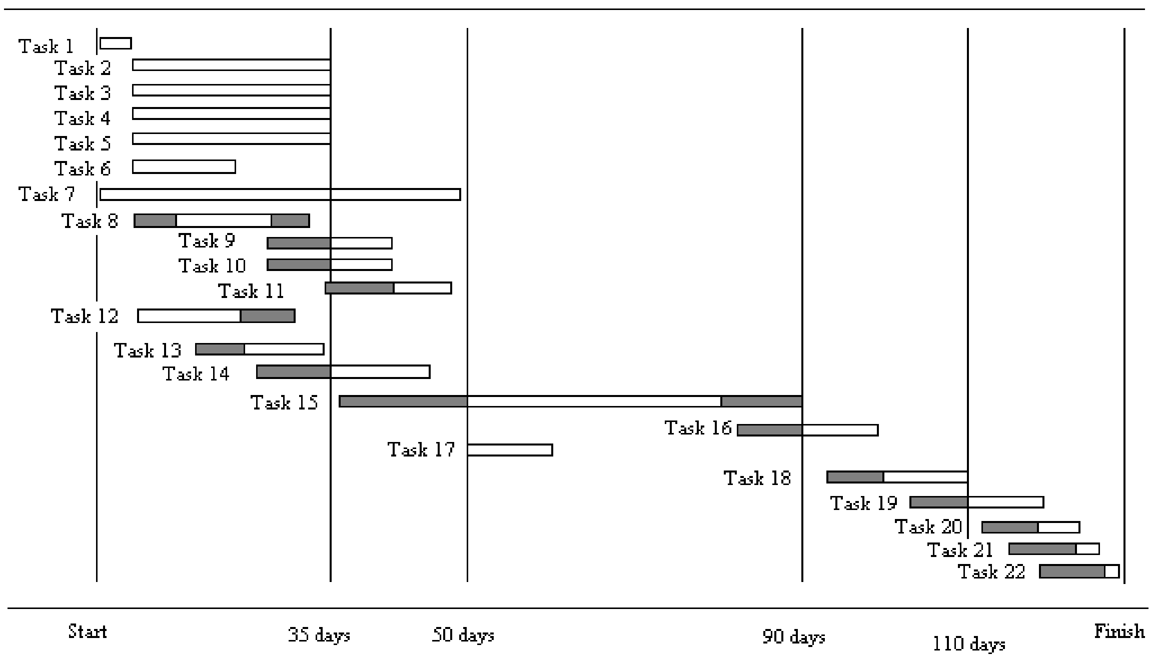


Figure 3-2 Activity Bar Chart

3.5 Staff Allocation verses Time

Although this chart shows who is working on what and when, the entire team is working together on all the non-management items. Management items such as T1, the milestone signoff's and T22 are Mary's responsibility.

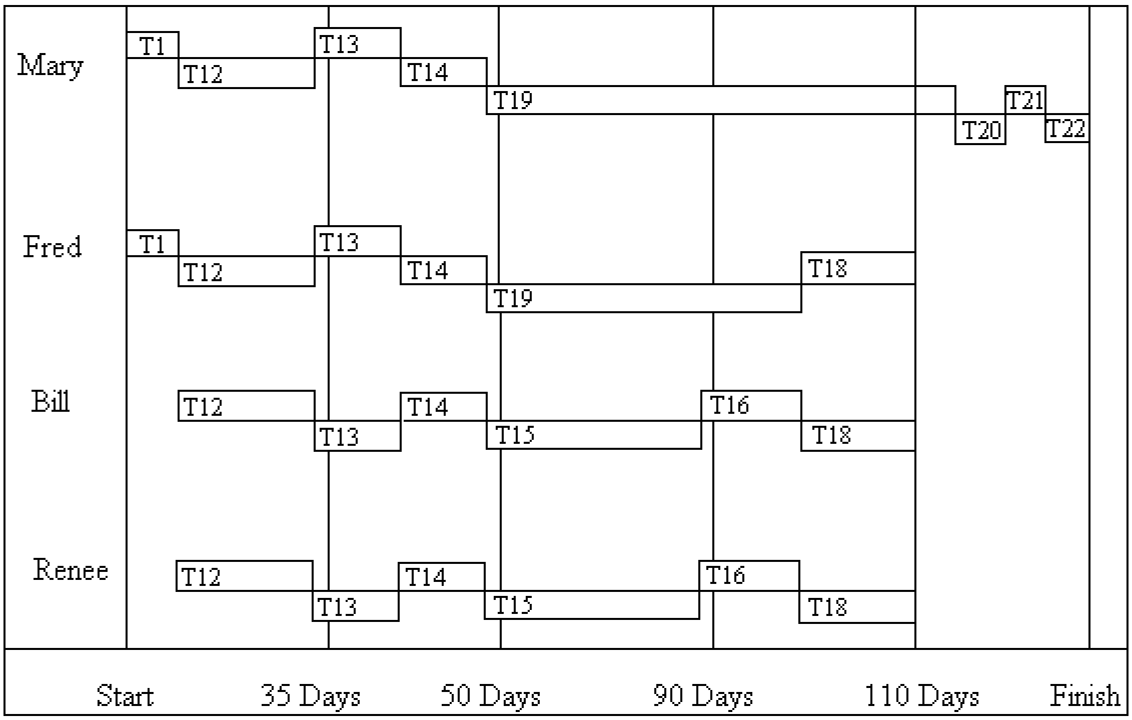


Figure 3-3 Staff Allocation vs. Time Chart

4. Requirements Analysis

4.1 Information Resources

4.1.1 Producing requirements analysis documents is a significant step in the design of a software product. The link below discusses the topic of requirements analysis in general:
http://www.cc.gatech.edu/classes/RWL/Process/Requirements_Analysis.html

The above source lists the objectives of requirements analysis as to analyze requirements data, produce a software requirements specifications document, and to resolve different perspectives on requirements. The source emphasizes the importance of addressing the numerous, and possibly conflicting, perspectives concerning the software. Through the requirements analysis activity, various viewpoints are discovered and resolved into a single, easily understandable set of requirements.

This requirements analysis document for the Dotcom traffic light provides five different types of system models demonstrating how the system works. Though they all describe the same system, each one contains unique information not contained in the others.

4.1.2 Section 4.6 of this Requirements Analysis contains a Stimulus Response model of the Dotcom traffic light. The following link provides a thorough description of what a Stimulus Response model is:
http://www.cs.ubc.ca/formalWARE/abstracts/tr97-001.pdf
These models are typically used for software intensive systems with complex data requirements. Their purpose is to show how the software will respond to certain stimuli and what outputs these responses will produce.

The Stimulus Response model is used for providing an overview of the capabilities of the system as seen from a user's perspective. It is a document that could be presented to the customer to clearly communicate what the final product should do. It describes externally visible stimuli and responses, but does not explain the internal design details. This is beneficial in that it simplifies maintenance of the specification and it is not restrictive to the design of the software.

This model has been chosen for use in the Dotcom traffic light project because the software involved will be largely dependent upon external events. It must be able to respond to stimuli such as cars triggering the road sensors, the functional conditions of the light and other associated hardware, or the event in which the user logs in to override the software. Since inappropriate responses to these stimuli could result in disaster, this model is especially valuable to the project.

4.2 Inheritance Model

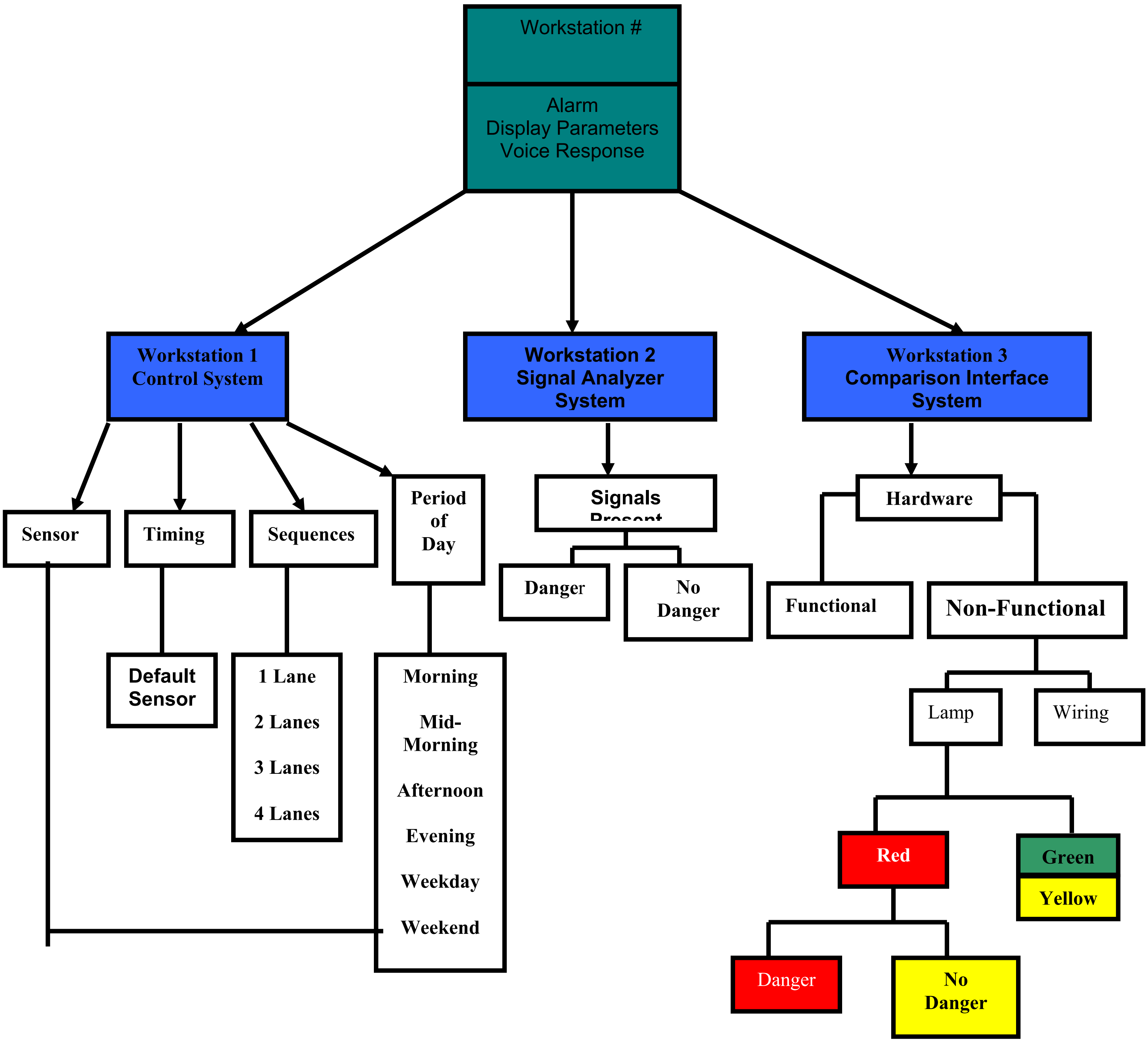


Figure 4-1 Inheritance Model

4.2.1 Data Dictionary for Inheritance Model (Figure 4-1)

Name	Description	Type
Activate_timing	Set timing to defualt or controlled by sensor information	Function
Alarm_sound	A defined audio sound that helps to define the alarm status	Sensor
Alarm_status	Distinguishes level of signal emergency	Attribute
Color	Color active	Sensor
Comparison Interface System	Type of system control	Attribute
Control System	Type of system control	Attribute
Hardware	Describes lamp functionality	Attribute
Light_id	ID number per traffic light	Attribute
Log_in_id	Controller id log in	Attribute
Maintenance	Technician will be called when dangerous signal is detected	Function
Period of Day	Describes time of day to determine level of timing	Attribute
Re_route	Traffic condition for lights to go to a flashing red state	Sensor
Re_route_clear	Take red flashers to normal	Function
Sensor	How many lanes are active along with the frequency of motion	Sensor
Sensor_pad	Weight/motion traffic action	Sensor
Sequence	How many lanes are active with traffic and the time of day.	Sensor
Signal Analyzer System	Type of system control	Attribute
Signal_present	Is there a present signal that is dangerous?	Sensor
System	Which system is in control	Sensor
Timing	Description of how the timing is set; default or sensor	Attribute

4.3 Structural/Composition – Part of Model

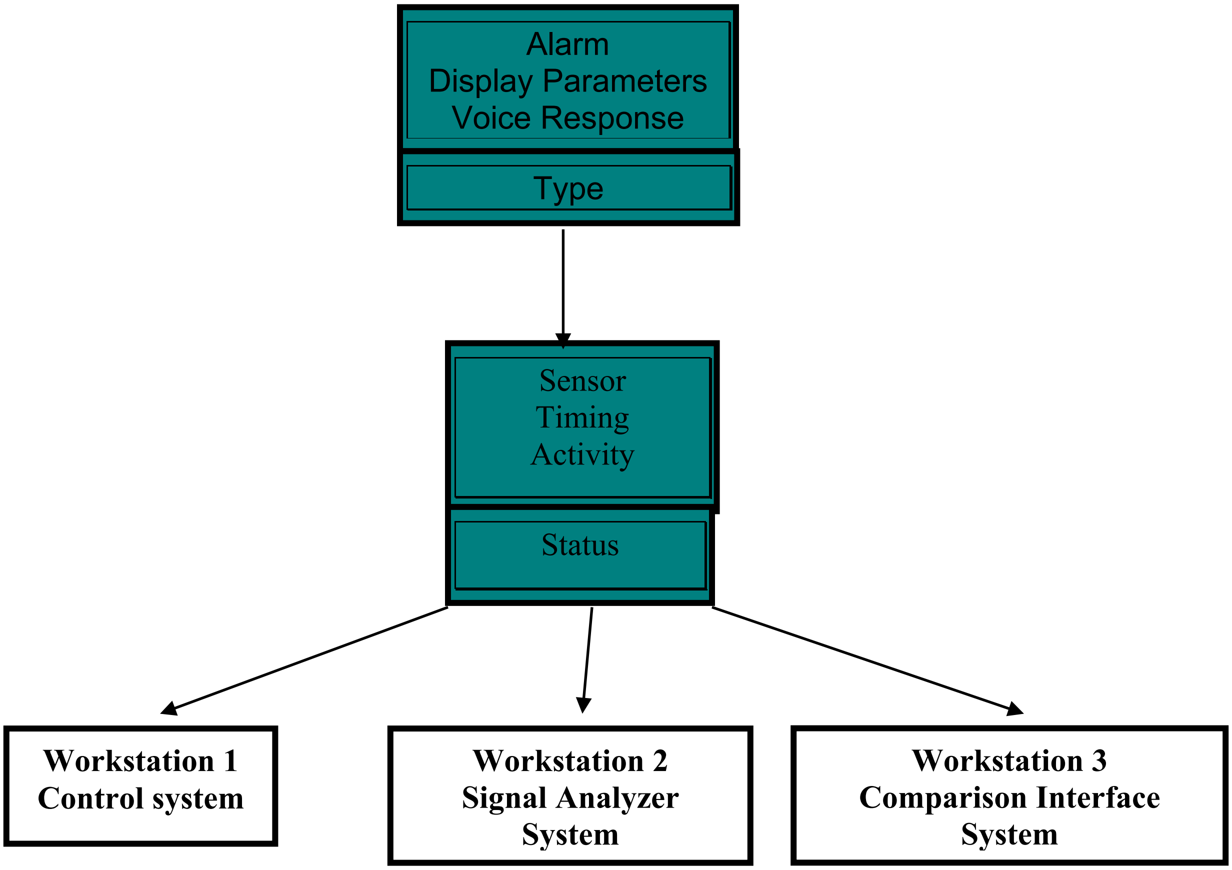
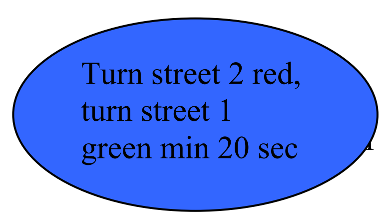


Figure 4-2 Structural Composition Model

4.3.1 Structural/Composition Data Dictionary

Name	Description	Type
Active Lanes	Number of lanes with movement in traffic	Sensor
Color	Color of lamp in use	Sensor
Comparison Interface System	System to monitor for hardware malfunctions	System
Control System	System monitoring sensors and timing of traffic	System
Direction	Direction of traffic active	Sensor
Signal Analyzer System	System to monitor for dangerous signals from control system	System
System	Type of system	Attribute
Time	Time of day	Sensor

4.4 Data Flow Model



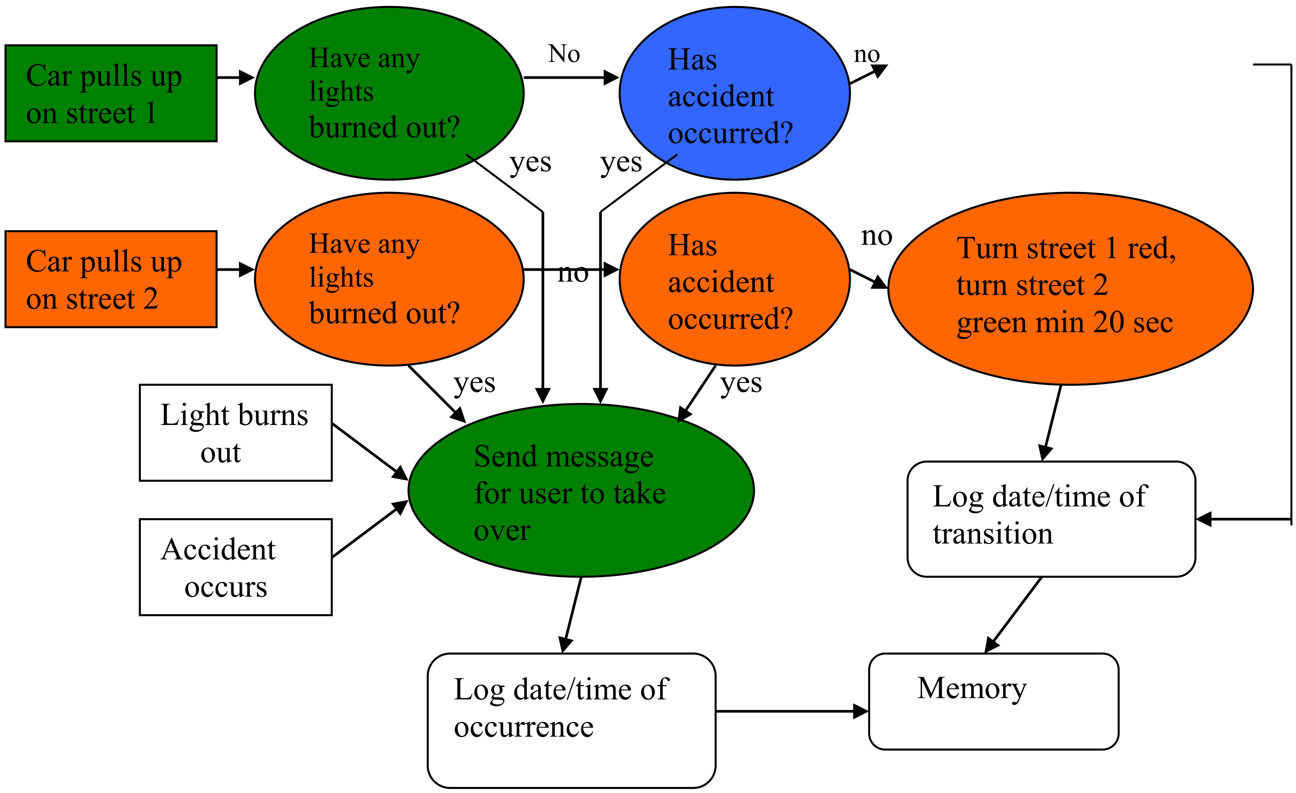


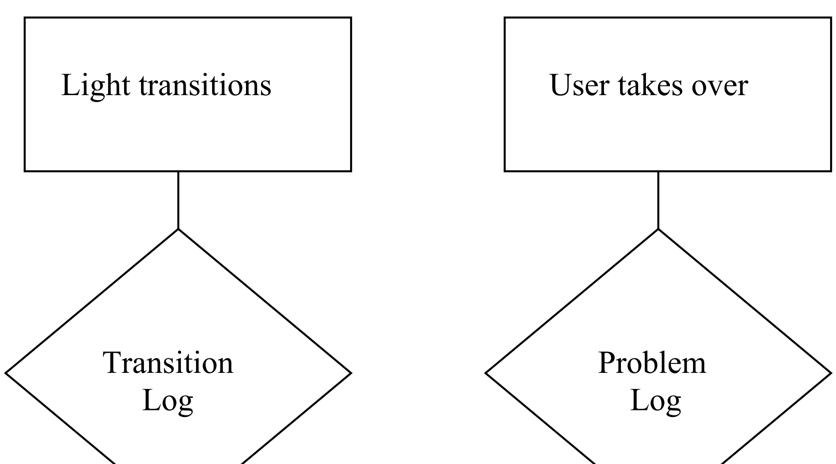
Figure 4-4 Data Flow Model

Data Dictionary for Data Flow Model (Figure 4-4)

Name	Description	Type
Accident occurs	An accident has been reported at the	Input

	intersection	
Car pulls up on street 1	Car activates sensor on street running north/south at intersection	Input
Car pulls up on street 2	Car activates sensor on street running east/west at intersection	Input
Lights burns out	One or more of the lights at the intersection burned out or ceased functioning.	Input
Log occurrence	All accidents and light malfunctions will be logged in records	Output
Log transition	All light transitions will be logged in records.	Output
Memory	Computer's main memory.	Output
Send message for user to take over	Display a message on the user's console to override regular functioning of software	Output
Turn street # green Min 20 sec	Light should turn green and stay green for a minimum of 20 seconds	Output

4.5 Semantic Model



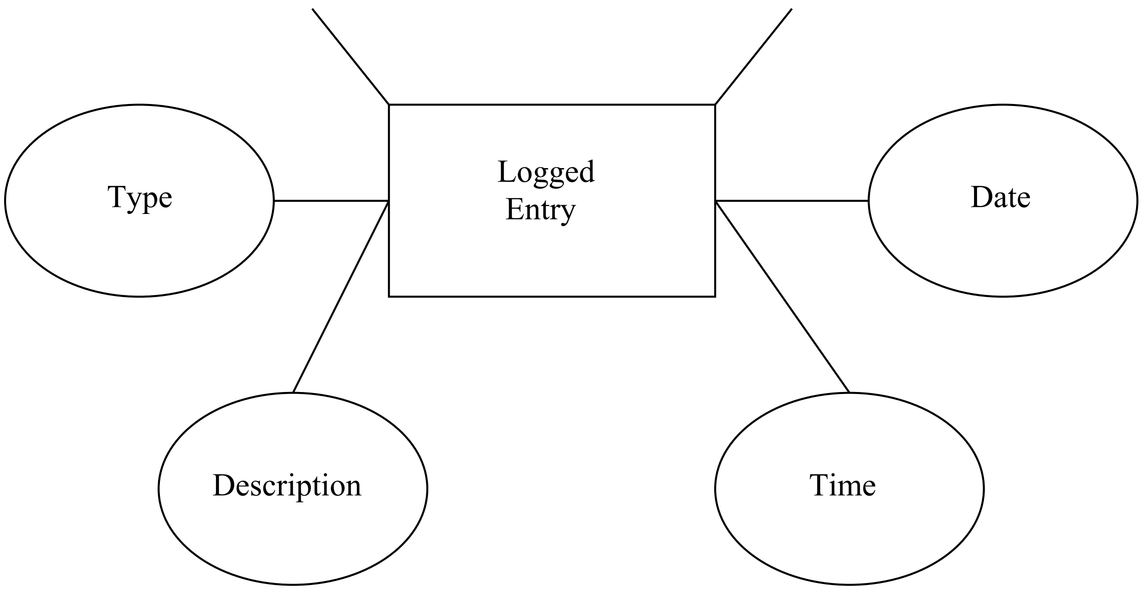


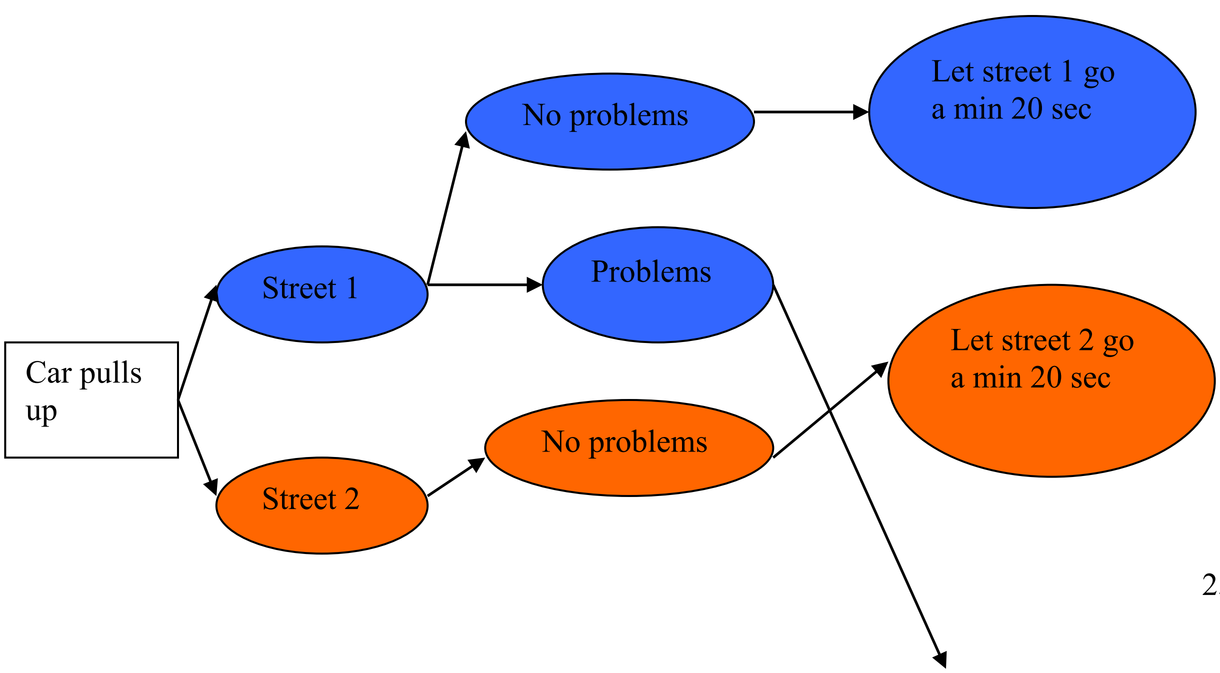
Figure 4-4. Semantic Model

Data Dictionary for Semantic Model (Figure 4-5)

Name	Description	Type
Date	Date of occurrence	Attribute
Description	Brief description of the occurrence. Of use for accident logs.	Attribute
Light transitions	Traffic light changes from being green on one street to green on the other	Entity

	street	
Logged entry	The data entry object for logged data.	Entity
Problem log	Record of logged entries for light malfunctions or accidents.	Entities
Time	Time of occurrence	Attribute
Transition log	Record of logged entries for light transitions	Entities
Type	The type of occurrence being logged (transition, accident, etc)	Attribute
User takes over	User overrides software's usual functioning.	Entity

4.7 Stimulus Response Model



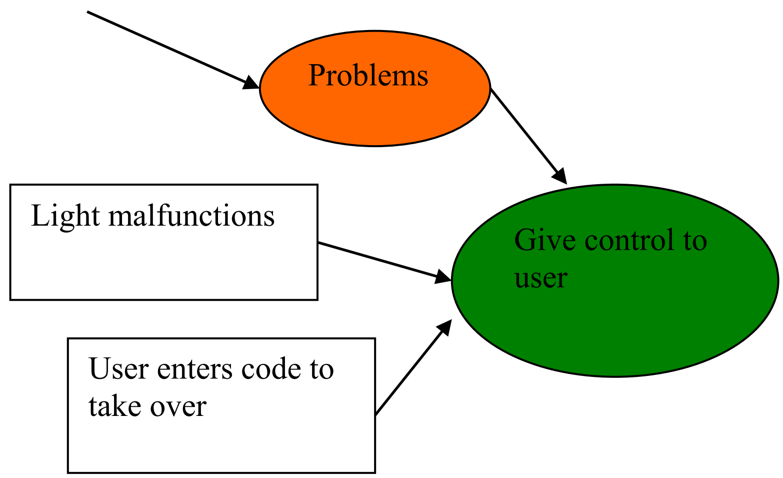


Figure 4-5. Stimulus Response Model

Data Dictionary for Stimulus Response Model (Figure 4-5)

Name	Description	Type
Car pulls up	Car activates one of the sensors at the intersection	Stimulus
Give control to user	The software's usual functioning is ceased and user can control the light from computer	Response
Let street 1 go Min 20 sec	Turn the light for street 2 to red and the light for street 1 to green for a minimum of 20 seconds	Response
Let street 2 go	Turn the light for street 1	Response

Min 20 sec	to red and the light for street 2 to green for a minimum of 20 seconds	
Light malfunctions	One or more of the lights at the intersection burned out or ceased functioning.	Stimulus
No problems	No accidents reported or light malfunctions detected	Response
Problems	Accident reported or light malfunction detected	Response
Street 1	Sensor activated was from the street going north/south at the intersection	Response
Street 2	Sensor activated was from the street going east/west at the intersection	Response
User enters code to take over	User overrides software's usual functioning from console.	Stimulus

5. <u>Requirements Definition</u>

5.1 Requirements Definition Introduction:

The Datacom Traffic Light Control System uses input from magnetic sensors, time of day, day of week, hardware, and flow frequency parameters to determine traffic light delay, switching, timing, and hazardous or emergency situations. The system will respond to these parameters that can be viewed through the controllers workstation displays.

5.2 Overall System Non Functional Requirements.

It shall be discernable, for vehicle operators and pedestrians, which light is illuminated, regardless of the position of the sun. The lights shall be sequenced so that no one has to wait more than 90 seconds for passage through the intersection.

A Counterclockwise round robin control of time will occur. The amount of time for each lane is determined through the number of active lanes.

<u>All cases described below can be simulated for verification / inspection checks.</u>

A. **0 active lanes:** If there are zero active lanes, the round robin control of time will occur through each direction of traffic for 3 minutes each in counterclockwise manner.

B. **0 active lanes become 1**: If a car pulls up to the light and no lanes are active, that lane is now determined active. Flow for that lane will be active for at least 90 seconds(lamp is green). After the 90 seconds, flow will continue until another lanes' sensor is pressed or 3 minutes has occurred. At that moment when one of the parameters occurs, the round robin control of flow will start over again, counterclockwise, with the next sensored direction of flowed traffic becoming active.
C. **1, 2, 3 active lanes**: If a car pulls up to the light and only one or two other lanes are triggered, the control will continue to be round robin, with the non-triggered lane skipped over.
D. **4 active lanes**: When all 4 lanes of traffic are triggered, control will be counterclockwise for 60 seconds first to the turn lane if triggered (where the intersection has that built in), then 90 seconds to the lane with the turn lane. Each direction will have 1-60 second(turn lane) and 1-90 second opportunity of flow, on their turn.
E. The Control System will scan each sensor every 20 seconds to ensure accuracy and frequency of active lanes.
F. Technician will physically maintain inspections of the hardware lamps every month to ensure lamp functionality and if a failure occurs between months, the technician can verify date of burnout to correspond to operator's workstation readout and log entry.
G. If there is a call-in emergency or hazardous situation, the light system reacts exactly to the operator's entry into the proper system of control.

5.3. Functional Requirements

5.3.1 (*Requirement 1.*) **Operation of a single four-way stoplight including a left turn arrow**

<Inspection Checks A-D of Requirement Definition>

5.3.1.1 The software shall provide the logic to operate one intersection which contains a four-way stoplight with/without a left turn arrow (See paragraph 1.1.1 of Concept Document).

Rationale: Computer driven traffic light system is the foundation of the city plan for expansion. The goal of city government is seen as promoting commerce and growth. One of the cornerstones of this concept is efficient transportation.

Non Functional Requirement: The software must be able to carry a design for a turn lane.

< See Table 6.1.1 of Requirement Specification>

5.3.1.2 The logic will provide for optimum traffic flow as a function of: time of day; day of the week; pavement temperature and traffic flow sensors (see paragraph 1.1.6 of Concept Document). Refer to the traffic study commissioned by the city planners for the traffic flow data.

<Inspection Check E. of Requirements Definition>

Rationale: The major demand placed on a traffic control system is the dynamic patterns of traffic flow. On Monday through Friday different demands are placed on the system corresponding to peak commuting hours.

Non Functional Requirement: Magnetic sensors must be functional.

< See Table 6.1.2 of Requirement Specification>

5.3.1.3 An Object Oriented approach will provide encapsulation required to make upgrades easier (see paragraph 1.1.3 of Concept Document). Each new traffic light added to the system will require only to be identified to the system. It will be assigned a grid location and will fit into the traffic flow model without rewriting code. The only code that would need to be written is a specific situation that has not come up before. <u>Verification of this requirement and its sub-requirements will be provided through simulations.</u>

Rationale: When adding a new intersection to the system the same basic algorithms will be used which provides for inheritance. Specific challenges addressed at each intersection will be written in reusable code allowing algorithms to be used in a similar situation later on.

Non Functional Requirement: The upgrade software must be easy for an administrator to install.

< See Table 6.1.3 of Requirement Specification>

5.3.1.4 The basic data structures provided for the single light system will allow for additional lights to be added without unwarranted redesign (see paragraph 1.1.4 of Concept Document). The main data structure used takes into account the four directions normally covered, and left turn arrows for each direction. If a particular intersection has no left turn arrow, there will not be a value passed.

Rationale: When the software is designed, the data structures created will contain all the data required for similar situations. A data structure used for a four-lane road intersecting with another four lane road will be different from a data structure used with a four lane road intersecting with a two lane road.

Non Functional Requirement: None

< See Table 6.1.3 of Requirement Specification>

5.3.1.5 All algorithms shall be based on the traffic study provided by the city planners. Traffic flow patterns are described with regards to time of day, and day of the week. The only other variable to take into account is the actual traffic flow, which will be provided by the sensors. All the target values for traffic flow are contained in the report. The software is to maintain a margin of error of 10%. That is to say if the predictions of the report turn out to be too conservative, the system is to be able to handle an increase of 10% over all predictions.

Rationale: The conclusions of the study are viewed as the standard model for all future plans.

Non-Functional Requirements:

a. The lights shall be sequenced so that no one has to wait more than 90 seconds for passage through the intersection.
b. It shall be discernable, for vehicle operators and pedestrians, which light is illuminated, regardless of the position of the sun.

< See Table 6.1.1 – 6.1.3 of Requirements Specification >

5.3.2 (*Requirement 2*) **Fifty stop/go lights to operate together throughout the city.**

5.3.2.1 The inheritance of software shall provide for up to fifty stoplights to operate in parallel throughout the city (see paragraph 1.1.2 of Concept Document).

Rationale: The standard being applied to the city for future growth cites fifty intersections where traffic problems will occur. They are given an order of precedence and this project is the first of the lights. Being the first we are also building the backbone for the rest of the system. This requirement and its sub-requirement will also be verified by simulation.

Non Functional Requirement: The city will eventually expand to many intersections.

< See Table 6.1.4 of Requirement Specification>

5.3.2.2 Traffic flow sensors will be installed throughout the city on all the thoroughfares identified by the city (see paragraph 1.1.3 of Concept Document).

The system will collect real-time data on traffic and quantify it in a database. The data will be used to validate the figures in the report, and to provide baseline data to help cope with unknowns.

Rationale: The system being provided requires traffic flow sensors to be installed. All the streets identified in the report will be instrumented to collect data.

Non-Functional Requirement: Fifty stop/go lights will be able to operate once one has interfaced with the control system. After this has taken place, inheritance can occur. Each of the fifty stoplights will be represented in the software by objects of the same abstract data type.

< See Table 6.1.5 of Requirements Specification >
<Inspection Check E. of Requirements Definition>

5.3.3 (*Requirement* 3) **Monitoring traffic flow and detecting bottlenecks or gridlock**

5.3.3.1 The software will provide for the monitoring of traffic flow and detection of bottlenecks or gridlock (see paragraph 1.1.3 of Concept Document).

Rationale: Facilitating the shortest possible travel times for people to go to and from work, and efficient deliveries of goods and services reduces stress, and promotes commerce and quality of life. This makes the city attractive to both workers and employers.

Non Functional Requirement: Magnetic sensors are working properly.

< See Table 6.1.6 of Requirement Specification>

<Inspection Check E. of Requirements Definition>

5.3.3.2 The traffic flow sensors can detect the number of vehicles in a given segment of road, and the speed that the vehicles are moving.

Real-time data is sent from the data collection nodes to the control center every 20 seconds. Armed with this data, you can tell if too many vehicles are going too slow. This constitutes a bottleneck. If the traffic is not moving at all, and the light is green, you have gridlock. Verification of this requirement will be by demonstration and observation: If the traffic flow system indicates one of these conditions, it can be verified by radio with the police. The converse is also true, if the police report that traffic has slowed or stopped, the traffic flow system should so indicate. For each occurrence of these conditions an entry is made into the database.

Rationale: This requirement is necessary for the entire model to work. The unimpeded flow of traffic is the heart of model being used.

Non-Functional Requirement: The software's database will contain an event's log capable of recording the rate of traffic flow and text messages describing specific events.

< See Table 6.1.7 of Requirements Specification >

<Inspection Check E. of Requirements Definition>

5.3.4. (*Requirement* 4) **Electric mass transit system adaptation**

5.3.4.1**The same software shall provide for adaptability to an electric mass transit system (See paragraph 1.1.8 of Concept Document).**

Rationale: The data gathered and the knowledge gained from running the traffic light system will be used to determine mass transit needs and through inheritance be capable of performing the same function for a mass transit system. **This requirement can be verified by simulation. A simulated virtual train system will be run and evaluated as part of phase two of the project.**

Non Functional Requirement: The city wants to have an Electric Mass Transit System

< See Table 6.1.8 of Requirements Specification >

5.3.5. (Requirement 5) **Deactivation and replacement of software**

5.3.5.1 Controller System shall provide provisions for the deactivation and the replacement of software due to upgrades or revisions (see paragraph 1.2.3 of Concept Document).

Rationale: The Object Oriented system lends itself easily to being replaced, since the principle of encapsulation allows each part of the system to operate separately. For example, the control room software doesn't care how it gets its real-time traffic flow data as long as it comes in every 20 seconds. We need only provide the proper inputs and outputs and therefore parts of the system can be upgraded without disabling the whole thing.

Non-Functional Requirement: The database of logged entries must be accessible outside of the software, so that upon deactivation, the logs may be integrated with the new software.

< See Table 6.1.9 of Requirements Specification >

5.3.6 (Requirement 6) **Overriding lighting control system**

5.3.6.1 Controller System shall provide for the operators to override the active functioning of the lamp switching (See paragraph 1.2.2 of Concept Document).

Rationale: There will be emergency or hazardous situations that need to be reflected within the light system of the intersection. If there is an accident, on the direction of flow that can still get through, the lamp switching should only allow for a timed system on those active lanes. If there is a repair of the physical system taking place on the wire unit, red flashers should come on to show caution and the slowing of flow for the awareness the car drivers.

Non-Functional Requirement: Users should be able to learn how to override and use the software after training for 3 hours. After this time users should make no more than 1 mistake for every 5 overrides.

< See 6.1.10 of Requirements Specification >

<Inspection Check F. Of Requirement Definition>

5.3.7 (Requirement 7) **Selecting Control System**

5.3.7.1 The system shall provide a menu for an operator to select the control system to monitor or override functions of a selected intersection. (See paragraph 1.2.3 of of Concept Document).

Rationale: The operator must be able to select which control system he/she wants to monitor or be able to override the functions within. The menu will allow for this selection.

Non Functional Requirement: Menu must be touch activated and functioning.

< See Table 6.1.11 of Requirement Specification>

<Inspection Check F. of Requirement Definition>

6. Requirements Specifications

6.1.1 Table

<See Requirement 5.3.1 of Requirement Definition>

Function	**Operation of a single four-way stoplight including a left turn arrow**
Description	**The software's data structures and algorithms will provide the logic to operate a four way stoplight with or without a left turn arrow. This logic will provide for optimum traffic flow based on time and day of the week. An object-oriented approach will be used in the design of the software.**
Inputs	**Information from traffic study commissioned by city planners. Signals from street sensors indicating traffic flow.**
Outputs	**Signals to traffic lights**
Destination	**The traffic light being monitored and controlled.**
Pre-Condition	**Accurate report of traffic study has been created and examined. Algorithms in the software are correct based on the traffic study report.**
Post Condition	**Traffic light is operating appropriately and traffic flow is optimal.**
Side effects	**If the lights do no receive their proper outputs, an accident could occur.**

6.1.2 Table

<See Requirement 5.3.1.2 of Requirement Definition>

Function	**Optimum traffic flow**
Description	**The magnetic sensors will provide information on the time of day, day of week, and flow of traffic. This information will be used to provide an optimum functionality of each intersection.**
Inputs	**Time of day, day of week, flow of traffic**
Outputs	**Intersection flow consistent with parameters and no problems**
Destination	**The control system that takes the input parameters and then functions accordingly.**
Requires	**Magnetic sensor functioning. Workstation software performance nominal.**
Pre-Conditions	**Magnetic sensors are talking to the control systems to provide this information.**
Post-Conditions	**The lamps are responding to the system to reflect the proper configuration.**
Side effects	**Lamps to not respond. Magnetic sensors are not responding to parameter input.**

6.1.3 Table

<See Requirement 5.3.1.3 and 5.3.1.4 of Requirement Definition>

Function	**Encapsulation requirement for upgrade ease along with basic data structure will allow for additional lights to be added.**
Description	**As each new traffic light system is added to the city, encapsulation within the software can occur. This will provide ease to the developers of the software, because each new system would be a duplicate of the past systems,containing all data structures previously used.**
Inputs	**Software for a single traffic system with turn lane and without turn lane.**
Outputs	**Software reuse to a new traffic system**
Destination	**New traffic system**
Requires	**Software code from existing traffic system**
Pre-Conditions	**A new traffic system is to be added to the city**
Post-Conditions	**The new traffic system contains reused code from a previous functioning traffic system in the city.**
Side effects	**Software code is not responding,**

6.1.4 Table

<See Requirement 5.3.2.1 of Requirement Definition>

Function	**Fifty stoplights to operate together throughout the city.**
Description	**The software will allow for up to fifty stoplights to operate in parallel throughout the city. Traffic flow sensor will be installed throughout the city to accomplish this. Data from the sensors will be collected and recorded for further research.**
Inputs	**Signals from street sensors indicating traffic flow.**
Outputs	**Signals to traffic lights. Data to record in logs.**
Destination	**The traffic light being monitored and controlled, and the data logs for recording information from the sensors.**
Pre-Conditions	**The first initial light must be set up and tested before the rest are installed.**
Post-Conditions	**Fifty stoplights will operate together throughout the city.**
Side effects	**Since all stoplights and sensors will be integrated, if one sensor is malfunctioning, it could affect traffic flow at many stoplights.**

6.1.5 Table

<See Requirement 5.3.2.2 of Requirement Definition>

Function	**Traffic sensors will be installed**
Description	**Traffic flow sensors can detect the number of vehicles in a given segment of road, and the speed that the vehicles are moving. Real-time data is sent from the data collection nodes to the control center every 20 seconds. Armed with this data, you can tell if too many vehicles are going too slow or how fast the traffic is moving.**
Inputs	**Traffic flow data from each node**
Outputs	**Quantitative and qualitative traffic flow data**
Destination	**Qualitative data goes to the traffic flow monitor in the control room. Quantitative data goes to the traffic flow database.**
Requires	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Pre-Conditions	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Post-Conditions	**Actual data on traffic flow, which will become the new standard defining actual conditions encountered on the roads**
Side effects	**Smoother transportation throughout the city.**

6.1.6 Table

<See Requirement 5.3.3 of Requirement Definition>

Function	**Monitor traffic flow and detect bottlenecks**
Description	**Traffic flow sensors can detect the number of vehicles in a given segment of road, and the speed that the vehicles are moving. Real-time data is sent from the data collection nodes to the control center every 20 seconds. Armed with this data, you can tell if too many vehicles are going too slow. This constitutes a bottleneck. If the traffic is not moving at all, and the light is green, you have gridlock. For each occurrence of these conditions an entry is made into the database.**
Inputs	**Traffic flow data from each node**
Outputs	**Quantitative and qualitative traffic flow data**
Destination	**Qualitative data goes to the traffic flow monitor in the control room. Quantitative data goes to the traffic flow database.**
Requires	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Pre-Conditions	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Post-Conditions	**Actual data on traffic flow, which will become the new standard defining actual conditions encountered on the roads**
Side effects	**Smoother transportation throughout the city.**

6.1.7 Table

<See Requirement 5.3.3.2 of Requirement Definition>

Function	**Monitor number of vehicles in a given segment of road**
Description	**Traffic flow sensors can detect the number of vehicles in a given segment of road, and the speed that the vehicles are moving. Real-time data is sent from the data collection nodes to the control center every 20 seconds. Armed with this data, you can tell how many vehicles go through any given intersection. For each occurrence of these conditions an entry is made into the database.**
Inputs	**Traffic flow data from each node**
Outputs	**Quantitative and qualitative traffic flow data**
Destination	**Qualitative data goes to the traffic flow monitor in the control room. Quantitative data goes to the traffic flow database.**
Requires	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Pre-Conditions	**Traffic flow sensors, communication from sensors to node, communication from node to the control center.**
Post-Conditions	**Actual data on traffic flow, which will become the new standard defining actual conditions encountered on the roads**
Side effects	**Smoother transportation throughout the city.**

6.1.8 Table

<See Requirement 5.3.4 of Requirement Definition>

Function	**Provide for the adaptation of an electric mass transit system**
Description	**The quantitative data collected in requirement 3 will provide insight on the transportation needs of the city. Through analysis of the data track layout ideas can be tested.**
Inputs	**Quantitative data from traffic flow sensors**
Outputs	**None**
Destination	**None**
Requires	**Database to store information collected.**
Pre-Conditions	**Nobody knows where the mass transit system should go although everyone agrees one will be required**
Post-Conditions	**A real plan for the mass transit system can be formed.**
Side effects	**Less commuter vehicles on the road providing space for more commercial vehicles.**

6.1.9 Table

<See Requirement 5.3.5 of Requirement Definition>

Function	Deactivation and Replacement of Software
Description	Deactivating and replacing of software for an updated version within the system. The operator will deactivate one workstation at a time, so that the control of the system will always be accessable through the other two workstations. The workstation in deactivation mode will have new software added to the system through disk/CD loading. The new software will override the existing software. All three workstations will end up on the new, current software.
Inputs	Disk/CD with new software
Source	Manufacturer of the software contract announcing new upgrades or replacements to existing system.
Outputs	Workstation to run on new software. May see displays different or system to run differently. Will have to train controllers for any advanced changes or possibly a memo will suffice.
Destination	Operating System
Requires	Disk/CD drive. Keyboard and screen accessibility.
Pre-Conditions	Workstation must be clear from any operations at the time.
Post-Conditions	Workstation will be running on the new software.
Side effects	The software does not load properly and the workstations may have to be on the existing software longer than projected. Some workstations are on the old versus some are on the new.

6.1.10 Table

<See Requirement 5.3.6 of Requirement Definition>

Function	**Override the active functioning of lamp switching**
Description	**Stop auto control of one of the workstation systems. Switch to manual control. Once in manual control, the operator can override the particular system to have the lamps respond to any needed changes. Ex. Emergency situation- need to have the lights change only for the direction of traffic that can still flow. Ex. Accident at intersection- have the lights change to red flashers.**
Inputs	**Emergency or maintenance to the outside unit at the intersection occurring.**
Outputs	**Manual control of the particular system. Operator overriding auto functioning of the workstation system. Lamps functioning correspond to set input of controller.**
Destination	**The system control that needs to be overridden to manual control.**
Requires	**Active workstation. Operator knowledgeable of the system to be overridden for emergencies or maintenance. Screen display output for a visual. Technician at the scene for confirmation that lamps functioning is corresponding.**
Pre-Conditions	**Emergency or technical repair is active at the particular intersection.**
Post-Conditions	**The lamps are responding to the manual override of the system to reflect the proper configuration.**
Side effects	**Lamps to not respond. All workstations are down and operator cannot get to manual intervention.**

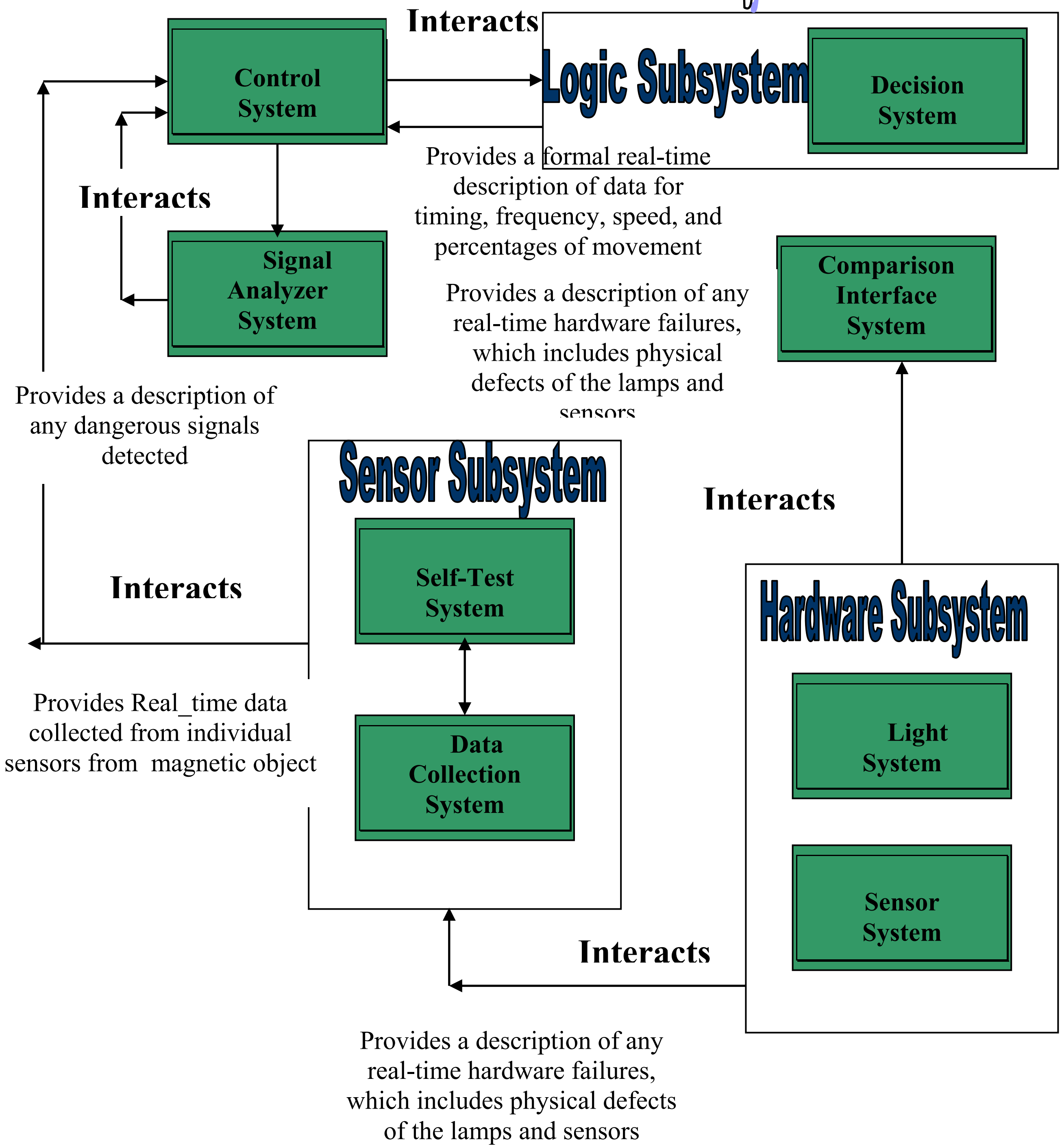
6.1.11 Table

<See Requirement 5.3.7 of Requirement Definition>

Function	**Selecting Control System**
Description	**The operators will select which control system they wish to monitor or override its functionality.**
Inputs	**Situation that arises for the specific monitoring or overriding of a particular system.**
Outputs	**Selected system being monitored or overidden**
Destination	**The control system that is selected**
Requires	**One of three active workstations. Operator knowledgeable of the system. Screen display output for a visual.**
Pre-Conditions	**Situation had occurred for the action of the operator to need to get into a system.**
Post-Conditions	**The selected system is being monitored or is being changed through the operator with manual intervention.**
Side effects	**All workstations are down and operator cannot get to manual intervention.**

7.0 Architecture

7.1 Structural /Data View of Architecture: Traffic Control System

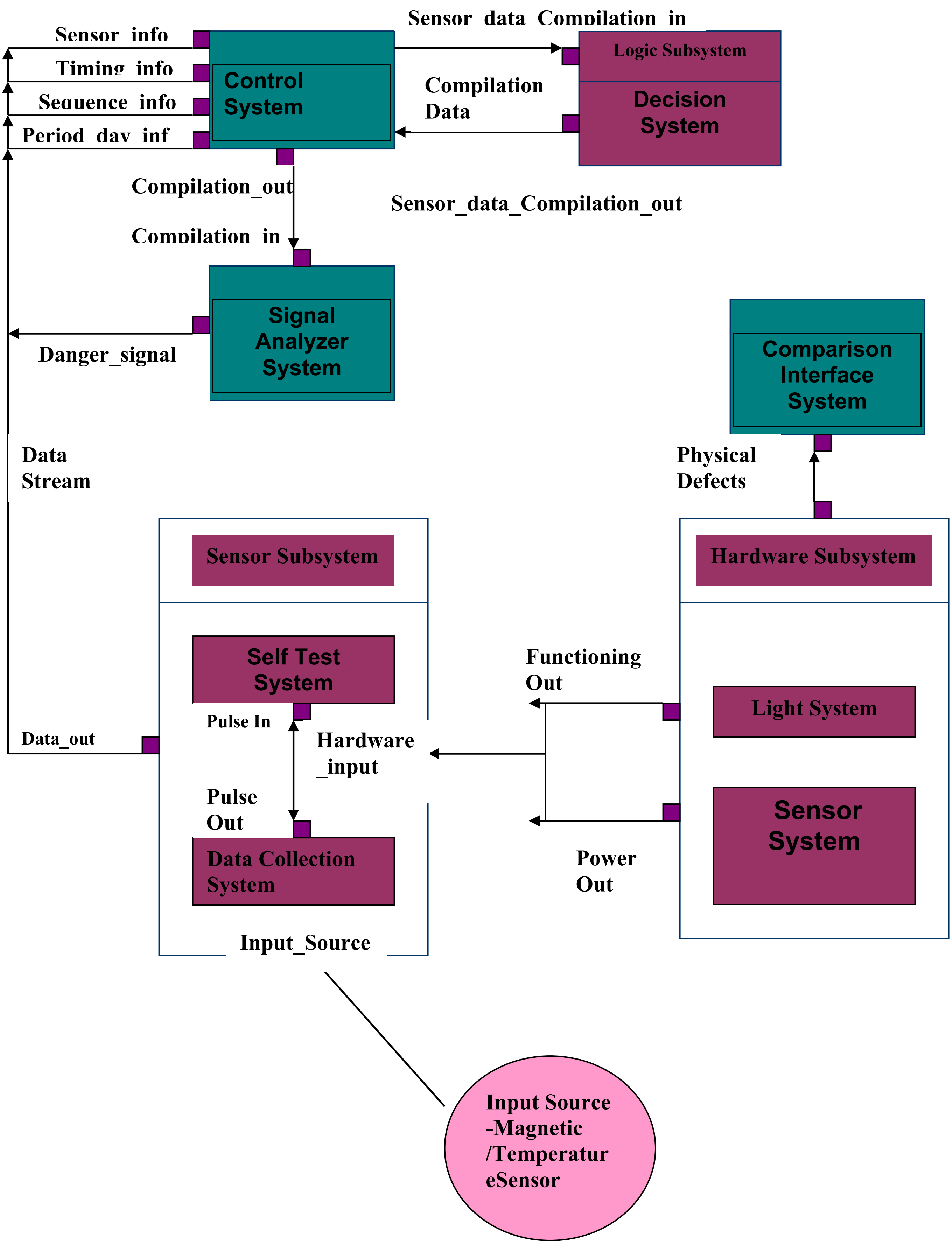


Textual annotations

<u>View:</u> **Shows the major types of data needed for each system along with major subsystems that make up the architectur e**

<u>Notation:</u> **Each Arch is labeled with the type of data provided**

7.2 Graphical-Interconnection View



7.3 Graphical Interface View Descriptions

This describes the graphical interface view of the architecture.

7.3.1 Component Control System

Input port list:

Port name Sensor_info: **data type** Sensor data
Port name Timing_info: **data type** Frequency data
Port name Sequence_info: **data type** Lane data
Port_name Period_Day_info: **data type** Time_of_day
Port name Compilation data: **data type** Collective_data
Port name Danger_signal: **data type** Danger_signs

Output port list:

Sensor_data_Compilation_in**: type** Real-time-data-description
Compilation_out **: type** Compilation-of-sensors.

Direction flow list:
Sensor_data_compilation_in => sensors.sensor_data_compilation_out
Compilation_out => compilation.compilation_in

General Behavior Spec.
The Control System actually is running off of several more parameters than are shown.

Constraints

The Control System will only work through its logic algorithm when the sensors are functioning.

7.3.2 Component Signal Analyzer System

Input port list:

Port name Compilation in: **data type** Collective_data

Output port list:

Danger Signal **: type** Danger signs

Direction flow list:
Danger Signal => Data Stream

General Behavior Spec.
A computational step provides the danger signal, if necessary, from the compilation in data.

Constraints

The Signal Analyzer System is dependant upon the functioning capabilities and correctness of outputs from the Control System.

7.3.3 Component Logic/Decision SubSystem

Input port list:

Port name Compilation in: **data type** Compilation-of-sensors

Output port list:

Compilation_Data **: type** Real-time-data-description

Direction flow list:
Sensor_data_compilation_out => sensors.sensor_data_compilation_in

General Behavior Spec.
The Logic/Decision Subsystem takes into account many data inputs to calculate the outputted sensor data.

Constraints

The Logic/Decision Subsystem is dependant upon the functioning capabilities and correctness of outputs from the Control System. Errors in the Control System will result in errors in the Logic/Decision Subsystem.

7.3.4 Component Sensor SubSystem

Input port list:

Port name Magnetic/Temp Sensors: **data type** Hardware data
Port name Functioning Out: **data type** Operational data
Port name Power Out: **data type** Operational data

Output port list:

Data Stream out **: type** Real-time-data-description

Direction flow list:
Data Stream out => sensor/timing/sequence/period of day info

General Behavior Spec.
The Sensor Subsystem takes into account the hardware inputs shown to gather and output sensor, timing, sequence, and period of day data.

Constraints

The light systems hardware and the Hardware Subsystem must be functioning and generating proper output to the Sensor Subsystem for it to work as expected.

7.3.5 Component Hardware SubSystem

Output port list:

Functioning Out **: type** Operational data
Power Out **: type** Operational data
Physical Defects **: type** Real-time-hardware data

Direction flow list:
Functionint Out => Hardware input
Power Out => Hardware input

General Behavior Spec.
The Hardware Subsystem strictly generates output to other subsystems based on sensor triggers from within the Hardware Subsystem itself.

Constraints

The lights and road sensors must be set up properly for the Hardware Subsystem to function correctly.

7.3.6 Component Comparison Interface System

Input port list:

Port name Physical Defects: **data type** Real-time hardware data

General Behavior Spec.

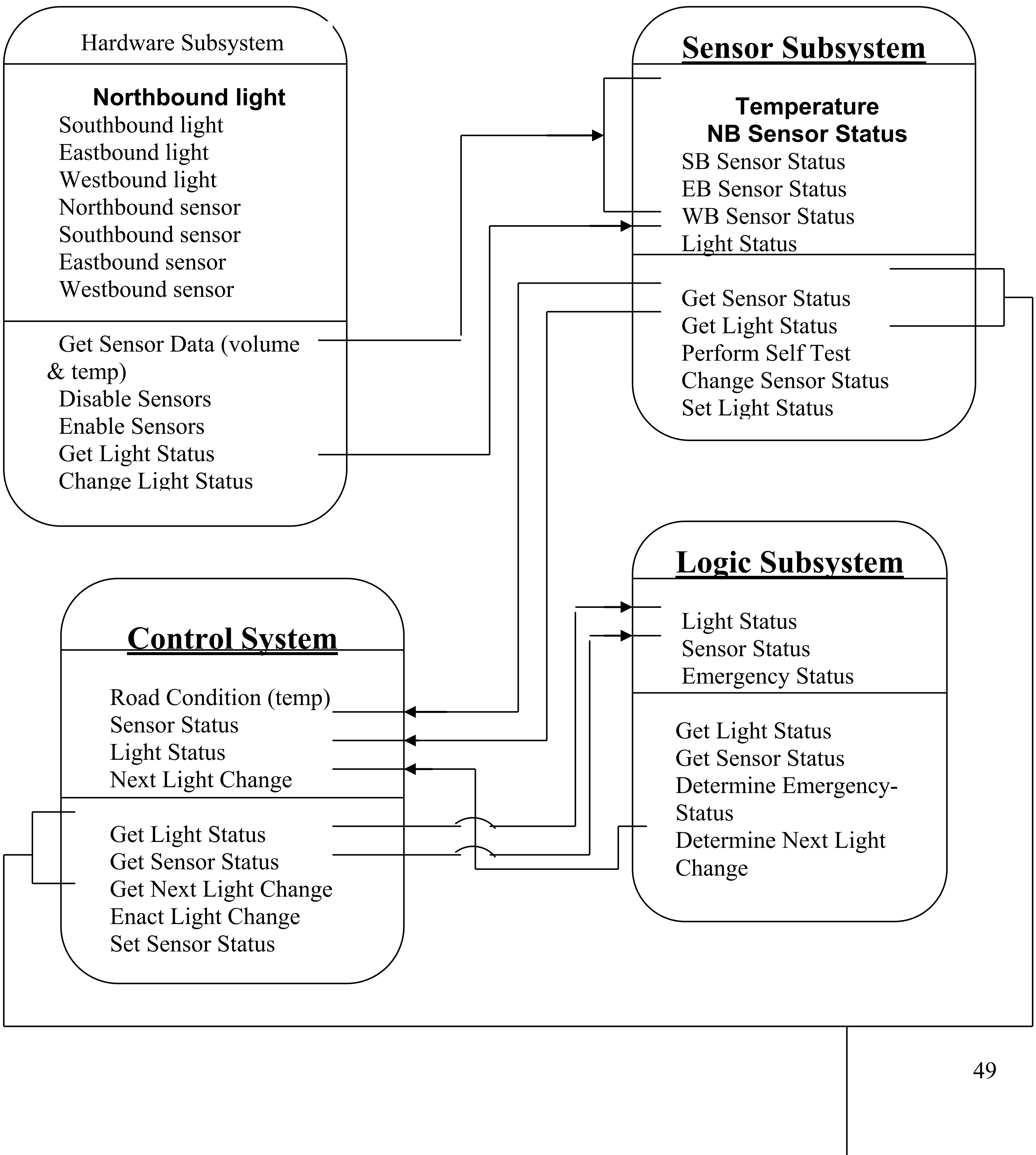
The Comparison Interface System strictly takes in input from the Hardware Subsytem to make records of descriptions of physical defects in the system.

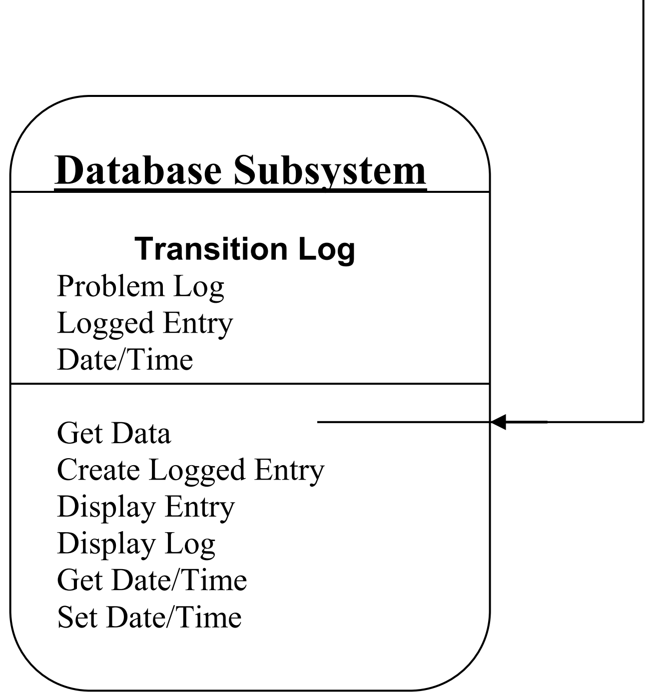
Constraints

The Comparison Interface System is dependent upon the Hardware Subsystem. The output of the Hardware Subsystem must accurately reflect the functioning of the hardware for the Comparison Interface System to correctly record data.

7.4 Object Oriented Decomposition of Subsystems

Each box diagramed below contains the name of the subsystem it represents, the data items contained in the subsystem, and the procedures the subsystem is responsible for. The lines with arrows show the direction of data flow among the subsystems.




Database Subsystem
Transition Log
Problem Log
Logged Entry
Date/Time
Get Data
Create Logged Entry
Display Entry
Display Log
Get Date/Time
Set Date/Time

8 Design

8.1 Object-Oriented Design. Group three has chosen an object-oriented approach to the traffic light project. A traffic light system is after all a collection of objects. Each object has a job to do and works independently of the other objects. The overall traffic system is held together by an operating system that makes decisions on what each object is to do, and collects data.

The traffic sensors are the heart of the system. They feed real time data on traffic conditions to the node, which sends it to the control center and the database. Each one works independently, registering a signal each time it detects a magnetic object going over. Each sensor, without regard for what the other sensors are doing, collects data, which is transferred to its respective node. The traffic sensors also have temperature sensors included in the pavement and send temperature data in case of freezing conditions. Each node and its associated sensors is also an object. It is an entity that collects data about its assigned stretch of road. It has a single input to the control center. It is also a sub-system of its own doing self-tests on the sensors that feed it information and flagging bad sensors as a measure of redundancy. An example of its self-testing is to take three sensors and compare the outputs. If the first and third register a strong blip and the middle one has a barely perceptible blip then the middle one will be identified as malfunctioning. This is done as a continuing self-test. The self-test is in parallel with the data collection and therefore there is no overhead. The malfunction data is stored in the node and is checked periodically. All the data required to troubleshoot the sensors is already collected thus less time and manpower is required to do maintenance cutting down on costs. The savings are realized not only by less man-hours, but also less down time for the traffic system.

The next object in the system is the decision making portion of the control system. The timing of traffic signals is based on the efficient distribution of the available time among the signal movements of phases. Along a street with synchronized signals there is a base cycle length of, say 60, 80, 100, or 120 or more seconds. Each intersection to be synchronized must have this cycle length. The computer figures out what the cycle length should be from the distance between the intersections, the speed limit, and the minimum time to service the busiest intersection's traffic. Although at this point in time there is only one traffic light, it is evident that if there were many lights, and each light was treated as a data object in a part of the whole system it is easier to control. If one light should malfunction, the other lights can become synchronized to each other. If the temperature sensors detect the likelihood of icing conditions the control room has the option to go into another mode of operation which uses a longer cycle, and a longer time between the yellow light going out and the green in the cross street coming on. The control room software produces a graphic representation of the city streets, derived from data on vehicle turning movement, pedestrian crossing volumes, channelization, parking restrictions, signal phasing and minimum timing, speed limit, pavement temperature, and percent of heavy vehicles. More concerns can be added if they become important. Control room personnel rely on this graphic representation to interact with the system.

The traffic lights themselves are objects as discussed earlier, but the most important object is the data itself. The Universal Traffic Data Format (UTDF) is used in this system. UTDF is a standard data format used for transferring data between traffic software packages. There are six file formats specified in UTDF: VOLUME, TIMING, PHASING, TIMEOFDAY, LAYOUT, and LANES. The VOLUME stores traffic volume counts. There are three different options to use for volume depending on each individual situation. TIMING stores timing plan information that varies by time of day. PHASING stores timing plan information that doesn't change. TIMEOFDAY stores a weekly schedule of when to use timing plans. LAYOUT stores intersection locations and connections. LANE stores lane and fixed information. Volume is by far the largest data file. With fifty intersections generating 96 counts per day for 30 days the VOLUME table can have 144,000 records. The rest of the files are relatively small.

8.2 Object Identification

Node

Data storage and
Communication Device
Hardware

1. Provides real-time data
2. Conducts diagnostics
3. Transmits data to the Control Center

Volume

Data Structure
Stores turning counts at each intersection

DATE	date of data
TIME	time of day
COMMENT	data sourse
INTID	intersection ID
MINUTES	minutes for count
xL	left turn volume
xT	thru lane volume
xR	right turn volume
xL2	special
xR2	special
xPD	special
XBS	count bus or HOV
xLO	lane occupancy time

x = NB (northbound), SB (southbound), EB (eastbound), WB (westbound), NE (north-east bound), NW (north-west bound), SE (south-east bound), SW (south-west bound)

Traffic sensors

Temperature Sensor
Magnetic detector
Hardware

Provides traffic flow information

Volume

Data Structure
Counting Vehicles if less than 10 counts per day

Header
Column Labels
Separator Line
Data lines 1 to n
Blank line
Additional Data sets

Temperature

Sensor for icing conditions on the road
Supplies continuous data, sampled more or less often depending on

Volume

Data Structure
Stores counts for each intersection

IntID:x	Specifies intersection as x
Reference #x	Specifies intersection ID
Date:	mm/dd/yy
Date:	mm/dd/yyyy
Date:	yyyy/mm/dd
Time:	hhmm military format
x Minute	Specifies minutes as x
Duration:x	Specifies duration as x
Collected by:x	Comment as x

Timing Table

Data Structure.
Stores multiple timing plans for each intersection

INTID	intersection ID
PLANID	timing plan
CycleLen	Cycle length in seconds
Split 1	Split for phase 1 in seconds
Split2	Split for phase 2 in seconds
...	
Split32	Split for phase 32 in seconds
Offset	Offset in seconds%
Leading	Leading phases
RefPhase	Coordinated phases for reference
Clearance	Yellow or Don't Walk clearance used with 170 offsets

TimeOfDay

Data Structure.
Stores the schedule of timing plans for each intersection by time of day

PlanID	Name of the timing plan
IntID	Intersection ID
Days	Days the plan is used
Start Time	Time to start
End Time	Time to end

Phasing

Data Structure
Stores detailed information on timing and phasing for each intersection

IntID	Intersection ID
RecordName	Name of data to follow
Data1	Data for phase 1
Data2	Data for phase 2
...	
Data32	Data for phase 32

8.3 Data Dictionary

(Some of these data types may not be used, but the capability is in the system.)

8.3.1 Special Data Structures

Entity Name	Type	Description
TIME	Structure	hhmm military time
DATE	Structure	mm/dd/yy
DATE	Structure	mm/dd/yyyy
DATE	Structure	yyyy/mm/dd
INTID	Structure	dd### where dd is direction nn for north nw for northwest etc. and # is the number assigned to the traffic signal
PLANID	Structure	User defined timing plans. Examples could be AM1, PM1, MIDDAY, PREGAME, POSTGAME.
COMMENT	Structure (Optional)	Up to 16 characters to describe the data collection method such as hand counts or automatic counts from detectors.
TEMPERATURE	Structure	Three digit Fahrenheit reading sensed in the pavement and decoded in the Node which adds Intersection ID

8.3.2 Volume Turning Counts

Entity Name	Type	Description
DATE	Structure	Date data collected
TIME	Structure	Time of Day
COMMENT	Structure	Data Source
INTID	Structure	Intersection identification
MINUTES	int	Minutes for count
xL	int	Left Turn Lane Volume
xT	int	Through Lane Volume
xR	int	Right Turn Lane Volume
xL2	int	Special
xR2	int	Special
xPD	int	Special
xBS	int	Count Bus or HOV
xLO	int	Lane Occupancy Time

8.3.3 Volume Intersection Counts

Entity Name	Type	Description
INTID	Structure	Intersection Identification
DATE	Structure	Data Structure, One of three date formats
DATE	Structure	Data Structure, One of three date formats
DATE	Structure	Data Structure, One of three date formats
TIME	Structure	Data Structure, Time of instance, military time format
MINUTES	int	Minutes for Count
DURATION	int	Time in lane
COMMENT	Structure	Data Source

8.3.4 Volume Small Counts

Entity Name	Type	Description
INTID	Structure	Intersection Identification
DATE	Structure	Data Structure, One of three date formats
MINUTES	Int	Minutes for Count
xR	Int	Right Turn Lane Volume
xL	Int	Left Turn Lane Volume

8.3.5 Timing Table

Entity Name	Type	Description
INTID	Structure	Intersection Identification
PLANID	Structure	Timing Plan
CycleLen	int	Cycle length in seconds
Split1	int	Split for phase 1 in seconds
Split2	int	Split for phase 2 in seconds
...	...	...
Split32	int	Split for phase 32 in seconds
Offset	int	Offsets are referenced to the phase indicated in the RefPhase field
Leading	int	Leading Phases are included so that the phase order can change for different timing plans. Made up of four digits with each position indicating the leading phase position
RefPhase	int	Coordinated phases for referencing offsets. A value of 26 indicates phases 2 and 6 are coordinated and offsets are referenced to the last phase to turn green
Clearance	int	Yellow or don't walk clearance used with 170 offsets

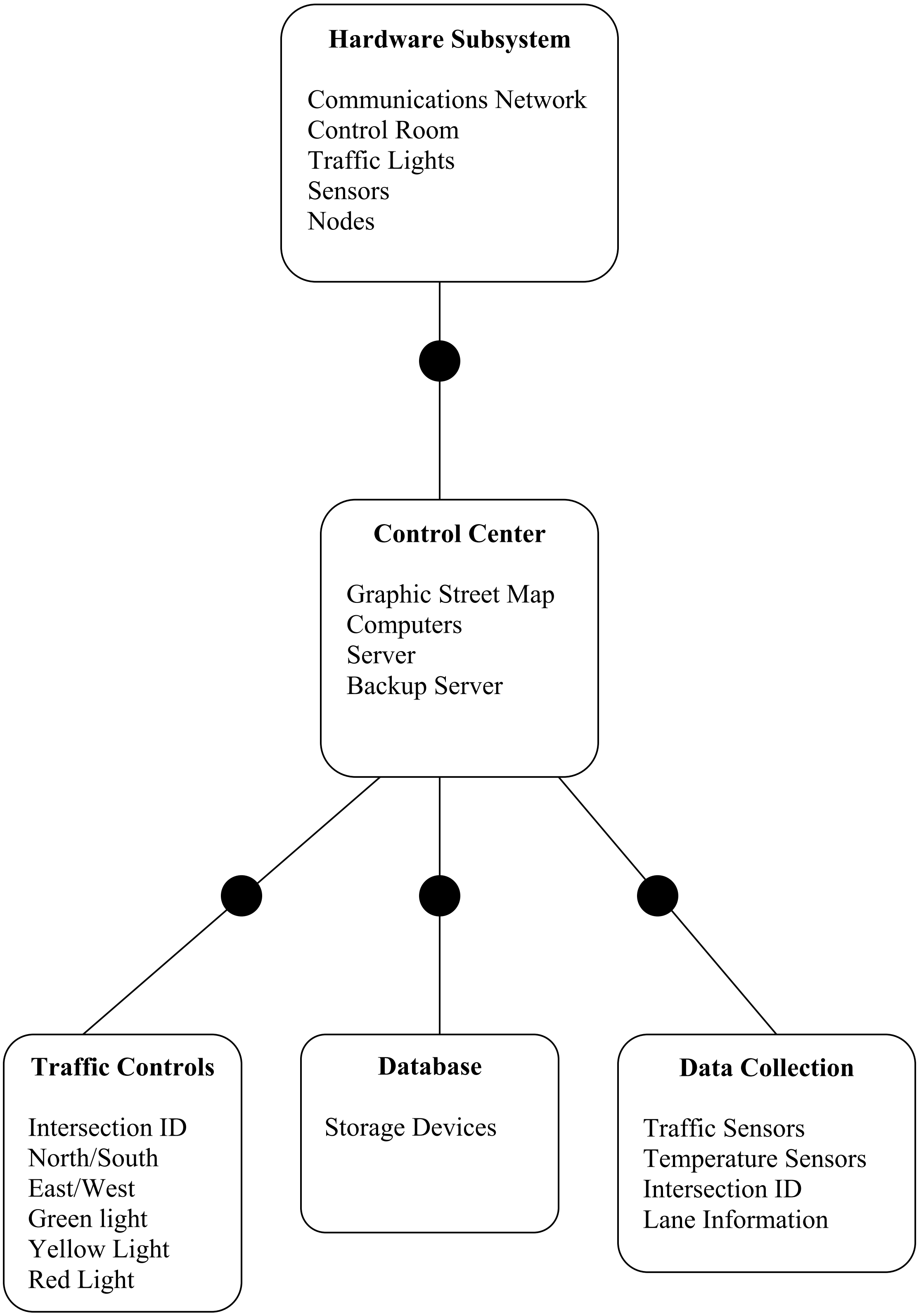
8.3.6 Time of Day

Entity Name	Type	Description
PLANID	Structure	Timing Plan
INTID	Structure	Intersection Identification
DAYS	int	Days of the week Sunday thru Saturday shown as 0 thru 6
StartTime	TIME	Starting time stamp
EndTime	TIME	Ending time stamp

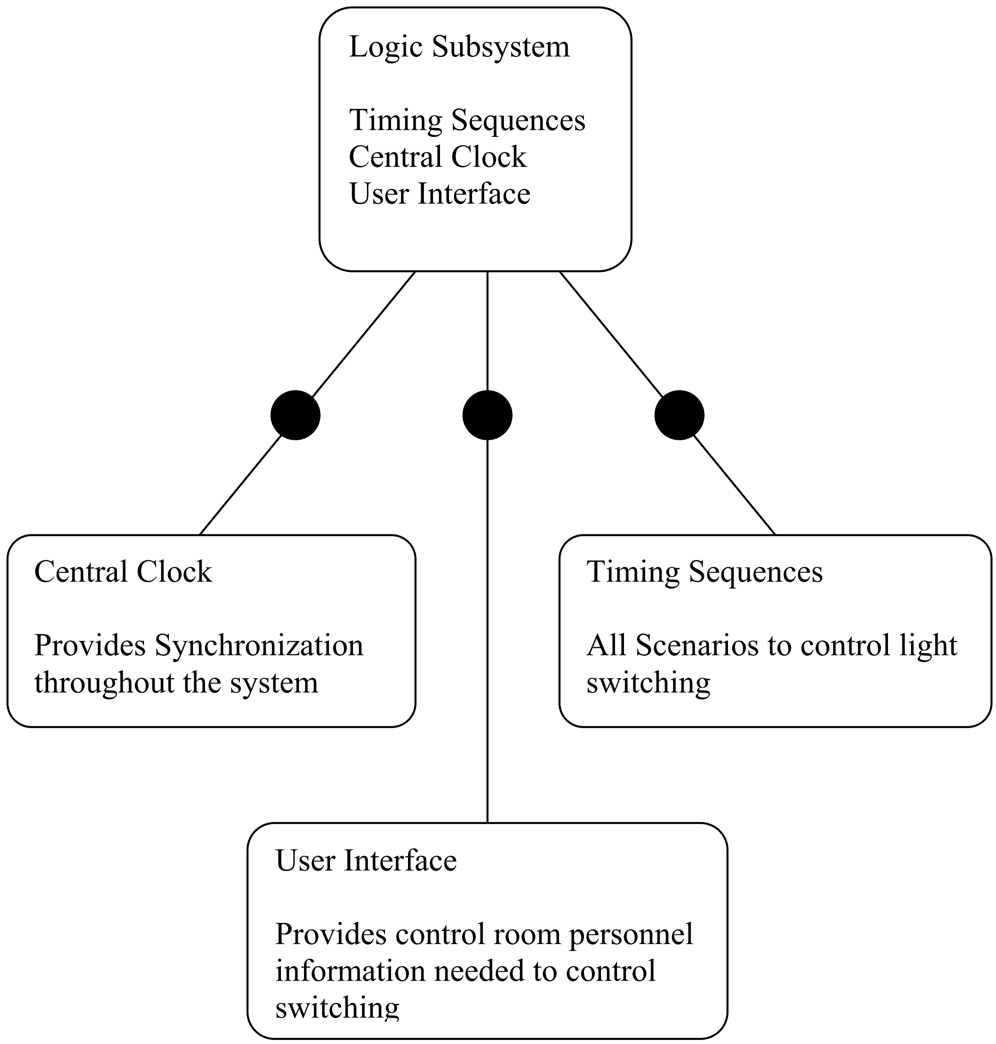
8.3.7 Phasing

Entity Name	Type	Description
INTID	Structure	Intersection Identification
RecordName		Name of data to follow
Data1		Data for phase 1
Data2		Data for phase 2
...		
Data32		Data for phase 32

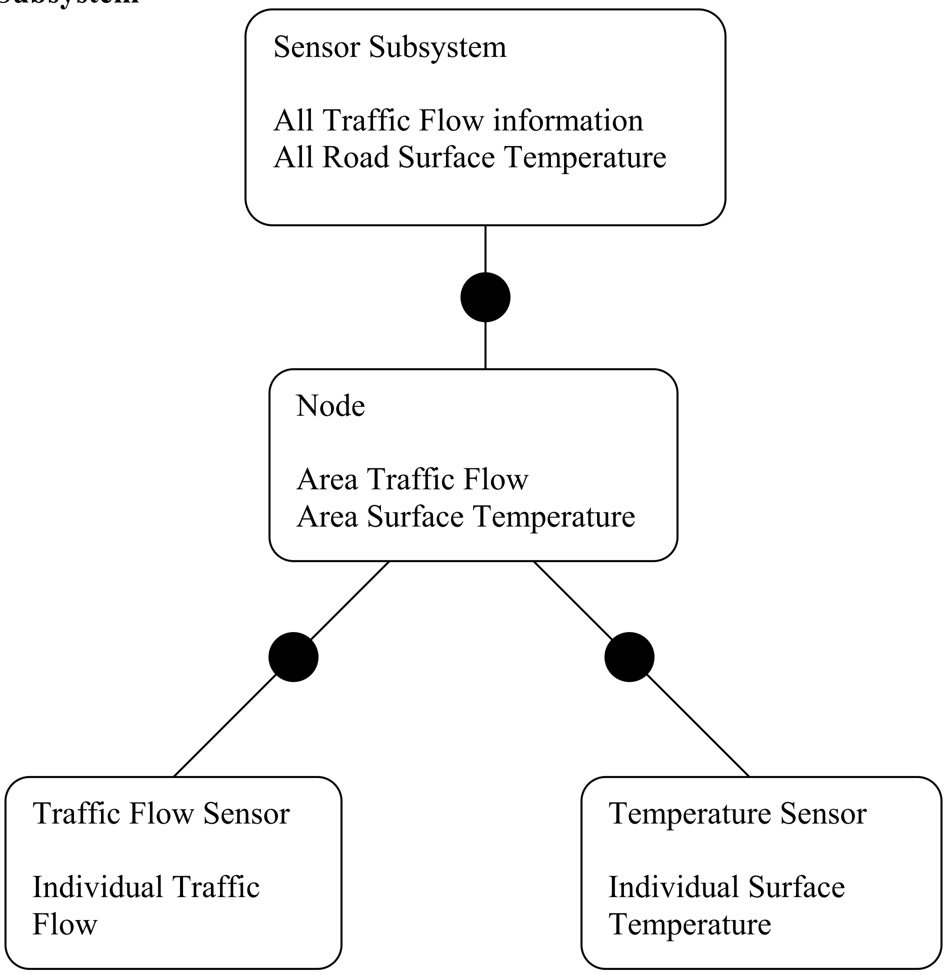
8.4 Object Aggregation Hierarchy
8.4.1 Hardware Subsystem



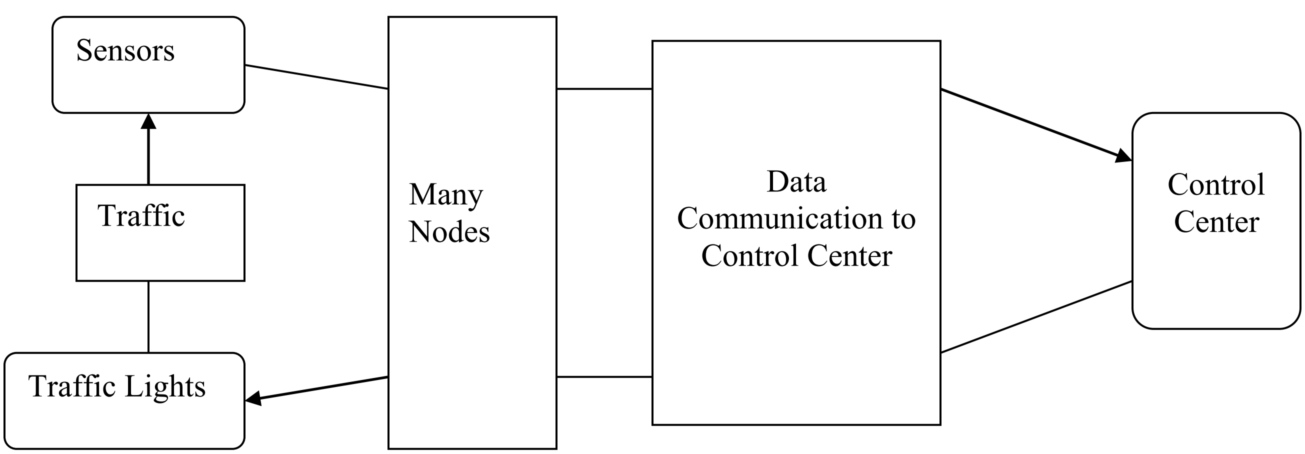
8.4.2 Logic Subsystem



8.4.3 Sensor Subsystem

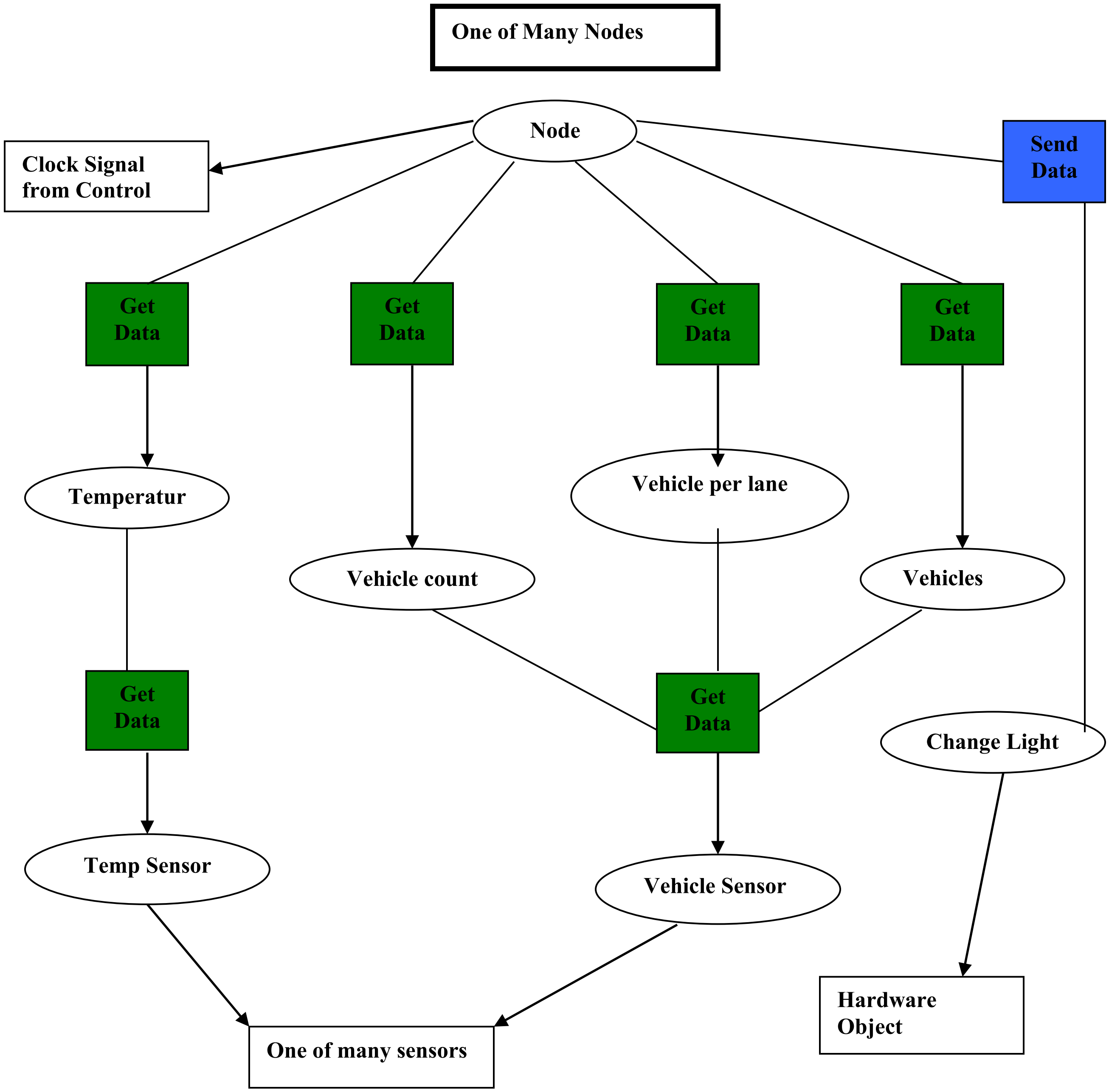


8.5 Object Interactions

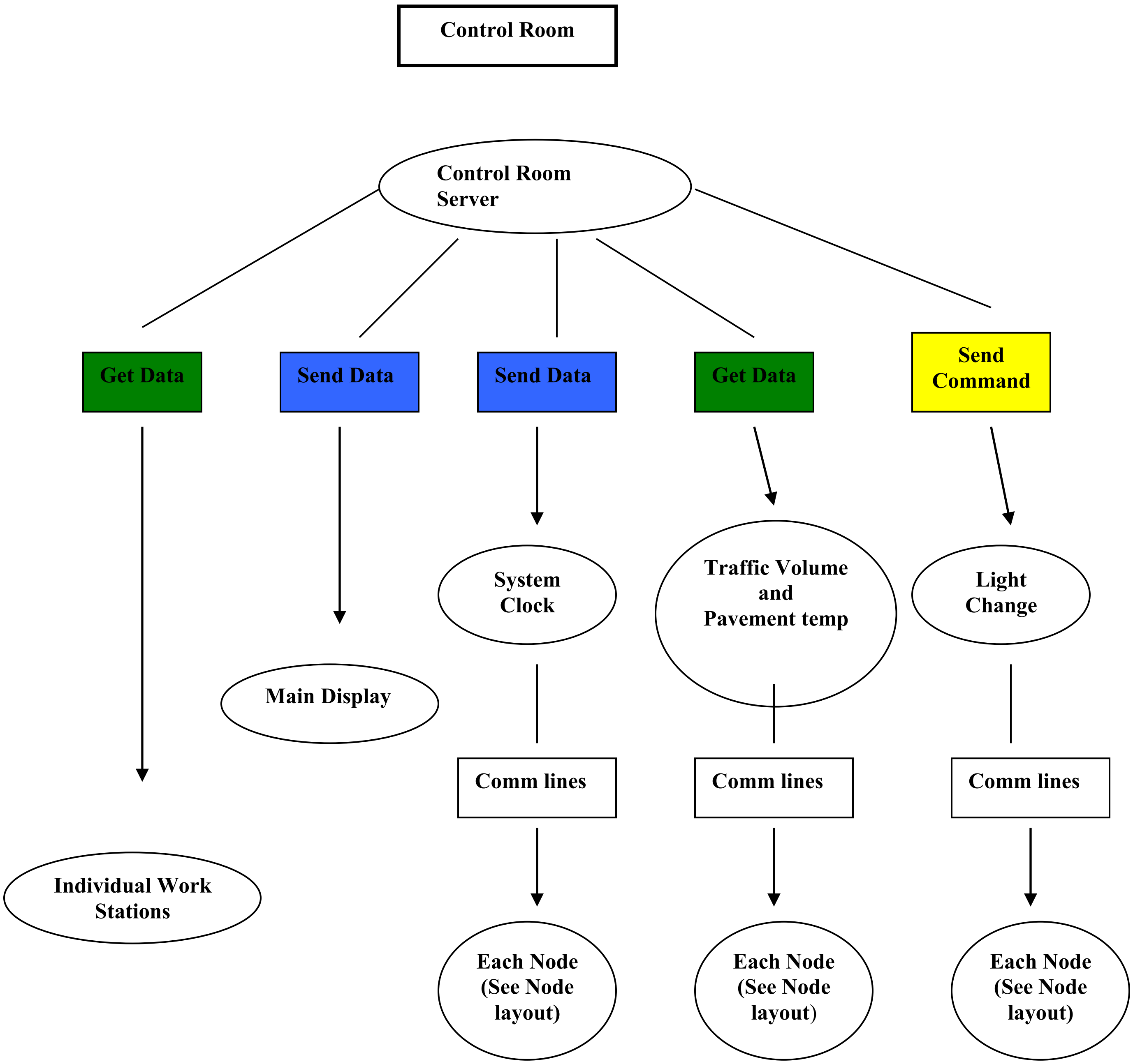


8.6 Composition Object Interaction Chart
8.6.1A

(These Object Interaction Layouts are much simplified to stay on two pages)



8.6.1B


Control Room
Control Room Server
Get Data
Send Data
Send Data
Get Data
Send Command
System Clock
Traffic Volume and Pavement temp
Light Change
Main Display
Comm lines
Comm lines
Comm lines
Individual Work Stations
Each Node (See Node layout)
Each Node (See Node layout)
Each Node (See Node layout)

8.7 Object Interfaces

```
class Node{
public:
        Node( );
                //constructor
        ~Node( );
                //destructor
        void Count(struct INTID, struct TIME);                  //out
                //sends traffic flow data to logic subsystem
        void Temperature(int temp, struct TIME);                      //out
                //sends pavement temperature to control room display
        void Switching(struct INTID, struct Phasing);           //in/out
                //tells the light when to switch
                //tells the logic subsystem that the switch took place
        void Sensor_Status( int selftest_sensor);               //out
                //this is actually stored in the node hardware for maintenance use
        void Light Status(int selftest_light);                        //out
                //feedback to control room on light failures
private:
        int sensor_count;                                       //in
                //actual integer value of vehicle count
        int sensor_temp;                                        //in
                //actual integer value of pavement temperature
        int selftest_sensor;                                    //in
                //selftest status input from the sensor
        int selftest_light;                                     //in
                //selftest status input from light
};
```

```
class Record{
        Record( );
                //constructor
        ~Record( );
                //destructor
private:
//These are all data elements stored in the database
        date(struct DATE);
        start_time(struct TIME);
        end_time(struct TIME);
        intid(struct INTID);
        nbl(int count);
        sbl(int count);
        ebl(int count);
        wbl(int count);
        nwbl(int count);
        sebl(int count);
        nebl(int count);
        swbl(int count);
        temp(struct TEMPERATURE);
};
```

```
class Plan{
        Plan( );
                //constructor
        ~Plan( );
                //distructor
public:
        AM1(struct TIME, int DAYS);                     //out
                //first of three different morning traffic plans
        AM2(struct TIME, int DAYS);                     //out
                //second of three different morning traffic plans
        AM3(struct TIME, int DAYS);                     //out
                //last of three different morning traffic plans
        PM1(struct TIME, int DAYS);                     //out
                // first of two different evening traffic plans
        PM2(struct TIME, int DAYS);                     //out
                // second of two different evening traffic plans
        MIDDAY(struct TIME, int DAYS);                  //out
                //normal mid day traffic pattern
        PREGAME(struct TIME, int DAYS);                 //out
                //special traffic plan for before sporting events
        POSTGAME(struct TIME, int DAYS);                //out
                //special traffic plan for after sporting events
```

```
};
class CLOCK{
        CLOCK( );
        ~CLOCK( );
private:
        set_time(int hh, int mm);                               // in / out
                //used to set the clock
        set_date1(int mm, int dd, int yy);                      // in / out
                //used to set the date in configuration 1
        set_date2(int mm, int dd, int yyyy)                     // in / out
                //used to set the date in configuration 2
        set_date3(int yyyy, int mm, int dd)                     // in / out
                //used to set the date in configuration 3
};
```

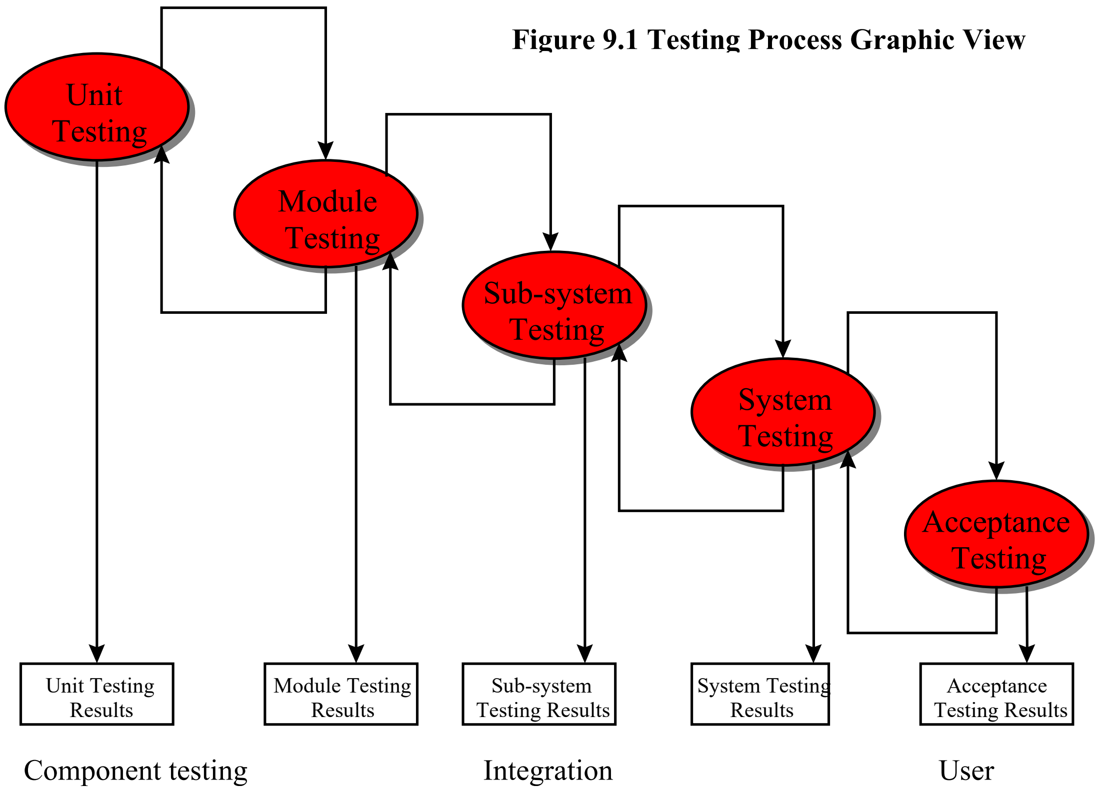
```
class INTID{
        INTID( );
                //constructor
        ~INTID( );
                //destructor
public:
        assign_ref_num(int num);                                //out
                //intersection number assignment
        assign_ref_direction(char dir[2]);                      //out
                //intersection direction assignment
        assign_PLANID(char[10]);                                //out
                //The timing plan assignment
        assign_CycleLen(int cyc);                               //out
                //The length of the cycle being used now
        Offset(int off);                                        //out
                // light color, 1- red; 2-yellow; 3-green
        assign_Leading(int lead);                               //out
                //assignment of leading phase
        assign_RefPhase(int refphase)                           //out
                //assignment of reference phase
        assign_Clearance(int clear)                             //out
                //assignment of clearance time
private:
        int num;
                //actual value for intersection number
        char dir[2]
                //actual direction assigned to the intersection
```

9 Testing

I. TEST PLAN

9.1. Testing process

Figure 9.1 Testing Process Graphic View



Description

9.1.1. Unit testing: Control System Testing, Signal Analyzer System Testing, Logic/Decision Sub-System Testing, Interface Control System Testing and Sensor Subsystem Testing involved in this test process. In this test process, each component testing tested independently.

- General System Testing: As soon as the program starts, seven windows are shown. The windows include : Control System, Signal Analyzer System, Interface Control System, Logic Subsystem, Sensor Subsystem, help, and exit.

  Each system window will proceed into individual selection options. Help function provides some software information to user.
  Within each system window, there are two options: Intervention and Monitoring.

- Control System Testing: It is a frame with controls related to light and sensor status

  Intervention buttons: Set Sensor Status
  Enact Light Change

Monitoring buttons: Display Sensor Status
Display Light Status
Display Next Light Change Info.

Signal Analyzer System Testing: It is a frame for displaying descriptions of dangerous signals detected

Monitoring buttons: Display Sensor Status
Display Light Status
Display Emergency Status
Display Emergency Description

Interface Control System Testing: Frame provides information and control of real time hardware failures

Intervention buttons: Disable Sensors
Disable Lights

Monitoring buttons: Display Hardware Status
Display Failure Description

Logic Subsystem Testing: This frame gives control over and a real time description of the data that goes into the light changing algorithms.

Intervention buttons: Change Time Delay
Set Time/Date
Change Light Status

Monitoring buttons: Display Light Status
Display Sensor Status
Display Next Light Change

Sensor Subsystem Testing: This frame monitors the flow of real time data collected from the road sensors

Intervention buttons: Perform Self Test
Change Sensor Status
Set Light Status

Monitoring buttons: Display Light Status
Display Sensor Status

9.1.2. Module Testing

Displaying Control Testing: Displaying the control testing is dependent upon the SignalAnalyzer System, the Logic Subsystem, and the Sensor Subsystem. Sensors are triggered through the Sensor Subsystem, the sensor data is passed through the Control System to the Logic System and Signal Analyzer System to be analyzed, and then looped back to the Control System.

9.1.3. Sub-system Testing

Sensor Sub-system Testing: The pressure sensors in the road intersections are being checked every 20 seconds for accuracy of information collection.

Logic/Decision Sub-system Testing: From the Control System window, this subsystem can be intervened. Under this option, there are several choices: Time delay changes and Lamp intervention. Time Delay changes allow per Sensor Subsystem information, the ability for light intersection timing to increase or decrease. Lamp Intervention gives the controllers the ability to change the lights accordingly in response to accidents or hazard /technician situations. This is the final stage testing for user to test operational usage.

9.1.4. System Testing

This testing process is concerned with finding errors which result from unanticipated interactions between sub-systems and system components. Select the Logic/Decision Subsystem from the Control System. Make sure that all Sensor Subsystem data is intact and that it is updating consistently

9.1.5. Acceptance Testing

This is the final stage testing for user to test operational usage.

9.2. Testing schedule

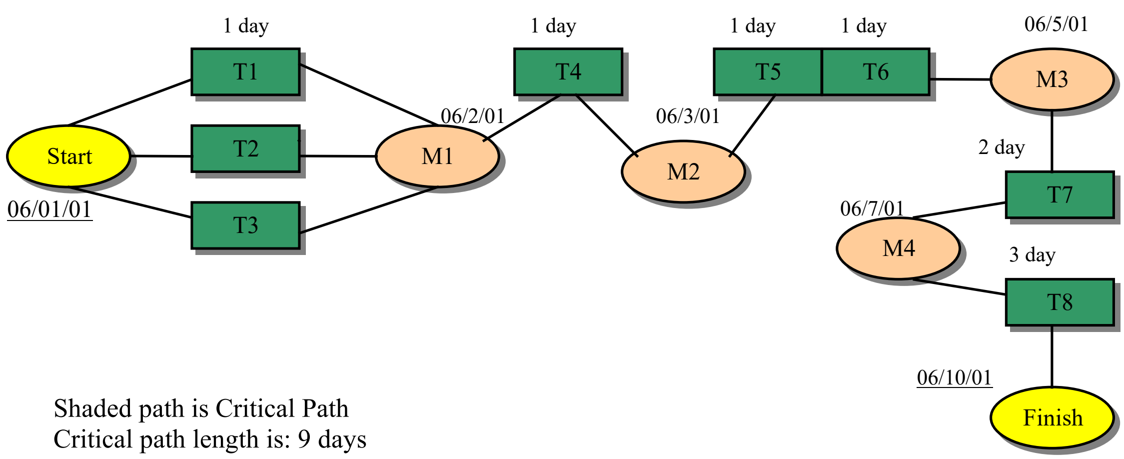
9.2.1 Task Duration and Dependent Table

Task	Description	Duration in Work Days	Dependent Upon
T1	Control System Testing	1 day	None
T2	Signal Analyzer System Testing	1 day	None
T3	Interface Comparison System Testing	1 day	None
T4	Sensor Sub-System Testing	1 day	T1,T2,T3
T5	Logic/Decision Sub-system	1 day	T4
T6	Sub-system Testing	1 day	T4
T7	System Testing	2 days	T5,T6
T8	Acceptance Testing	3 days	T7

9.2.2 People -Task Allocation Table

Task	People Name
Testing 1	Mary and Renee
Testing 2	Renee
Testing 3	Frank
Testing 4	Mary, Renee, Frank, and Bill
Testing 5	Mary, Renee, Frank, and Bill
Testing 6	Frank and Bill
Testing 7	Mary, Renee, Frank, and Bill
Testing 8	Frank and Mary

9.2.3 Activity Network



Shaded path is Critical Path
Critical path length is: 9 days

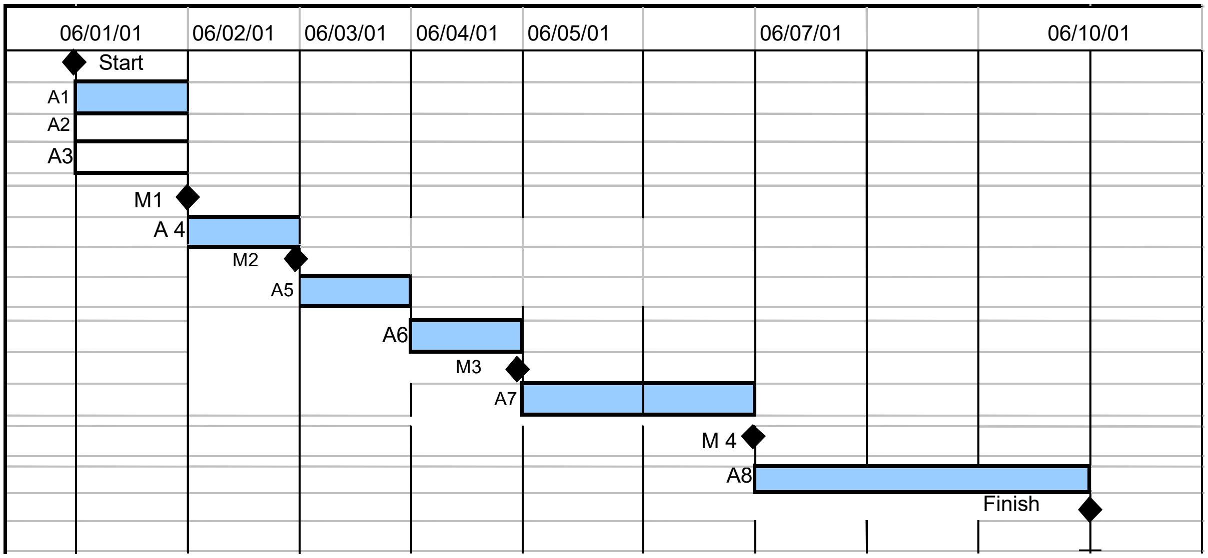
Milestone M 1 : The conclusion of the Unit Testing will be generated. Completion of this testing indicates that the milestone has been reached.

Milestone M 2: The conclusion of the Module Testing will be generated. Completion of this testing indicates that the milestone has been reached.

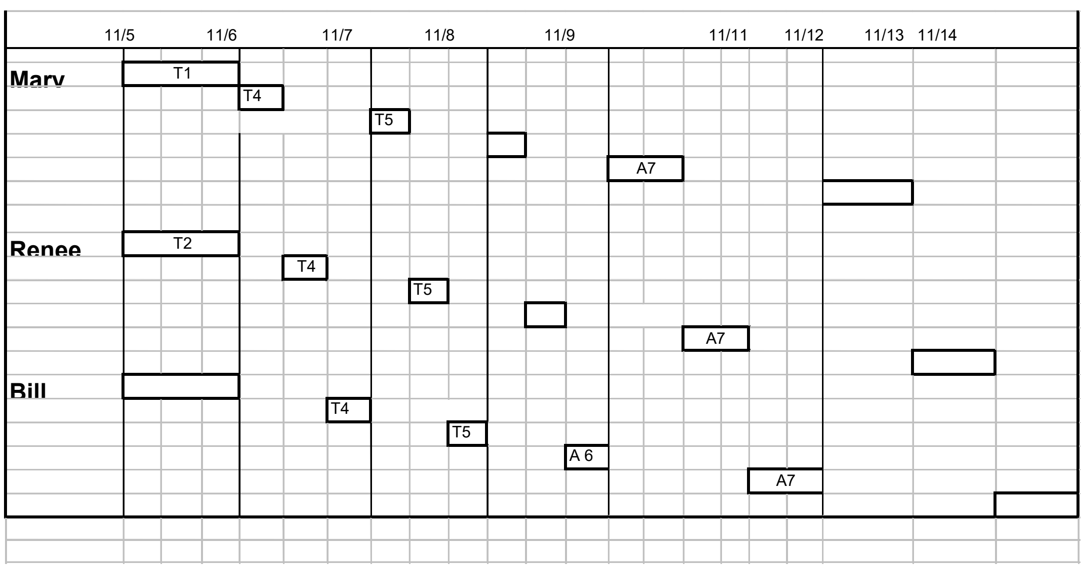
Milestone M 3: The conclusion of the Sub-system Testing will be generated. Completion of this testing indicates that the milestone has been reached.

Milestone M 4: The conclusion of the System Testing will be generated. Completion of this testing indicates that the milestone has been reached.

9.2.4 Activity Bar



9.2.5 People Allocation verses Time



9.3. Testing recording procedure

Step	Test	Action	Result	Error Handling
	Test 1			
Step 1	Test 2			
	Test 3			
Step 2	Test 4			
Step 3	Test 5			
Step 4	Test 6			
Step 5	Test 7			
Step 6	Test 8			

9.4. Hardware and Software Requirement

- Hardware: A computer with a CD ROM driver, floppy disk drive, keyboard and mouse or other pointing device

- Software: Windows Environment (Windows 95 & 98), Excel type database spreadsheet for logging entries.

Category	Availability		
	On-Hand	**To Be Purchased**	**Leased**
Hardware			
CD ROM/Floppy Drive	**X**		
Pentium III Computer	**X**		
Keyboard	**X**		
Mouse	**X**		
Software			
Windows OS		**X**	
Spreadsheet		**X**	

9.5. Tractability recording requirement

Test	Software Configuration
Test 1	Interface Design, Requirements, Specification for Control System
Test 2	Interface Design, Requirements, Spec. for Signal Analyzer System
Test 3	Interface Design, Requirements, Spec. for Interface Comparison System
Test 4	Interactive, Requirement, Spec.SensorSub-system
Test 5	Interactive, Interface Design, Requirement, Specification for Logic Sub-system
Test 6	Requirements, Specification for whole Sub-system.
Test 7	Reliability, Requirement, Specification, Executing Correction for whole system
Test 8	Special Requirement for user's request

II. Static Verification:

~~**9.6. Inspection Checklist**~~

Fault Class	Inspection Check	Code
Data Fault	Is the Sensor Database Created?	1.1
	Have all constants been named?	1.2
	Isthe Sensor data updating consistently?	1.3
	Does every sensor data match the updates	1.4
Control Fault	Are the intervention and monitor functions set correctly?	2.1
	Is the error message showing on the screen correct?	2.2
	Is the information gain from searching correct?	2.3
Input/output Fault	Are all input variables used?	3.1
	Are the error message and correct parameter outputs showing	3.2
	on this screen appropriate	
Interface Fault	Is every conponent on the frame arranged appropriately?	4.1
	Are the functions shown on the menu efficient?	4.2
	Is the menu designed for user convenience?	4.3

~~**9.6.1. Inspection Report Table**~~

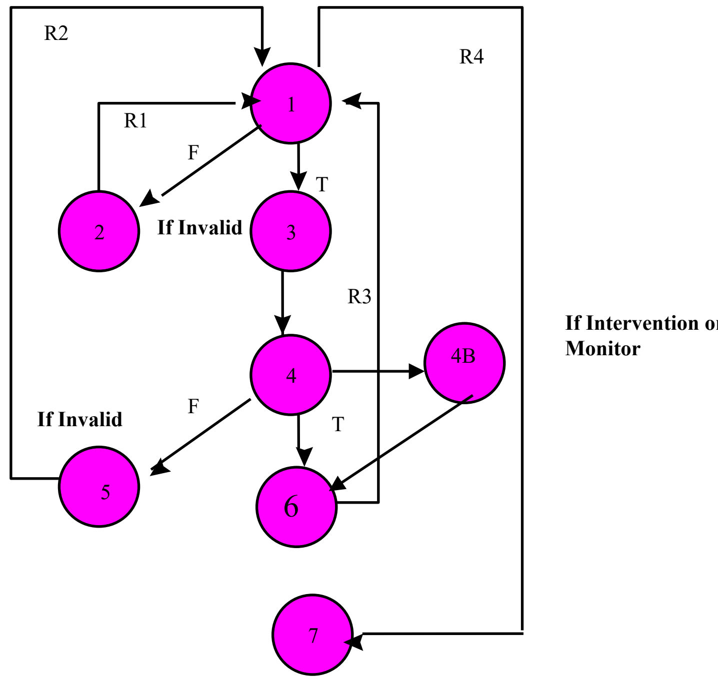
Fault Class	Code	Test Results		If No, Description of error
		Yes	No	
Data Fault	1.1			
	1.2			
	1.3			
	1.4			
Control Fault	2.1			
	2.2			
	2.3			
Input/output Fault	3.1			
	3.2			
Interface Fault	4.1			
	4.2			
	4.3			

9.7 Dynamic Testing / Defect Testing

9.7.1 Black Box Testing

Test	Input	Output
Test 1	Start the program	Show a main frame with file and help menu bars
Test 2	Choose System & Click Submit	Show intervention or monitor selections
	Button in main frame	OK, Cancel and Clear buttons
Test 3	Choose a SubSystem	Show a SubSys. frame w/intervention-monitor
	selected frame and click OK	
Test 4	Do the Test 1, Test 2, Test 3	Show a System corresponding the System we have
	Input again	chosen in the main frame
Test 5	Start the program, go to the System	Show a frame with text area allows you to enter the
	menu, choose intervention, enter	Time delay number. With Submit, Clear buttons.
	timing delay.	
Test 6	User will type in letters	Show a frame with an error message, letters are
		Invalid. Enter numbers.
Test 7	Do Test 4 again	Show intervention changes in the monitor
		Mode from action given.
Test 8	For user to test special options	

9.7.2 Structural (White Box) Testing



Begin
Procedure

1.Do While

Main Frame: Select the system

if invalid
2. Error message, back to Main

else

3. Choose Intervention or Monitor mode
4. If Intervention
or
4B. Monitor mode

Select option or monitor system functionality
if invalid selection

5. Error message, back to Main Frame
else

6. Select different system, show on the screen
back to the main frame

7. End do
Exit

9.7.3 Cyclomatic Complexity

V(G) = Number of regions (4)
V(G) = E(11)-N(8(or statement)) + 1 = (4)
V(G) = P+1 = (4)

G = Flow Graph
E = Edges
P = Predicate Nodes
N = Graph Nodes

9.7.4 Independent Path:

Path 1: 1,3,4,6,1,7
Path 2: 1, 2, 1, 7
Path 3: 1, 3, 4, 5, 1, 7
Path 4: 1, 7

Testing Design

In addition to the tests designed above, two more tests designs are shown below.

T9 (Testing 9): Select the Logic Subsystem and click the button for that system. Select Intervention to set the time delay. Enter letters for the time delay. An error message will appear saying that a number of seconds must be entered for the time delay.

T10 (Testing 10): Click the Exit from the file menu bar in the main frame. The program exits.

Path	Test	Input	Output
1	T1, T2, T3	Start the program	Show a main frame with file and help menu bars
		Choose system & Click Submit	Show intervention or monitor selections
	T4	Button in main frame	OK, Cancel and Clear buttons
		Choose a subsystem selected	Show Subsystem frame with intervene
	T7	frame and click OK	Or monitor
2	T6	Type in letters for time delay	Show a frame with an error message, a number
		In Logic System	Must be entered
3	T9	Start the program	Show a frame with an error message, a number
		Choose System & Click Submit	Must be entered
		Button in main frame	
		Choose intervention and	
		Enter invalid time delay	
4	T10	Main Frame, click exit	Program exits
		from the file menu bar	

10 USER'S MANUAL

WARNING

This manual is required to go through a series of in-progress reviews, and a verification effort. Until final acceptance by the customer and initial signoff by the contractor it is not to be used without assistance from engineering. Failure to do so could result in damage to equipment or injury to personnel. This warning will be removed before final publication.

10.1 INSTALLATION AND SETUP

Installation and setup will be performed as part of the initial contract. Since this is a complete self-contained system of computer hardware and software, the user will not be required to perform installation and setup.

10.2 OVERVIEW

This section provides the user with operational theory, instructions for use, and troubleshooting help.

10.3 FUNCTIONALITY

10.3.1 General System Operation.

The installed software is meant to control the flow of traffic through use of real-time traffic flow data, and timing algorithms. A model channel creates a continuous simulation, which provides an illustration of optimum performance based on the number and types of vehicles on the road. Through the use of simulation the user can see how traffic should be moving, so that a comparison can be made on this movement. This simulation is used to predict traffic problems and provide possible solutions before the problems occur. The solutions may involve changing the length of cycles, or switching to an alternate timing algorithm.

10.3.2 General Data Analysis.

Quantitative traffic data is analyzed for trends, and is used to build on the basic data obtained in the traffic study. When the system contains one month (30 days) of data, the database is the source of information on which future decisions are based. Though only one stoplight is being installed, traffic data will be collected from the entire city. Some of the data will be explicit, and from that data inferences may be drawn. Only logical inferences may be used. Inferential data conforming to the schema that if a vehicle is detected in a certain place, it was previously in another place. There are very few choices on where that previous place was. For example it may be a parking lot, it may be a street, or it may be a group of streets. In this way the collected data can be enhanced without adding more hardware to the streets.

10.3.3 Control Room Operation.

The Control Room is the nerve center of the system. The operator observes and monitors the movement of traffic. He/she is there to provide continuity and problem solving expertise. The operator monitors the system hardware for indicated faults, and the movement of traffic for interruptions. They maintain contact with the police by monitoring the police radio frequencies, and communicate with them when necessary. The police can be called upon to investigate the cause of traffic flow anomalies, and recommend solutions. In an emergency, the police may be called upon to manually direct traffic.

10.3.4 Catastrophic System Shut-Down

In case the system has been shut down the following steps need to be taken to re-boot the system.

Step 1. Ensure the power switches are turned off on each Computer Terminal and the Server.

Step 2. Ensure external inputs are switched off.

Step 3. Turn on the server, and the Main Console terminal.

- Monitor the messages on the screen to ensure normal boot up.
- Synchronize the system clock with Coordinated Universal Time by connecting to the Navel Observatory master clock.
- Once the clock is set isolate the system from the internet and run a virus check.

WARNING

If any leg of the system fails to synchronize, disconnect the external inputs and try it again. If it still fails to synchronize switch them on one at a time until the bad leg is isolated. Leave the bad leg isolated and call a technician. The isolated leg will remain in fail-safe mode.

Step 4. When the virus check is finished running, boot-up the other Computer Terminals.

- Ensure the operating system is synchronized in each terminal.
- Switch on the external inputs and check each leg of the system to ensure synchronization.

WARNING

If any error messages appear in step 5 perform the indicated actions. If the faults will not clear you should call a technician.

Step 5. At the Main Console type the following command and hit enter:

<start_Traffic_sys>

- Observe the onscreen messages to ensure all legs of the system are operating together.
- Observe the system for five minutes for stability.

- Run a series of traffic simulations and ensure there are no error messages.

Step 6. Reconnecting the control system to the traffic lights.

- At the Main Console cycle the fail-safe switch to off, and then on.
- The red indicator light flashes for approximately 60 seconds and goes off.
- The amber indicator light comes on steady for approximately 120 seconds.
- The green indicator light comes on showing that all the traffic lights that are connected to the system are operating properly.

Step 7. As data are gathered the system will catch up and get the traffic moving again. Close monitoring of the system is indicated until the crisis is over and traffic is flowing normally.

10.4 TROUBLESHOOTING

There are only two modes of failure: software and hardware.

10.4.1 Hardware Failure. The software will identify and locate a hardware failure. After a failure is detected an algorithm will take one of three actions 1) redirect lights to the fail-safe condition (red flashers), 2) notify the operator to call in a technician, 3) prompt the operator to alert Police to redirect traffic accordingly.

10.4.2 Software Failure. For software failures, the software will identify the node experiencing the failure, and what type of failure it is. A software failure is identified in several ways: 1) A catastrophic system crash, 2) erratic selection of the control algorithm, 3) loss of synchronization between the system elements. A loss of synchronization could be fixed by rebooting the leg experiencing the malfunction.

10.5 ALPHABETICAL INDEX **Paragraph**