California AHMCT Program
University of California at Davis
California Department of Transportation

OVERVIEW OF CALTRANS DISTRICT 10 IRIS DEMONSTRATION DESIGN*

Michael T. Darter[1], Stephen M. Donecker[1],
Kin S. Yen[1], Bahram Ravani[1], &

Ty A. Lasky[1], Principal Investigator

AHMCT Research Report
UCD-ARR-07-12-31-02

Interim Report of Contract IA 65A0210 - Task Order 06-22

December 31st, 2007

Affiliations:
1. AHMCT Research Center, Department of Mechanical & Aeronautical Engineering, University of California, Davis, CA 95616-5294

* This report has been prepared in cooperation with the State of California, Business and Transportation Agency, Department of Transportation and is based on work supported by Contract Number IA 65A0210 - Task Order 06-22 through the Advanced Highway Maintenance and Construction Technology Research Center at the University of California at Davis.

Technical Documentation Page

1. Report No. F/CA/RI-2006/10	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Overview of Caltrans District 10 IRIS Demonstration Design		5. Report Date December 31st, 2007 6. Performing Organization Code
7. Author(s): Michael T. Darter, Kin S. Yen, Stephen Donecker, Bahram Ravani, & Ty A. Lasky		8. Performing Organization Report No. UCD-ARR-07-12-31-02
9. Performing Organization Name and Address AHMCT Research Center UCD Dept of Mechanical & Aeronautical Engineering Davis, California 95616-5294		10. Work Unit No. (TRAIS) 11. Contract or Grant IA 65A0210 - Task Order 06-22
12. Sponsoring Agency Name and Address California Department of Transportation Division of Research and Innovation 1127 O Street Sacramento, CA 94273-0001		13. Type of Report and Period Covered Interim Report October 2005 - June 2008 14. Sponsoring Agency Code
15. Supplementary Notes		

16. Abstract

This document describes Task 6, "Overview of Caltrans District 10 IRIS Demonstration Design," within the Open ATMS multi-year research project undertaken by the Advanced Highway Maintenance & Construction Technology (AHMCT) Research Center at the University of California, Davis. The Open ATMS project is implementing an open-source Advanced Transportation/Traffic Management System (ATMS) within the California State Department of Transportation (Caltrans) District 10 (D10) Transportation Management Center (TMC). This document is an overview of the District 10 Intelligent Roadway Information System (IRIS) demonstration design.

17. Key Words ATMS, Open-Source Software (OSS), TMC, Remote Weather Information System / Roadway Weather Information System (RWIS), Highway operations, IRIS		18. Distribution Statement No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.	
20. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 66	22. Price

Form DOT F 1700.7 (872) Reproduction of completed page authorized

(PF V2.1, 6/30/92)

Abstract

This document describes Task 6, "Overview of Caltrans District 10 IRIS Demonstration Design," within the Open ATMS multi-year research project undertaken by the Advanced Highway Maintenance & Construction Technology (AHMCT) Research Center at the University of California, Davis. The Open ATMS project is implementing an open-source Advanced Transportation/Traffic Management System (ATMS) within the California State Department of Transportation (Caltrans) District 10 (D10) Transportation Management Center (TMC). This document is an overview of the District 10 IRIS demonstration design.

Table of Contents

Abstract **iii**

Table of Contents **viii**

List of Figures **ix**

Disclaimer/Disclosure **xi**

Acronyms and Abbreviations **xiii**

Acknowledgments **xvii**

1 Introduction **1**

1.1 Document Purpose 1

1.2 System Overview 1

1.3 Acronyms and Abbreviations 2

1.4 References 2

1.5 Overview 3

2 Traffic Monitoring Stations **7**

2.1 Purpose and General Description 7

2.2 Architecture 7

2.3 Hardware Utilization 8

2.4 Concept of Execution 8

2.5 Interface Design 8

2.6 Module Detailed Design . . . 9
2.7 Security . . . 9
2.8 Error and Exception Handling . . . 10
2.9 Requirements Traceability . . . 10

3 Changeable Message Signs **11**
3.1 Purpose and General Description . . . 11
3.2 Architecture . . . 11
3.3 Hardware Utilization . . . 12
3.4 Concept of Execution . . . 13
3.5 Interface Design . . . 13
3.6 Module Detailed Design . . . 15
3.7 Security . . . 15
3.8 Error and Exception Handling . . . 15
3.9 Requirements Traceability . . . 15

4 Vehicle Incident Mapping **17**
4.1 Purpose and General Description . . . 17
4.2 Architecture . . . 17
4.3 Hardware Utilization . . . 18
4.4 Concept of Execution . . . 18
4.5 Interface Design . . . 18
4.6 Module Detailed Design . . . 19
4.7 Security . . . 19
4.8 Error and Exception Handling . . . 19
4.9 Requirements Traceability . . . 19

5 Video Monitoring and Control **21**
5.1 Purpose and General Description . . . 21
5.2 Architecture . . . 21
5.3 Hardware Utilization . . . 23

5.4 Concept of Execution 23
5.5 Interface Design 24
5.5.1 Internal Interfaces 24
5.5.2 External Interfaces 25
5.6 Module Detailed Design 25
5.7 Security 25
5.8 Error and Exception Handling 25
5.9 Requirements Traceability 25

6 Ramp Metering 27
6.1 Purpose and General Description 27
6.2 Architecture 27
6.3 Hardware Utilization 28
6.4 Concept of Execution 28
6.5 Interface Design 28
6.6 Module Detailed Design 29
6.7 Security 29
6.8 Error and Exception Handling 29
6.9 Requirements Traceability 29

7 Weather Stations and Sensors 31
7.1 Purpose and General Description 31
7.2 Architecture 31
7.3 Hardware Utilization 32
7.4 Concept of Execution 32
7.5 Interface Design 33
7.6 Module Detailed Design 33
7.7 Security 33
7.8 Error and Exception Handling 33
7.9 Requirements Traceability 34

8 Event Logging **35**

8.1 Purpose and General Description 35

8.2 Architecture 35

8.3 Hardware Utilization 36

8.4 Concept of Execution 36

8.5 Interface Design 36

8.6 Module Detailed Design 37

8.7 Security 37

8.8 Error and Exception Handling 37

8.9 Requirements Traceability 37

9 Highway Advisory Radio and Extinguishable Message Signs **39**

9.1 Purpose and General Description 39

10 Intersection Traffic Signals **41**

10.1 Purpose and General Description 41

References **44**

A District 10 InfoTek Wizard GSM Modem **45**

List of Figures

1.1 IRIS Server class diagram 4
1.2 IRIS Server class diagram, hardware device communication 5
1.3 IRIS Server class diagram, hardware device driver 6

Disclaimer/Disclosure

The research reported herein was performed as part of the Advanced Highway Maintenance & Construction Technology (AHMCT) Research Center, within the Department of Mechanical and Aeronautical Engineering at the University of California Davis, and the Division of Research and Innovation at the California Department of Transportation. It is evolutionary and voluntary. It is a cooperative venture of local, State and Federal governments and universities.

The contents of this report reflect the views of the authors who are responsible for the facts and the accuracy of the data presented herein. The contents do not necessarily reflect the official views or policies of the State of California, the Federal Highway Administration, or the University of California. This report does not constitute a standard, specification, or regulation.

Acronyms and Abbreviations

Acronyms used within this document are defined below.

AHMCT	Advanced Highway Maintenance & Construction Technology
ATMS	Advanced Transportation/Traffic Management System
Caltrans	California State Department of Transportation
CAWS	California Automated Warning System
CCIT	California Center for Innovative Transportation
CHP	California Highway Patrol
CMS	Changeable/Dynamic Message Sign
CSU/DSU	Channel Service Unit/Data Service Unit
CTNet	Caltrans Central Signal Control System
D10	District 10
DRI	Division of Research and Innovation
DSL	Digital Subscriber Line
EDGE	Enhanced Data Rates for GSM Evolution
EIA	Electronic Industries Alliance
ESRI	Environmental Systems Research Institute
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
HAR	Highway Advisory Radio
HTTP	HyperText Transfer Protocol

IP	Internet Protocol
IRIS	Intelligent Roadway Information System
ITS	Intelligent Transportation System
ITS	Intelligent Transportation Systems
JPEG	Joint Photographic Experts Group
M170	Model 170
Mn/DOT	Minnesota Department of Transportation
M-JPEG	Motion JPEG
MPH	Miles Per Hour
NTCIP	National Transportation Communications for ITS Protocol
OSS	Open-Source Software
PC	Personal Computer
PROM	Programmable Read-Only Memory
PTZ	Pan Tilt Zoom
RS-232	EIA serial communications standard
RWIS	Remote Weather Information System / Roadway Weather Information System
SDRMS	San Diego Ramp Metering System
SMS	Short Message Service
SONAR	Simple Object Notification And Replication
SQL	Structured Query Language
SV170	SignView M170 Programmable Read-Only Memory (PROM) chip
TAG	Technical Advisory Group
TCP/IP	Transmission Control Protocol / Internet Protocol
TLS	Transport Layer Security
TMC	Transportation Management Center
TMS	Transportation Management System

UDP/IP	User Datagram Protocol / Internet Protocol
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
VPN	Virtual Private Network
XML	eXtensible Markup Language

Acknowledgments

The authors thank the California State Department of Transportation for their support; in particular, the guidance and review provided by the Open ATMS project team and Technical Advisory Group. The authors also acknowledge the dedicated efforts of the AHMCT development team members. Special thanks to Caltrans Headquarters Traffic Operations (Stan Slavin and Alan Benson), Caltrans Division of Research and Innovation (DRI) (Fred Yazdan, Roya Hassas, and Sean Campbell), Caltrans District 10 Traffic Operations (Clint Gregory, Veronica Cipponeri, Arlene Cordero, Mohammad Battah, Joe Silvey, and James Collins), Caltrans District 12 Traffic Operations (Omid Segal and Morteza Fahrtash), Larry Tjoelker (Caltrans Headquarters), and Minnesota Department of Transportation (Mn/DOT) (James Kranig, Doug Lau, Timothy Johnson, and Ralph Adair).

Chapter 1

Introduction

1.1 Document Purpose

This document is a brief overview of the Caltrans D10 IRIS demonstration design. The design of each functional area within the project is covered in subsequent chapters. These findings are the result of a single task within a multi-year research project undertaken by the Advanced Highway Maintenance & Construction Technology (AHMCT) Research Center at the University of California, Davis[1].

1.2 System Overview

The Minnesota Department of Transportation (Mn/DOT) Intelligent Roadway Information System (IRIS) Advanced Transportation/Traffic Management System (ATMS) provides system capabilities which are organized by functional area. The prioritized list below identifies functional areas that are being implemented within Caltrans D10. Some of these functional areas require more customization than others. For a more detailed description of IRIS and the Open ATMS project, see Section 1.4 on the following page. The following prioritized list of desired system functionality was previously developed by the project's Technical Advisory Group (TAG) [3]:

1. Traffic Monitoring Stations,
2. Changeable Message Signs and Incident Mapping,
3. Video Monitoring,
4. Ramp Meters,

[1]For AHMCT see http://ahmct.ucdavis.edu

5. Weather Stations,
6. Video Control,
7. Event Logging,
8. Highway Advisory Radio and Extinguishable Message Signs (*deferred for later development*),
9. Intersection Traffic Signals (*deferred for later development*).

1.3 Acronyms and Abbreviations

For a complete list of acronyms and abbreviations used in this document, see page xiii. In addition, the following terms are used:

- IRIS: indicates the version of the IRIS server and client developed by Mn/DOT,
- D10 IRIS: indicates the software developed by Mn/DOT that is customized and extended by AHMCT for use within Caltrans D10. Specific features critical for Caltrans have been added, including support for San Diego Ramp Metering System (SDRMS) and SignView.

1.4 References

This document cites specific requirements by number—*these requirements are defined within the Task 4 report* [3]. For additional insight, the reader may consult the documents listed below and in the References section on page 44:

- Intelligent Roadway Information System (IRIS) As Built System Design Document [6],
- Project Task 2 report, ATMS literature review [1],
- Project Task 3 report, D10 operations and equipment [2],
- Project Task 4 report, functional requirements [3],
- Project Task 5 report, IRIS review [4],
- Project Final report [5],
- Caltrans Systems Engineering Document Template Tree [9],
- Caltrans Systems Engineering Methodology, Interface Detailed Design [10].

- Caltrans Systems Engineering Methodology, System Architecture [11],
- Caltrans Systems Engineering Methodology, Software Detailed Design [12],
- IRIS JavaDoc source code documentation [7],
- IRIS source code documentation: some modules contain additional documentation. See the doc sub-directory within each module's repository [8].

1.5 Overview

The structure of this document is based on the Caltrans SP-061 Software Detailed Design template[12]. The following chapters provide an overview of the Caltrans D10 IRIS demonstration design, by prioritized functional area. The Caltrans Systems Engineering Methodology Software Detailed Design template was used [12]. Throughout the report, the reader is referred to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6 for discussions related to the IRIS class structure.

This document was developed using software source code, project documentation, research reports, software and hardware specifications, software documentation, field and laboratory data, and guidance and information gained through collaboration with the project engineers and staff from the TAG, D10, and Mn/DOT.

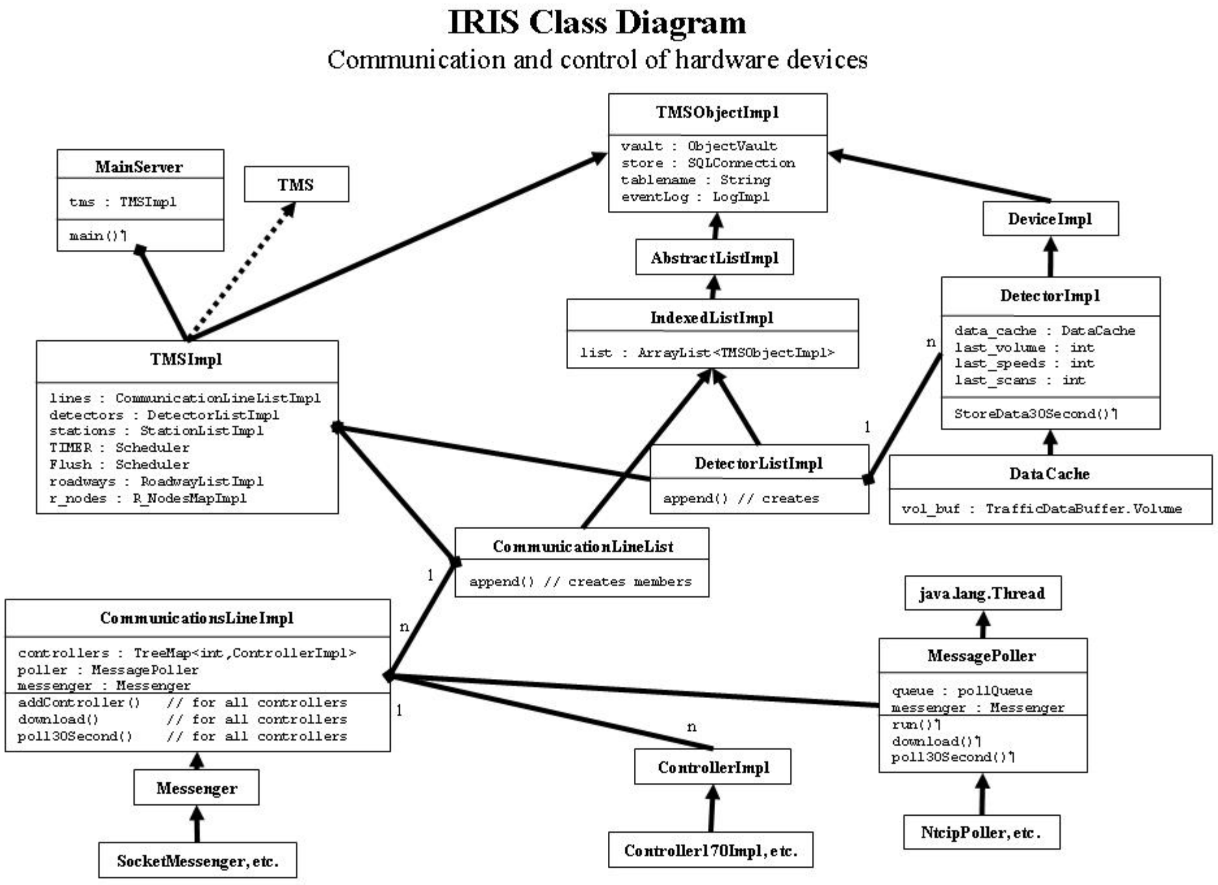

Figure 1.1: IRIS Server class diagram

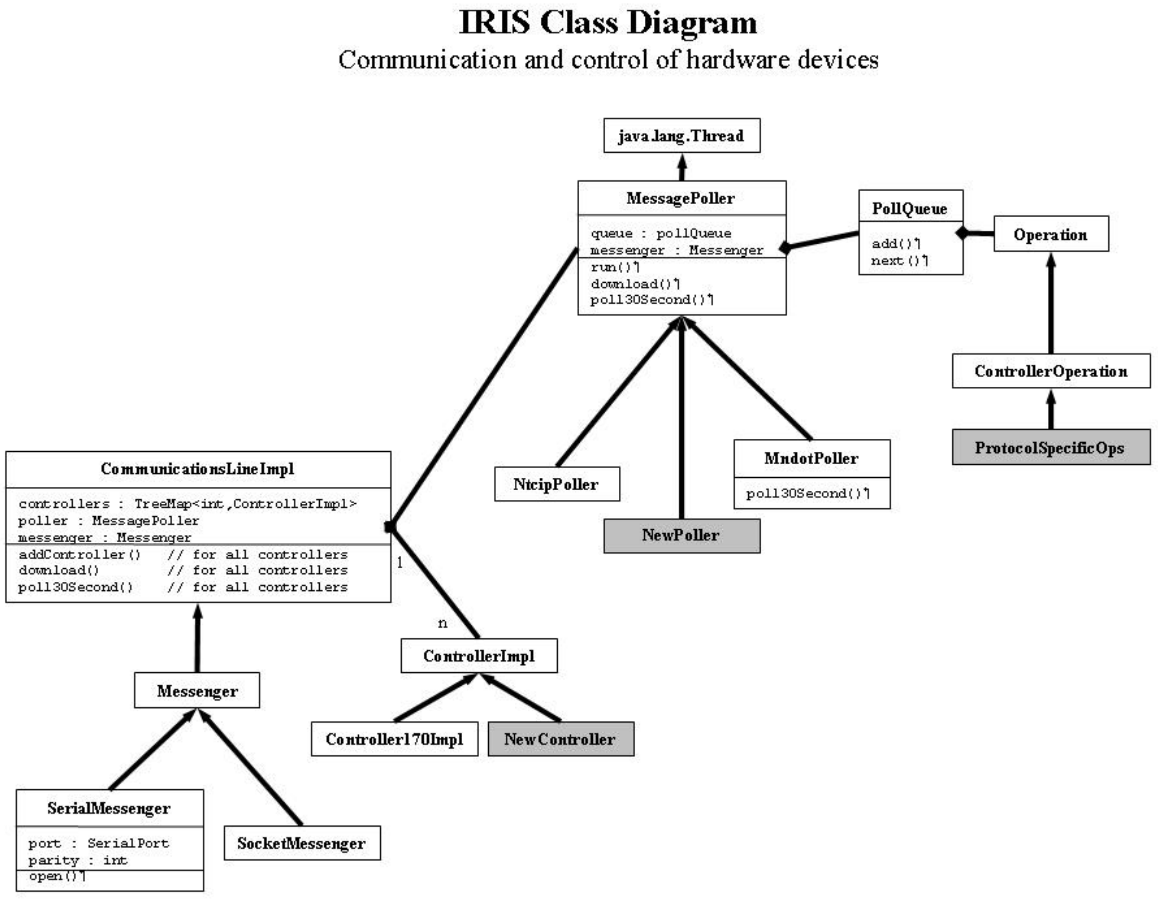

Figure 1.2: IRIS Server class diagram, hardware device communication

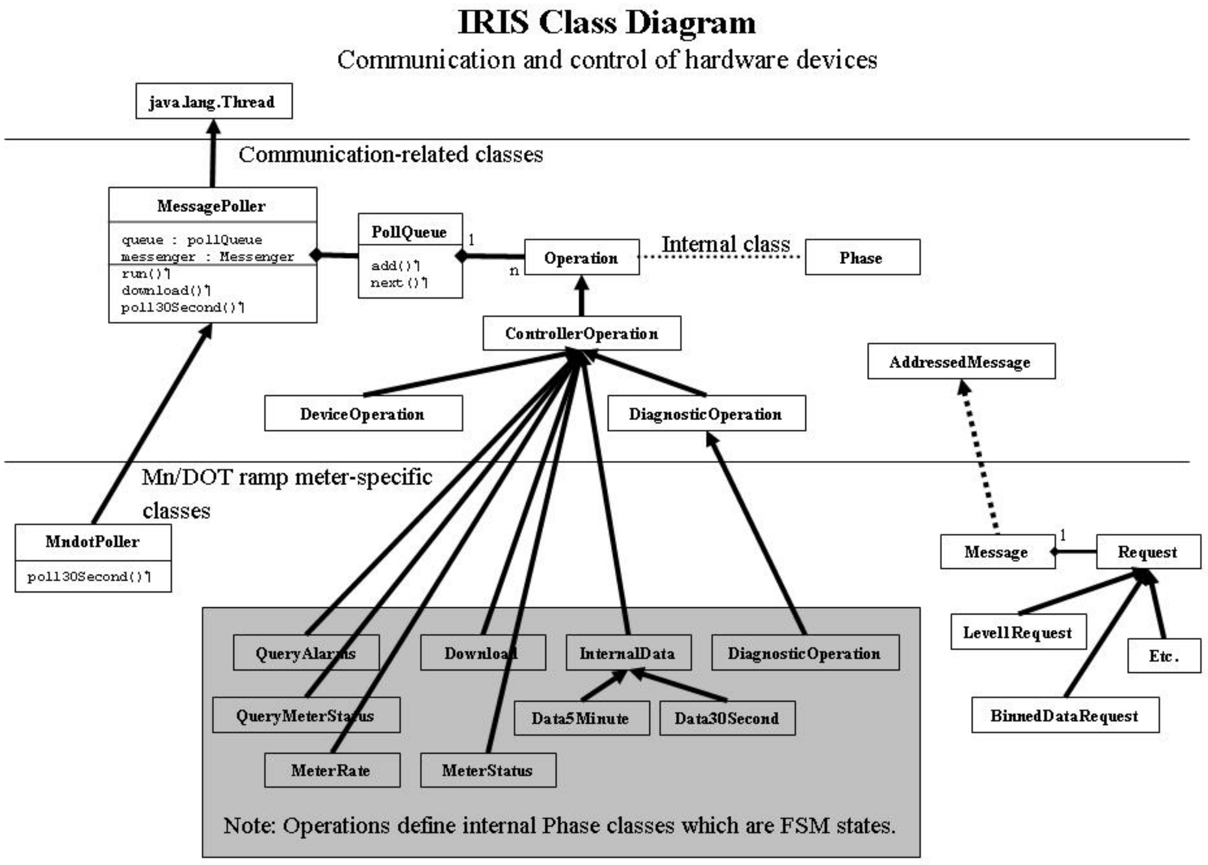

Figure 1.3: IRIS Server class diagram, hardware device driver

Chapter 2

Traffic Monitoring Stations

2.1 Purpose and General Description

The Traffic Monitoring functional area has been identified as priority 1 by the TAG. The traffic monitoring module provides a visual display of real-time traffic data for TMC operators. Real-time traffic is displayed on a specific map layer within the IRIS client (Requirement 1,2). In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

2.2 Architecture

Relationships among primary structural elements within the module are:

- Existing D10 traffic infrastructure: this includes traffic detectors, controllers, communications infrastructure, and router. Controllers include the Caltrans Model 170 (M170) and InfoTek Wizard (see Appendix A on page 45). This hardware and software forwards real-time traffic data to the Traffic Server (Requirement 3)[2, 3].

- Traffic Server application: this application will be developed to: receive traffic data from the existing D10 infrastructure, re-format and queue the traffic data, provide a connection for the IRIS server, and forward the traffic data over this connection (Requirement 4).

- IRIS server: connects to the Traffic Server and periodically requests traffic data. This data is provided to the IRIS client to display on a map.

- IRIS client mapping functionality: the client displays real-time traffic data on a traffic map.

2.3 Hardware Utilization

Existing hardware will be used and no specialized hardware is required. Memory, processing, and disk storage use are expected to be within the bounds of that provided by the existing IRIS system (Requirement 10).

2.4 Concept of Execution

The structural elements identified in the Architecture section operate as discrete applications and communicate using Internet Protocol (IP). The traffic server application uses a number of threads internally to: read inbound traffic data, wait for outbound IRIS server connections, and forward traffic data to a connected server.

2.5 Interface Design

Interface information for IRIS classes and interfaces is available in the JavaDoc [7].

External Interfaces for the Traffic Server

The Traffic Server is a stand-alone application that receives User Datagram Protocol / Internet Protocol (UDP/IP) data from D10 middleware. This traffic data is from both loop and microwave detector sources (Requirement 6). The data syntax is defined by D10 and is line-oriented, with one station per location per line. The field delimiter is a comma. Speed is in Miles Per Hour (MPH) and occupancy is in tenths of a percent. The date field uses 24-hour time. The line syntax is:

```
<Unique Site name>,<number of lanes>,
   <volume lane 1>, <speed lane 1>, <occupancy lane 1>,
      <repeat volume, speed, occupancy for each lane>,
         <date time>
```

Sample Data:

```
1013010,2,5,76,30,6,69,60,2007-11-01 10:16:39
```

The Traffic Server's outbound interface with IRIS is the Wavetronix SS105 SmartSensor Data Protocol [13], which is supported by IRIS (Requirement 12).

Internal Interfaces for the Traffic Server

The namespace `us.ca.state.dot.tms.trafserver` will contain developed classes and interfaces, including those listed below.

- MainServer: contains the entry point main(), TrafficListener, TrafficQueue, and TrafficServer. A properties file is read on start-up.
- TrafficListener: responsible for opening a connection with the inbound traffic data arriving from the D10 middleware and router. Received data is parsed and queued in a TrafficQueue. Contained by MainServer.
- TrafficQueue: receives traffic station data from the TrafficListener. Contained by MainServer. Items are de-queued by TrafficServer.
- TrafficServer: listens for connections from the IRIS server. Responses to inbound Wavetronix protocol requests contain de-queued D10 traffic data in Wavetronix format. Contained by MainServer.

IRIS Client Map Interface

The real-time traffic map displayed within the IRIS client is an Environmental Systems Research Institute (ESRI) shape file associated with the gpoly layer. It contains embedded meta-data associating polygons with station identifiers. Station identifiers within IRIS are implemented as text strings.

2.6 Module Detailed Design

For detailed design information see the Traffic Server JavaDoc[1].

2.7 Security

The following security assumptions are made:

- Network access is limited and closely defined for the machine the Traffic Server application is executing on (for example, using security parameters in hosts.allow and hosts.deny and/or firewall rules).
- Firewall rules on the machine the Traffic Server application is executing on are used to restrict inbound and outbound network traffic to specific machines.
- The Traffic Server and existing traffic functionality will use existing IRIS and D10 security features (Requirement 17).

[1]For JavaDoc see `http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/`

2.8 Error and Exception Handling

Standard Java exception handling will be used in the Traffic Server and is already used within the IRIS client and server (Requirement 13). Error handling within the Traffic Server will be through log files.

2.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

Chapter 3

Changeable Message Signs

3.1 Purpose and General Description

Changeable/Dynamic Message Sign (CMS) functionality has been identified as priority 2 by the TAG. D10 IRIS will interface with existing CMS which are connected to the TMC through:

1. The Cingular wireless Virtual Private Network (VPN) by the InfoTek Wizards (see Appendix A on page 45).
2. Dial-up lines and a Lantronix terminal server [2, 3].

All CMS are controlled with M170 controllers using the Caltrans SignView protocol (Requirement 25). IRIS uses the National Transportation Communications for ITS Protocol (NTCIP) protocol (Requirement 20). CMS messages originate from these sources:

- California Automated Warning System (CAWS): automatically generated as a function of sensor inputs (Requirement 28),
- TMC operators: either manually entered or selected from a message library (Requirement 27),
- IRIS-generated: travel-time related messages. The IRIS server calculates travel times[6], which are displayed on CMS[1]. Use of IRIS built-in travel time calculations outside of Minnesota will require route definitions for the areas of interest.

In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

3.2 Architecture

Relationships among primary structural elements are listed below. The primary elements to be developed are the IRIS CAWS Client and CMS Server.

[1]See the IRIS source code documentation iris/docs/travel_time.html.

- CAWS System: the portion of the D10 middleware system that automatically generates CMS messages based on inputs from sensors.
- D10 IRIS Client: used by TMC operators to monitor and control CMS.
- IRIS CAWS Client: an application that interfaces the CAWS system with IRIS. Conceptually, this application acts as a virtual IRIS client, forwarding CAWS generated CMS messages to the IRIS server. Additionally, if non-IRIS generated travel time messages are desired (e.g. originating from California Center for Innovative Transportation (CCIT) or other sources), the IRIS CAWS client would be responsible for reading and forwarding these CMS messages to the IRIS server. Multiple instances of the IRIS CAWS may be used for dedicated purposes. Use of IRIS built-in travel time calculations outside of Minnesota will require route definitions for the areas of interest.
- D10 IRIS Server: the hardware on which IRIS software is executing.
- D10 IRIS: software executing on the server which communicates with the CMS Server using IRIS Hardware Drivers.
- IRIS Hardware Drivers: the portion of the IRIS server responsible for communicating with CMS controllers using the NTCIP protocol. This collection of classes and interfaces is in the `us.mn.state.dot.tms.comm` namespace [4]. See Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.
- CMS Server: a stand-alone application developed to interface IRIS with SignView controllers in the field (Requirement 24). It provides a Transmission Control Protocol / Internet Protocol (TCP/IP) server for IRIS with inbound NTCIP messages which are converted to (possibly multiple) outbound SignView messages and forwarded to M170 controllers in the field. In the other direction, SignView responses are converted to NTCIP messages and returned to IRIS.
- Communications: managed by the CMS Server and consists of either a dial-up line (Requirement 21) and Lantronix terminal server (Requirement 26) or wireless VPN and InfoTek Wizard (Requirement 22).
- M170 Controller: contain SignView M170 PROM chip (SV170), connected to the CMS, communicate with the CMS Server using the SignView protocol.

3.3 Hardware Utilization

Existing hardware will be used, including in-field InfoTek Wizards, M170s, and communication lines. IRIS server memory, processing and storage use are expected to be within the bounds of that provided by the existing D10 IRIS system. In-field M170 controllers will contain SV170 PROM chips. The CMS Server will interface with in-field controllers on dial-up lines using a Lantronix terminal server (Requirement 26).

3.4 Concept of Execution

Structural elements are shown in the Architecture Section. The order listed below indicates the time sequence of events:

CMS Messages Initiated by TMC Operators

1. D10 IRIS Client: communicates with the IRIS Server and is used by the TMC operators to monitor existing messages and generate new messages.
2. D10 IRIS: receives client message requests and interfaces with the CMS Server using IRIS Hardware Drivers.
3. IRIS Hardware Drivers: communicates directly with the CMS Server using NTCIP over TCP/IP.
4. CMS Server: translates messages between CMS protocols, sends/receives messages to field controllers. CMS configuration information is used to determine if a specific CMS is located on a wireless network or on a dial-up line via the terminal server.
5. Communications: handled by the CMS Server on one end and a modem and M170 on the other end.
6. M170 Controller: receives and responds to SignView CMS messages from the CMS Server.

CMS Messages Initiated by CAWS

1. CAWS System: a sensor within the system triggers a CAWS event such as high wind speed. D10 middleware formulates an automated message for a specific CMS sign.
2. IRIS CAWS Client: receives an automated message from CAWS and communicates with the IRIS to place the message on the specified CMS.
3. D10 IRIS: continue with Step 2 above in "CMS Messages Initiated by TMC Operators".

3.5 Interface Design

Primary module components under development are the IRIS CAWS Client and CMS Server. External interfaces between components and internal interfaces within components are discussed below.

Internal Interfaces for IRIS CAWS Client

The namespace `us.ca.state.dot.tms.cawsclient` will contain the following developed classes and interfaces:

- Main: contains the entry point main(), CAWSListener, MessageQueue, and IRISConnection. A properties file is read on start-up.
- CAWSListener: contained by Main, subclass of Thread. Periodically reads a file (HyperText Transfer Protocol (HTTP)) containing CMS messages generated by the CAWS system. Read frequency, file Uniform Resource Locator (URL), and CMS characteristics are defined by the properties file. New messages and actions are instantiated and added to the MessageQueue.
- MessageQueue: contained by Main, contains subclassed Operations.
- IRISConnection: contained by Main, subclass of Thread. Waits for new subclassed Operation objects to show up in the MessageQueue. Uses the IRIS interface to execute each subclassed Operation. These will typically be new messages to add to CMS.
- Operations: represent work units to be handed off to the IRIS Server. They are contained by MessageQueue, created by CAWSListener, executed by IRISConnection.

Internal Interfaces for CMS Server

The namespace `us.ca.state.dot.tms.cmsserver` will contain the following developed classes and interfaces:

- MainServer: contains the entry point main(), IRISListener, MessageQueue, and ControllerConnection for each active CMS. A properties file is read on start-up that contains property information for each CMS. The syntax of the existing D10 middleware CMS configuration file will be used if possible.
- IRISListener: contained by MainServer and responsible for opening a connection with the IRIS server. NTCIP requests are received, converted to SignView messages, instantiated as new subclassed Operation objects and added to the CMS-specific MessageQueue.
- MessageQueue: contains subclassed Operation objects inserted by an IRISListener. Receives traffic station data from the TrafficListener, contained by MainServer. Items are de-queued by TrafficServer.
- ControllerConnection: contained by MainServer and corresponds to a connection with a single CMS controller. Extends Thread and waits for new Operation objects to appear in the MessageQueue.
- Operation: created by IRISListener objects and contained by MessageQueue for a specific CMS.

External Interfaces

- CAWS and IRIS CAWS Client: the IRIS CAWS Client receives inbound CMS messages from the CAWS system via a specific disk file. The file syntax is defined by the D10 middleware. HTTP is used to access the file.

- IRIS CAWS Client and IRIS: the IRIS CAWS Client will use the same communications mechanism with IRIS as other IRIS clients: Simple Object Notification And Replication (SONAR) over Transport Layer Security (TLS).
- IRIS Server and CMS Server: NTCIP over TCP/IP.
- CMS Server and InfoTek Wizards: SignView over TCP/IP.
- CMS Server and terminal servers: SignView over TCP/IP.

3.6 Module Detailed Design

For detailed design information see the JavaDoc[2].

3.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security and reliability features and policies (Requirement 32).
- Existing Caltrans communication between the field devices and the TMC will be used.
- If secure communication with field devices over land lines is desirable, a field controller should be developed that will provide:
  - Support for the SignView protocol (Requirement 25),
  - Interface with CMS,
  - Support encrypted TCP/IP communication with the TMC,
  - Interface with IRIS.

3.8 Error and Exception Handling

Standard Java exception handling will be used in this module and is already used within the IRIS client and server. Existing IRIS event logging functionality will be used. CMS- and SignView-specific events may be developed.

3.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

[2]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

Chapter 4

Vehicle Incident Mapping

4.1 Purpose and General Description

Vehicle and incident mapping has been identified as priority 2 by the TAG. D10 IRIS will use existing IRIS functionality (Requirement 44) to display incident data on the client map. Real-time California Highway Patrol (CHP) incident data will be used (Requirement 45). In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

4.2 Architecture

Relationships among primary structural elements are listed below. The primary elements to be developed are subclasses of existing IRIS incident classes specialized for D10 CHP incidents (Requirement 48).

- Incident data source: this is an eXtensible Markup Language (XML) data file available over a TCP/IP connection that is periodically updated by the CHP (Requirement 46,47). It contains detailed incident data. The URL is specified in the IRIS client properties file.
- IRIS Client Incident Map: incidents received from the data source are displayed on this map. Programmatically, this is an instance of IncidentLayer, which is contained by a MapPane, which is contained by a MapBean, which is contained within a MapTab in the IRIS client [4]. The IncidentLayer is a container for XmlClient.
- IncidentListModel: extends AbstractListModel, implements IncidentListener, in the `us.mn.state.dot.tms.client.incidents` namespace. Contained by IncidentTab in the client.
- Incident: an interface in the `us.mn.state.dot.tdxml` namespace. It will be implemented to represent a CHP incident. It will be named CHPIncident.
- IncidentException: extends Exception. May be used to represent CHP incident exceptions.

- XmlClient: implements Runnable. Contained by IncidentLayer. Subclasses read XML files with specific formats on an interval. Contains a list of DdsListener objects which are notified when new data arrives. A subclass will be implemented named CHPXmlClient.
- IncidentListener: an interface that extends DdsListener. Classes that implement it receive notifications from XmlClient when new incident data arrives.

4.3 Hardware Utilization

Existing hardware will be used. IRIS client and server memory, processing and storage use are expected to be within the bounds of that provided by the existing D10 IRIS system.

4.4 Concept of Execution

Structural elements are shown in the Architecture section. The order listed below indicates the time sequence of events:

1. Periodic read of incident data: a subclass of XmlClient is created by trafmap.IncidentLayer, which has a lifetime of the application. This is a thread that periodically reads (Requirement 46) the XML file (via method readXmlFile()). The file is parsed and new Incident objects are created and validated. This includes geographic coordinate transformation (Requirement 49). Listeners are notified that new Incidents have arrived.
2. Listener Notification: the object client.incidents.IncidentTab adds itself as a listener to the IncidentLayer instance and receives notification of new Incidents.
3. Client: the IncidentTab in the client contains an IncidentListModel, which is a list of current incidents.

4.5 Interface Design

For existing internal interfaces see the JavaDoc for the classes and interfaces listed above[1].

External Interfaces

The interface between D10 IRIS and the incident data is the XML incident file. The syntax is show here:

```
<State>
   <Center ID="BFCC">
      <Dispatch ID="BFCC">
         <Log ID="0789D0103">
```

[1]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

```
            <LogTime>"1/1/2008 6:20:07 PM"</LogTime>
            <LogType>"1122 - Traffic Collision - No Injuries"</LogType>
            <Location>"EB CLAIREMONT MESA BLVD ONR TO SB I805"</Location>
            <Area>"San Diego"</Area>
            <ThomasBrothers>"1248 1J"</ThomasBrothers>
            <TBXY>"6279748:1883182"</TBXY>
            <LogDetails>
               <details>
                  <DetailTime>" 7:12PM"</DetailTime>
                  <IncidentDetail>
                     "1039 AFN LAS COLINAS/SPOKE TO D"
                  </IncidentDetail>
               </details>
               <details>
                  <DetailTime>" 7:11PM"</DetailTime>
                  <IncidentDetail>
                     "15B BM 1192 TO LAS COLINAS / START BLOOD TECH"
                  </IncidentDetail>
               </details>
               ⋮
            </LogDetails>
         </Log>
         ⋮
      </Dispatch>
      ⋮
   </Center>
   ⋮
</State>
```

4.6 Module Detailed Design

For detailed design information see the JavaDoc[2].

4.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security and reliability features and policies (Requirement 50).

4.8 Error and Exception Handling

Standard Java exception handling will be used in this module and is already used within the IRIS client and server. The existing IncidentException class is expected to be used.

4.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

[2]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

Chapter 5

Video Monitoring and Control

5.1 Purpose and General Description

Video monitoring (priority 3) and control (priority 6) are both covered in this chapter. The video module provides a visual display of real-time video feeds for TMC operators. Video feeds are selectable on a specific map layer within the IRIS client or by way of a drop down list. If applicable, camera Pan Tilt Zoom (PTZ) can be controlled using on screen buttons or a Universal Serial Bus (USB) joystick. See the references for additional background material [2, 3, 4].

In light of IRIS capabilities, various hardware interfaces to existing camera networks are in a state of flux. Additionally, several digital network cameras are being procured for inclusion in the D10 camera network. The existing hardware configuration is as follows.

- City of Stockton Cameras: this network consists of Bosch cameras (analog) and Pelco cameras (analog) with the integral Bosch protocol converter board. PTZ is accomplished through the use of PTZ joysticks communicating with Bosch protocol hardware. Analog video is transmitted to the D10 TMC over several direct fiber connections.
- D10 Cameras: this network consists of several Cohu cameras (analog) which support the Pelco D protocol for PTZ control. In the field, these cameras are currently connected to Personal Computers (PCs) running Microsoft Windows XP. They capture the analog video and support remote PTZ over Digital Subscriber Line (DSL). Currently about 80 network video cameras are being purchased for implementation in D10 with anticipated use by IRIS TMC operators.

In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

5.2 Architecture

Primary structural elements within the module include:

D10 Video Network

- D10 Video Cameras: this includes the existing Cohu analog cameras.
- D10 VLC Video Servers: PC/Linux based VLC software video servers.
- D10 Communications: the DSL communication channels.
- D10 Network Video Cameras: these are newly procured IP-based network cameras, which support camera PTZ control and Motion JPEG (M-JPEG) video streaming through a web interface (Requirement 54,55).

City of Stockton Video Network

- City of Stockton Video Cameras: this includes the existing PTZ Bosch video cameras (analog), and the PTZ Pelco video cameras (analog) with integrated Bosch protocol converter board.
- City of Stockton Control: this includes the analog video switcher hardware, and the PTZ control hardware.
- City of Stockton Communications: the fiber optic communication channels between the City of Stockton and D10 TMC which are currently used for analog video feeds. It is anticipated that these channels will be converted to support networked video traffic.
- City of Stockton Axis Video Servers: these are the web based video servers that IRIS natively supports for serving existing analog video feeds.

D10 IRIS

- IRIS Client: used by TMC operators to monitor and control the video system.
- IRIS Server: the main IRIS server which communicates with all of the various systems.
- IRIS Back-end Video Server: the server that collects the video streams from the various cameras/video-servers using the IRIS video drivers, and makes them available to clients via a Tomcat/Apache web server.
- Video Drivers: the drivers that interface IRIS to the various web based HTTP video servers. This includes the native Axis video servers, VLC streaming video servers, and network video cameras.
- Protozoa Protocol Converter: the IRIS PTZ protocol converter. It converts the standard IRIS Pelco D control protocol to other popular protocols.

5.3 Hardware Utilization

Existing hardware will be utilized to the extent possible. However, since IRIS fundamentally requires that the server requests video streams from remote web-based video servers supporting M-JPEG streams, some additional hardware is needed. IRIS currently supports full compatibility to Axis video servers.

The City of Stockton network presently provides video feeds to all cameras through fiber into the D10 TMC. Video from eight cameras can be selected and channeled to D10 TMC displays. Instead of capturing the output from these fiber channels at D10, Axis servers will be procured and installed locally at the City of Stockton, and the digital streams will then be sent over the fiber to D10. In this configuration D10 TMC operators will have access to all Stockton cameras simultaneously, limited only by bandwidth. Control of these cameras will most likely be performed using existing hardware (Requirement 56). However, this issue is in flux, and will be clarified in subsequent design and reporting.

Current D10 cameras (analog) utilize remote PC boxes to implement a video server. This hardware will be reused and only the software (open-source) on these remote machines will be replaced. The selection of new video cameras will determine if they are directly compatible with IRIS or require software modifications, as in the case of the analog cameras.

Existing server hardware will be used and no additional hardware is required. Memory, processing, and disk utilization are expected to be within the bounds of that provided by the existing IRIS system (Requirement 53,59).

5.4 Concept of Execution

Structural elements are shown in the Architecture Section. The order listed below indicates the time sequence of events:

1. IRIS Client: communicates with the IRIS Server and is used by TMC operators to monitor and control various cameras in the deployed system.

2. IRIS Server: receives client video stream requests, PTZ commands, and interfaces with the IRIS Back-end Video Server and the Protozoa protocol converter.

3. IRIS Back-end Video Server: receives requests for specific video streams and utilizes various Video Drivers to make requests, and collect streams from the specific camera video servers. These streams are then relayed to one or more IRIS clients.

4. Protozoa Protocol Converter: receives native IRIS Pelco D PTZ protocol, and converts control commands to alternate protocols depending on camera.

5. Video Drivers: used by the Back-end Video Server to communicate and receive streams from the Axis Video Servers, D10 VLC Video Servers, and D10 Network Video Cameras using the HTTP protocol.

6. City of Stockton and D10 Communications: the TCP/IP network channel over which the Video Drivers send commands to request video streams from the Video Servers and Network Video Cameras.

7. Video Servers and Network Video Cameras: receive HTTP stream requests from the Backend Video Server and respond with streaming M-JPEG video (Requirement 58)

5.5 Interface Design

Primary module components under development are the VLC Video Server Driver, Network Video Camera Video Driver, Video Source Interface, and modifications to the Transportation Management System (TMS) database and Server Factory code to instantiate appropriate drivers based on camera configuration in the database. External and internal interface design is discussed below. For more detailed information about existing classes and interfaces, see the JavaDoc[1].

5.5.1 Internal Interfaces

D10 IRIS development will focus primarily on extending and customizing functionality within the IRIS server. Relevant video server classes and interfaces include:

- VideoServlet: abstract, extends HttpServlet, and is the base class for all video servlets in the video module.
- ImageServer: extends VideoServlet, and is the main thread for the still video server.
- StreamServer: extends VideoServlet, and is a servlet that responds to client requests for a video stream.
- ImageFactoryDispatcher: creates and distributes client streams.
- ServerFactory: responsible for creating VideoSource objects and verifying they are synchronized with the database.
- AbstractImageFactory: a subclass connects to an HTTP video stream from the stream server and notifies each listener following arrival of a new image.
- VideoSourceImageFactory: extends AbstractImageFactory, and interacts with a video source server to continually produce images until there are no more requests for images.
- VideoSource: interface class which defines the digital video sources.
- VideoSourceAxis: implements the VideoSource interface and encapsulates functionality of an Axis video stream server.
- VideoSourceVLC: implements the VideoSource interface and encapsulates functionality of a VLC video stream server.

[1]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

- VideoSourceIPCamera: implements the VideoSource interface and encapsulates functionality of a D10 network video camera.

5.5.2 External Interfaces

Interfaces between architecture elements include:

- Axis video server HTTP protocol,
- VLC video server HTTP protocol,
- D10 network video camera HTTP protocol,
- Pelco D PTZ control protocol.

5.6 Module Detailed Design

For detailed design information see the JavaDoc[2].

5.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security features (Requirement 53),
- Video servers and network video cameras support user names and passwords,
- Video server IP numbers and ports will be kept according to existing Caltrans policies,
- PTZ control will use existing D10 and City of Stockton security policies.

5.8 Error and Exception Handling

Standard Java exception handling will be used in the video monitoring and control modules and is already used within the IRIS client and server (Requirement 59,62).

5.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

[2]For JavaDoc see `http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/`

Chapter 6

Ramp Metering

6.1 Purpose and General Description

Ramp metering functionality has been identified as priority 4 by the TAG. D10 has two ramp meters and no mainline meters or freeway connector meters. A new D10 IRIS hardware driver will be developed to communicate with in-field ramp meters using the Caltrans San Diego Ramp Metering System (SDRMS) protocol. InfoTek Wizards (see Appendix A on page 45) will be used to communicate with the M170 controllers in the field. In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

6.2 Architecture

Relationships among primary structural elements are listed below.

- D10 IRIS Server: the hardware on which IRIS software is executing.
- D10 IRIS: software executing on the server which communicates with the M170 controllers in the field using IRIS hardware drivers.
- IRIS Hardware Drivers: the portion of the IRIS server responsible for communicating with hardware devices such as ramp meter controllers. This collection of classes and interfaces are in the `us.mn.state.dot.tms.comm` namespace [4]. Classes developed specifically to support the SDRMS protocol will interface with these driver classes.
- Wireless communication: infrastructure between the in-field InfoTek Wizard (see Appendix A on page 45) and D10 TMC[2, 3].
- In-field Hardware Devices: ramp meters connected to M170 controllers connected to InfoTek Wizards [2]. The InfoTek Wizard wirelessly connects the controller to the TMC. The M170 controller contains an SDRMS PROM chip.
- IRIS Client User Interface (UI): used by TMC operators to monitor and control ramp metering functionality.

6.3 Hardware Utilization

Existing hardware will be used. IRIS server memory, processing and storage use are expected to be within the bounds of that provided by the existing D10 IRIS system. In-field M170 controllers will contain SDRMS V7 PROM chips.

6.4 Concept of Execution

Structural elements are shown in the Architecture Section. The order listed below indicates the time sequence of events:

1. D10 IRIS: initiates events through its job queue, which includes downloading information from field controllers [4]. The server contains multiple CommunicationLineImpl objects, which contain a subclassed MessagePollerImpl object.
2. IRIS Hardware Drivers: the MessagePoller contains a PollQueue object which contains Operation objects which drive communication actions with device hardware.
3. IRIS Client UI: asynchronously accesses module state information on the server.

6.5 Interface Design

Internal Interfaces

Module design consists of developing functionality for communicating with in-field ramp meters using the Caltrans SDRMS protocol (Requirement 69). These developed classes are extended from interfaces and classes in the `us.mn.state.dot.tms.comm` namespace as shown in Figure 1.2 on page 5 and Figure 1.3 on page 6. Developed classes and interfaces will be within the namespace `us.ca.state.dot.tms.comm.sdrms`. Multiple ramp metering protocols are supported by this namespace convention (e.g. `us.ca.state.dot.tms.comm.urms`) (Requirement 64). Developed classes will include:

- Extended MessagePoller: extends the existing MessagePoller class and may implement existing interfaces such as MeterPoller and SignPoller.
- Extended ControllerOperation: extends the existing ControllerOperation class. This may include operations such as: QueryAlarms, Download, and QueryMeterStatus. Each Operation will include internally defined subclasses of Phase [4].
- ControllerException: an extended IOException, specific to SDRMS.

External Interfaces

The primary external interface is the SDRMS protocol.

6.6 Module Detailed Design

For detailed design information see the JavaDoc[1].

6.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security features and policies.
- Existing Caltrans communication between the field devices and the TMC will be used.
- If secure communication with field devices over land lines is desirable, a field controller should be developed that will provide:
  - Support for the SDRMS protocol (Requirement 68),
  - Interface with ramp meter hardware,
  - Support encrypted TCP/IP communication with the TMC,
  - Interface with IRIS.

6.8 Error and Exception Handling

Standard Java exception handling will be used in this module and is already used within the IRIS client and server (Requirement 71). The ControllerException class extends java.lang.Exception and is used to specify SDRMS-specific exceptions. Existing IRIS event logging functionality will be used. SDRMS specific events may be developed.

6.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

[1]For JavaDoc see `http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/`

Chapter 7

Weather Stations and Sensors

7.1 Purpose and General Description

Weather station functionality has been identified as priority 5 for D10. IRIS does not presently use weather station data. The primary motivation for adding weather station functionality is to provide TMC operators with information about the origin of CMS messages and the ability to verify weather station sensor inputs for CAWS-generated messages. This will be achieved with a dedicated map layer in the IRIS client that depicts the location of weather stations. This new layer will be consistent with existing IRIS mapping capabilities and conventions (Requirement 75). Colored placemarks may be used to indicate important conditions such as wind speed. Clickable placemarks may also be used to provide weather station drill-down capability for access to specific detailed information such as wind direction. The TAG has identified weather station functionality as a valuable generalized enhancement to IRIS. In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

7.2 Architecture

Relationships among primary structural elements are listed below. The primary elements to be developed are subclasses of existing IRIS classes specialized for D10 weather stations and sensors.

- Weather Stations: this includes both SSI and Qualimetrics stations presently in use [2].
- Communications Infrastructure: weather stations are connected to the TMC via leased digital lines via a Channel Service Unit/Data Service Unit (CSU/DSU) and the Cingular wireless VPN via InfoTek Wizards (see Appendix A on page 45). For additional communication details see the Task 3 and Task 4 documents[2, 3].
- D10 Middleware: performs weather station data format normalization. Provides weather station data in the form of an XML file over HTTP or streaming XML data over a TCP/IP network connection to IRIS (Requirement 76).

- IRIS Client: provides maps and other information integrated into a single client for TMC operators.
- IRIS Client Weather Station Map: station position and symbolic placemarks will be displayed on this map (Requirement 77,78). Programmatically, this is a subclassed instance of the Layer class and will be named WSLayer. It is contained by a MapPane, which is contained by a MapBean, which is contained within a MapTab in the IRIS client [4]. WSLayer is a container for an XmlClient object.
- WSListModel: extends AbstractListModel, implements WSListener, in the `us.mn.state.dot.tms.client.incidents` namespace. Contained by a subclassed IrisTab object in the client.
- XmlClient: implements Runnable and is contained by WSLayer. Subclasses periodically read XML files with specific formats. Contains a list of DdsListener objects which are notified when new data arrives. A subclass will be implemented named WSXmlClient.
- WSObservation: an interface to be developed, extends Comparable. It will be implemented to represent a weather station observation.
- WSListener: an interface that extends DdsListener. Classes that implement it receive notifications from XmlClient instances when new station observations arrive.
- WSObservationException: extends Exception. May be used to represent observation related exceptions.

7.3 Hardware Utilization

Existing hardware will be used. IRIS client and server memory, processing and storage use are expected to be within the bounds of that provided by the existing D10 IRIS system.

7.4 Concept of Execution

Structural elements are shown in the Architecture Section. The order listed below indicates a time sequence of events:

1. Weather Stations: collect sensor data.
2. Communications Infrastructure: transmits weather station data to D10 middleware.
3. D10 Middleware: normalizes weather station data into a single format and provides HTTP or streaming TCP/IP data.
4. IRIS Client: contains a MapTab, which contains a WSListModel which implements the WSListener interface. The MapTab also contains a MapBean, which contains a MapPane, which contains a weather station layer, WSLayer.

5. WSLayer: contains a subclassed XmlCLient named WSXmlClient.

6. WSXmlClient: a thread which is periodically reading station observations from D10 middleware. The XML data is parsed, new Observation instances are created, which are passed to WSListModel which implements WSListener.

7. WSListener Notification: an interface that extends DdsListener. Classes that implement it receive notifications from XmlClient instances when new station observations arrive.

7.5 Interface Design

For existing internal interfaces see the JavaDoc for the classes and interfaces listed above[1].

External Interfaces

The interface between D10 IRIS and the weather station data is the XML middleware data format.

7.6 Module Detailed Design

For detailed design information see the JavaDoc[2].

7.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security and reliability features and policies (Requirement 79).

7.8 Error and Exception Handling

Standard Java exception handling will be used in this module and is already used within the IRIS client and server. A new WSObservationException class may be developed.

[1]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/
[2]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

7.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

Chapter 8

Event Logging

8.1 Purpose and General Description

Event logging functionality has been identified as priority 7 by the TAG. D10 IRIS will use existing IRIS logging functionality. In addition, D10-specific events will be logged. This includes CAWS events such as changes in visibility, wind speed, etc. In discussions related to the IRIS class structure, refer to Figure 1.1, Figure 1.2, and Figure 1.3 on pages 4–6.

8.2 Architecture

Relationships among primary structural elements within the module are listed below.

- Hardware devices: typically the point of origin for events to be logged. Examples include weather station events (e.g. change in visibility) and device communication events (e.g. communication time-out).

- Application event logging: specific events within the D10 IRIS code base will be logged as they occur. Subclassed instances of the TMSEvent class are created and logged by an instance of the LogImpl class. New subclasses of TMSEvent will be created for D10 IRIS specific events (Requirement 84,85).

- IRIS server Log interface: Java classes within the namespace `us.mn.state.dot.tms.log`. These classes and interfaces provide logging functionality to the IRIS server and interface with the Log database.

- Log database: events are logged into this PostgreSQL database. The existing structure will be used (Requirement 86). The log database consists of tables, procedures, views, and other definitions.

8.3 Hardware Utilization

Existing hardware will be used and no specialized hardware is required. Memory, processing and storage use are expected to be within the bounds of that provided by the existing IRIS system.

8.4 Concept of Execution

Structural elements are shown in the Architecture section. The order listed indicates the time sequence of events during logging:

1. Hardware device: triggers log events (typically),
2. Application code: calls a LogImpl method to log the event, passing a subclassed TMSEvent object specific to the type of event being logged,
3. Logging Classes: receives a log event request and communicates with the database,
4. Log Database: write the event to a file stored on disk.

8.5 Interface Design

Internal Interfaces

Internal communication is through the following classes and interfaces:

- TMSEvent: an event created by code that detects a condition to be logged. Subclasses are instantiated and passed as an argument to the add(event) method.
- LogImpl: used by code to access the log. An instance is contained by TMSObjectImpl and created when the IRIS server starts. Callers use the method add(event) to log new events. An event must be created and passed to this method.
- Log: provides the interface to the database for LogImpl.
- TMSEvent: base class for all event objects used for logging. If new event types are necessary (Requirement 87), they will be subclasses of TMSEvent and associated with new event tables within the log database.
- EventVault: provides an interface to the log database.

External Interfaces

External access to logging information can be provided through direct access to PostgreSQL log database via Structured Query Language (SQL).

8.6 Module Detailed Design

For detailed design information see the JavaDoc[1].

8.7 Security

The following security assumptions are made:

- This module will use existing IRIS and D10 security features.
- The PostgreSQL database will be configured to protect access to the contents of the log database.

8.8 Error and Exception Handling

Standard Java exception handling will be used in this module and is already used within the IRIS client and server (Requirement 88,89). Error handling within the traffic server will be through log files. The EventVaultException class extends java.lang.Exception and is used to specify logging-related exceptions.

8.9 Requirements Traceability

For requirements cited by number, see the Task 4 report [3].

[1]For JavaDoc see http://iris.ahmct.ucdavis.edu/iris/javadoc/iris/

Chapter 9

Highway Advisory Radio and Extinguishable Message Signs

9.1 Purpose and General Description

Highway Advisory Radio (HAR) functionality has been identified as priority 8 by the TAG. *The addition of HAR functionality to D10 IRIShas been deferred to a future project and will not be performed in the existing project.* It is included here for future reference.

For additional technical details beyond that provided here, see the Task 3 and Task 4 documents[2, 3]. Quixote's DR2000 software is used within the TMC to control HAR and the associated proprietary hardware. A dedicated TMC workstation is presently used by TMC operators to perform this task. The workstation also contains proprietary Quixote hardware that interfaces with the HAR system.

The anticipated approach to integrate D10 IRISwith existing HAR functionality will use an additional Quixote software module that specifically provides 3rd party application access to proprietary Quixote hardware and software. If this technical approach is used, the purchase of this Quixote software module would be required for HAR development and integration with D10 IRIS. Other technical approaches may be considered. For further information see [2, 3].

Chapter 10

Intersection Traffic Signals

10.1 Purpose and General Description

Intersection traffic signal monitoring has been identified as priority 9 for D10. *The addition of intersection traffic signal functionality to D10 IRIShas been deferred to a future project and will not be performed in the existing project.* It is included here for future reference.

For additional technical details beyond that provided here, see the Task 3 and Task 4 documents[2, 3]. Approximately six traffic signals are presently connected to the TMC with dial-up lines, but are not used. D10 has long term plans to connect traffic signals to the TMC. IRIS presently has the ability to read signal-generated data files on a daily basis, but otherwise does not control signals. D10 is presently using Model 2070 controllers.

When traffic signal support is added to D10 IRIS, it is anticipated that the Caltrans Central Signal Control System (CTNet) protocol will be used. For further information see [2, 3].

References

[1] M.T. Darter, K.S. Yen, B. Ravani, and T.A. Lasky. Literature Review of National Developments in ATMS and Open Source Software. Technical Report UCD-ARR-06-12-08-01, AHMCT, December 2006.

[2] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. Review of Caltrans District 10 Transportation Management Center Operations and Equipment. Technical Report UCD-ARR-07-09-30-01, AHMCT, September 2007.

[3] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. Review of Mn/IRIS and Caltrans District 10 TMC Compatibility and Functional Requirements for D10 IRIS Demonstration Study. Technical Report UCD-ARR-07-09-30-02, AHMCT, September 2007.

[4] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. Review of Mn/IRIS Software and Test Cases for Caltrans District 10 IRIS Demonstration Study. Technical Report UCD-ARR-07-12-31-01, AHMCT, December 2007.

[5] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. Research and Development of Open-Source Advanced Traffic Management System Hardware and Software Components. Technical Report UCD-ARR-08-06-30-01, AHMCT, July 2008.

[6] Minnesota Department of Transportation. Intelligent Roadway Information System (IRIS) As Built System Design Document. Technical report, Minnesota Department of Transportation, June 2007.

[7] Minnesota Department of Transportation IRIS Project. Mn/DOT IRIS JavaDoc, December 2007.

[8] Minnesota Department of Transportation IRIS Project. Mn/DOT IRIS source code, December 2007.

[9] Traffic Operations. Systems Engineering Document Template Tree. Technical Report EP-00, California Department of Transportation, October 2007.

[10] Traffic Operations. Systems Engineering Methodology, Interface Detailed Design. Technical Report SP-063, California Department of Transportation, November 2005.

[11] Traffic Operations. Systems Engineering Methodology, System Architecture. Technical Report SP-051, California Department of Transportation, November 2005.

[12] Traffic Operations. Systems Engineering Methodology, Software Detailed Design. Technical Report SP-061, California Department of Transportation, November 2005.

[13] Wavetronix LLC. SS105 SmartSensor Data Protocol V2.02, February 2004. URL http://www.wavetronix.com/.

Appendix A

District 10 InfoTek Wizard GSM Modem

The Wizard is a wireless modem jointly developed by D10 and InfoTek Inc. (CommTech Systems)[1]. The Wizard is a Global System for Mobile Communications (GSM) based General Packet Radio Service (GPRS)/Enhanced Data Rates for GSM Evolution (EDGE) modem. It has 32 digital inputs, 8 digital outputs, and an EIA serial communications standard (RS-232) port. It has an embedded Java runtime that supports the TCP/IP stack. The Java development environment also supports Short Message Service (SMS) for remote command and alert and HTTP. Each modem has an associated phone number and one or two IP addresses.

[1]For more detailed Wizard information, see `http://commtechsystems.com/wizard.htm`.