



# **Advanced Highway Maintenance and Construction Technology Research Center**

Department of Mechanical and Aerospace Engineering  
University of California at Davis

## **Research & Development of Open-Source Advanced Transportation Management System Hardware and Software Components**

Michael T. Darter, Kin S. Yen,  
Travis Swanston, Bahram Ravani &  
Ty A. Lasky: Principal Investigator

Federal Report Number: CA09-0981  
AHMCT Research Report: UCD-ARR-09-08-31-01  
Final Report of Contract: 65A0210, T.O. 06-22

August 31<sup>st</sup>, 2009

## **California Department of Transportation**

Division of Research and Innovation



**Research & Development of Open-Source  
Advanced Transportation Management System  
Hardware and Software Components\***

Michael T. Darter, Kin S. Yen,  
Travis Swanston, Bahram Ravani &

Ty A. Lasky: Principal Investigator

AHMCT Research Report  
UCD-ARR-09-08-31-01

Final Report of Contract 65A0210, T.O. 06-22

Prepared for the California Department of Transportation  
Division of Research and Innovation

August 31<sup>st</sup>, 2009

**Affiliations:**

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

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



**TECHNICAL REPORT DOCUMENTATION PAGE**

TR0003 (REV. 10/98)

1. REPORT NUMBER <b>CA09-0981</b>	2. GOVERNMENT ASSOCIATION NUMBER	3. RECIPIENT'S CATALOG NUMBER
4. TITLE AND SUBTITLE <b>Research &amp; Development of Open-Source Advanced Transportation Management System Hardware and Software Components</b>		5. REPORT DATE <b>August 31<sup>st</sup>, 2009</b>
		6. PERFORMING ORGANIZATION CODE <b>AHMCT</b>
7. AUTHOR(S) <b>Michael T. Darter, Kin S. Yen, Travis B. Swanston, Bahram Ravani &amp; Ty A. Lasky</b>		8. PERFORMING ORGANIZATION REPORT NO. <b>UCD-ARR-09-08-31-01</b>
9. PERFORMING ORGANIZATION NAME AND ADDRESS <b>AHMCT Research Center UCD Dept of Mechanical &amp; Aerospace Engineering Davis, California 95616-5294</b>		10. WORK UNIT No. (TRAIS)
		11. CONTRACT OR GRANT NUMBER <b>65A0210, T.O. 06-22</b>
12. SPONSORING AGENCY NAME AND ADDRESS <b>California Department of Transportation Division of Research &amp; Innovation 1227 O Street Sacramento, CA 94273-0001</b>		13. TYPE OF REPORT AND PERIOD COVERED <b>Final Report October 2005 - August 2009</b>
		14. SPONSORING AGENCY CODE
15. SUPPLEMENTAL NOTES		
16. ABSTRACT <p>This is the final report for the study "Research &amp; Development of Open-Source Advanced Transportation Management System Hardware and Software Components." The study researched, implemented, and extended the Intelligent Roadway Information System (IRIS) open-source Advanced Transportation Management System (ATMS) within the <a href="#">California Department of Transportation</a> (Caltrans) District 10, as a demonstration of the potential effectiveness of the collaborative open-source development approach between multiple transportation agencies. The IRIS ATMS was originally developed and open-sourced by the Minnesota Department of Transportation (Mn/DOT). This study provided the first implementation of an open-sourced ATMS outside of its agency of origin. The report provides background material and discusses the study method, testing and validation, IRIS portability, study results, and conclusions and recommendations.</p>		
17. KEY WORDS <b>ATMS, Open-Source Software (OSS), Transportation Management Center (TMC), Remote Weather Information System / Roadway Weather Information System (RWIS), Highway operations, IRIS, Traffic Management, Transportation Management</b>	18. DISTRIBUTION STATEMENT <b>No restrictions. This document is available to the public through the National Technical Information Service, Springfield, Virginia 22161.</b>	
19. SECURITY CLASSIFICATION (OF THIS REPORT) <b>Unclassified</b>	20. NUMBER OF PAGES <b>112</b>	21. PRICE

Reproduction of completed page authorized



# Abstract

This is the final report for the study “Research & Development of Open-Source Advanced Transportation Management System Hardware and Software Components.” The study researched, implemented, and extended the Intelligent Roadway Information System (IRIS) open-source Advanced Transportation Management System (ATMS) within the [California Department of Transportation](#) (Caltrans) District 10, as a demonstration of the potential effectiveness of the collaborative open-source development approach between multiple transportation agencies. The IRIS ATMS was originally developed and open-sourced by the Minnesota Department of Transportation (Mn/DOT). This study provided the first implementation of an open-sourced ATMS outside of its agency of origin. The report provides background material and discusses the study method, testing and validation, IRIS portability, study results, and conclusions and recommendations.





# Table of Contents

<b>Abstract</b>	<b>iii</b>
<b>Table of Contents</b>	<b>viii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xi</b>
<b>Disclaimer/Disclosure</b>	<b>xiii</b>
<b>Acronyms and Abbreviations</b>	<b>xv</b>
<b>Acknowledgments</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Document Purpose . . . . .	1
1.2 Study Objectives and Motivation . . . . .	1
1.3 The Collaborative Approach . . . . .	2
1.4 Project History . . . . .	2
1.4.1 Project Scope . . . . .	3
1.4.2 The History of Caltrans' Attraction to Mn/DOT's IRIS . . . . .	4
1.4.3 Internal Caltrans Project Demonstrations . . . . .	5
1.4.4 Additional Caltrans IRIS Deployments . . . . .	5
1.4.5 Study Time-line . . . . .	5
1.5 Project Documentation . . . . .	8

<b>2</b>	<b>Method</b>	<b>9</b>
2.1	Study Management . . . . .	9
2.1.1	Software Engineering . . . . .	9
2.1.2	Project Coordination . . . . .	9
2.1.3	Project Management and Communication Tools . . . . .	10
2.1.4	Release Scheduling . . . . .	11
2.1.5	Focus on Reliability . . . . .	11
2.2	Requirements Management . . . . .	12
2.2.1	Stage 1: Deployment of Primary Feature Set . . . . .	12
2.2.2	Stage 2: Iterative Refinement and Extension . . . . .	12
2.3	Cooperative Development Process . . . . .	12
2.3.1	Single IRIS Code-base . . . . .	13
2.3.2	Distributed Source Repository Management System . . . . .	14
2.3.3	Typical Development Sequence . . . . .	14
<b>3</b>	<b>Testing, Validation, and Verification</b>	<b>17</b>
3.1	Test Plan . . . . .	17
3.2	Automated Unit Test Cases . . . . .	18
3.3	User Acceptance Tests . . . . .	19
3.4	Defect Tracking . . . . .	20
3.5	Testing and Verification Results . . . . .	20
3.6	Automated Warning System Verification . . . . .	20
3.7	IRIS Scalability Testing . . . . .	21
3.8	Future Testing and Verification . . . . .	22
<b>4</b>	<b>IRIS Portability</b>	<b>25</b>
4.1	Relevance of Portability . . . . .	25
4.2	IRIS Modularity . . . . .	25
4.3	IRIS Clients Written In Other Software Languages . . . . .	25
4.4	Configuring IRIS for an Agency . . . . .	26

4.4.1	System Attributes . . . . .	26
4.4.2	Internationalization . . . . .	27
4.4.3	Properties Files . . . . .	27
4.4.4	User Permissions . . . . .	28
4.4.5	Interfacing IRIS with Devices and Software Systems . . . . .	29
4.4.6	Help System . . . . .	29
4.5	IRIS Operating System Portability . . . . .	30
<b>5</b>	<b>Results</b>	<b>31</b>
5.1	TMC Operational Safety Enhancements . . . . .	31
5.1.1	CMS Safety Enhancements . . . . .	31
5.1.2	Automated Warning System Safety Enhancements . . . . .	33
5.1.3	Functional Integration Safety Enhancements . . . . .	36
5.2	Functional Enhancements to TMC Operations . . . . .	36
5.2.1	General Enhancements to TMC Operations . . . . .	36
5.2.2	Enhancements to TMC Traffic Monitoring . . . . .	37
5.2.3	Enhancements to TMC CMS Monitoring and Control . . . . .	37
5.2.4	Enhancements to TMC Video Monitoring and Control . . . . .	38
5.2.5	Enhancements to TMC Reporting . . . . .	38
5.3	IRIS Integration with Proprietary Protocols . . . . .	39
5.4	Project Contributions to IRIS . . . . .	40
5.4.1	Quantitative Contributions to IRIS . . . . .	40
5.4.2	Qualitative Contributions to IRIS . . . . .	41
5.4.3	Mn/DOT's Perspective on Study Contributions . . . . .	44
5.5	Other Study Products . . . . .	45
5.5.1	CASPER Field Controller Simulator . . . . .	45
5.5.2	IRIS Developer Ticket System . . . . .	45
5.5.3	Sensor Server . . . . .	46
5.5.4	Defect Discovery and Repair . . . . .	46
5.5.5	Traffic Server . . . . .	47

5.5.6	Watchdog . . . . .	48
5.6	Costs . . . . .	48
5.6.1	Estimating the Free Economic Value of IRIS . . . . .	49
5.6.2	Component and Functional Area Effort Breakdown . . . . .	49
5.6.3	Estimated IRIS Maintenance Costs . . . . .	50
5.6.4	High-level Cost Comparison of Open and Proprietary ATMS . . . . .	51
5.7	Effort Estimates for New IRIS Implementations . . . . .	56
5.8	Future Enhancements . . . . .	56
<b>6</b>	<b>Conclusions and Recommendations</b>	<b>59</b>
6.1	Strengths of the Collaborative Approach . . . . .	59
6.2	Obstacles and Concerns with the Collaborative Approach . . . . .	60
6.3	Lessons Learned . . . . .	62
6.4	IRIS Strengths and Areas for Improvement . . . . .	62
6.5	Conclusions . . . . .	64
6.6	Recommendations . . . . .	65
	<b>References</b>	<b>69</b>
	<b>Appendices</b>	<b>71</b>
<b>A</b>	<b>Functional and Non-functional Requirements</b>	<b>71</b>
<b>B</b>	<b>IRIS Installation Verification Procedure</b>	<b>87</b>

# List of Figures

1.1	Caltrans District Map . . . . .	3
2.1	ITS Project Life-cycle Phases and Life-cycle Tasks [20] . . . . .	10
2.2	Iterative and Incremental Development Model (see Section 2.2.2) . . . . .	13
2.3	IRIS Source Code Management (see pg. 14) . . . . .	15
3.1	IRIS Scalability Test Results (see Section 3.7) . . . . .	23
4.1	IRIS Binary Modules . . . . .	26
4.2	IRIS Attribute Editor Form (see Section 4.4.1) . . . . .	27
4.3	Internationalized IRIS Menu (see pg. 27) . . . . .	28
4.4	IRIS User Permissions Configuration Form (see pg. 28) . . . . .	28
4.5	IRIS Device Driver Configuration Form (see pg. 29) . . . . .	29
5.1	Caltrans District 10 Simplified Data Flow Diagram . . . . .	32
5.2	A Portion of the CMS Communication Network Health Report (SignScope) . . . . .	33
5.3	A Portion of the CMS Communication Network Health Report (SignScope) . . . . .	33
5.4	List of Real-time Automated Warning System Generated Messages . . . . .	34
5.5	Verification of CAWS CMS Activated by Mapped Traffic Congestion . . . . .	35
5.6	Verification of CAWS CMS Activated by Mapped Incident . . . . .	36
5.7	IRIS integration of incidents, CMS, camera positions, and traffic (see pg. 36) . . . . .	37
5.8	Screenshot of the Chainsaw Application Log Viewing Tool (see pg. 38) . . . . .	39
5.9	Sensor Server Architecture and Relationship with IRIS (see pg. 39) . . . . .	40
5.10	Cumulative SLOC for all IRIS Modules . . . . .	42

5.11 Collaborative Development: Cumulative IRIS Contributions by Agency . .	43
5.12 Cumulative SLOC for AHMCT-Developed Open ATMS Applications . . . .	44
5.13 Traffic Server Relationship with IRIS (see pg. 47) . . . . .	48

# List of Tables

1.1	Study Timeline . . . . .	6
5.1	Quantified Project Results by Component (see Section 5.4.1) . . . . .	41
5.2	Estimating the Free Economic Value of IRIS . . . . .	49
5.3	Component and Functional Area Effort Breakdown (see pg. 49) . . . . .	50
5.4	Open and Proprietary ATMS Acquisition Costs For One Agency . . . . .	52
5.5	Installation, Configuration, and Customization Costs for Open and Proprietary ATMS (see pg. 53) . . . . .	53
5.6	Costs: Installation and Configuration Only, i.e. When No Customization Needed for Open ATMS (see pg. 53) . . . . .	53
5.7	Annual Maintenance Costs For One Agency for Open and Proprietary ATMS (see pg. 54) . . . . .	54
5.8	Five-Year Acquisition, Configuration, & Maintenance Costs of Open and Proprietary ATMS For One Agency (see pg. 55) . . . . .	55
A.1	Tasks, Functional, and Non-functional Requirements . . . . .	71





# 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 Aerospace 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.

<b>AHMCT</b>	<a href="#">Advanced Highway Maintenance &amp; Construction Technology</a>
<b>AMBER</b>	America's Missing: Broadcast Emergency Response Plan
<b>ATMS</b>	Advanced Transportation Management System
<b>AWS</b>	Automated Warning System
<b>BSD</b>	Berkeley Software Distribution
<b>Caltrans</b>	<a href="#">California Department of Transportation</a>
<b>CASPER</b>	Controller Array Simulator for Performance and Enhanced Reliability
<b>CAWS</b>	Caltrans Automated Warning System
<b>CCB</b>	Change Control Board
<b>CMS</b>	Changeable/Dynamic Message Sign
<b>COCOMO</b>	COConstructive COst MOdel
<b>D1</b>	District 1
<b>D3</b>	District 3
<b>D4</b>	District 4
<b>D5</b>	District 5
<b>D10</b>	District 10
<b>DMS</b>	Dynamic Message Sign
<b>DOT</b>	Department of Transportation
<b>DRI</b>	Division of Research and Innovation

<b>FSR</b>	Feasibility Study Report
<b>FTE</b>	Full Time Equivalent
<b>GNU</b>	<a href="#">GNU's Not Unix</a>
<b>GPL</b>	<a href="#">General Public License</a>
<b>GUI</b>	Graphical User Interface
<b>HAR</b>	Highway Advisory Radio
<b>HP</b>	Hewlett-Packard
<b>HTML</b>	HyperText Markup Language
<b>HTTP</b>	HyperText Transfer Protocol
<b>I18N</b>	Internationalization
<b>IID</b>	Iterative and Incremental Development
<b>IP</b>	Internet Protocol
<b>IRIS</b>	Intelligent Roadway Information System
<b>IT</b>	Information Technology
<b>ITS</b>	Intelligent Transportation System
<b>JVM</b>	Java Virtual Machine
<b>KML</b>	Keyhole Markup Language
<b>LDAP</b>	Lightweight Directory Access Protocol
<b>M170</b>	Model 170
<b>MDT</b>	Montana Department of Transportation
<b>Mn/DOT</b>	Minnesota Department of Transportation
<b>MPEG-4</b>	Moving Picture Experts Group 4
<b>NDA</b>	Non-Disclosure Agreement
<b>NPE</b>	Null Pointer Exception
<b>NTCIP</b>	National Transportation Communications for ITS Protocol
<b>OSI</b>	Open Source Initiative
<b>OSS</b>	Open-Source Software

<b>PeMS</b>	Freeway Performance Measurement System
<b>PROM</b>	Programmable Read-Only Memory
<b>PTZ</b>	Pan Tilt Zoom
<b>RFP</b>	Request for Proposal
<b>RHEL</b>	Red Hat Enterprise Linux
<b>RMI</b>	Remote Method Invocation
<b>RPM</b>	RPM Package Manager
<b>RWIS</b>	Remote Weather Information System / Roadway Weather Information System
<b>SDRMS</b>	San Diego Ramp Metering System
<b>SLOC</b>	Source Lines of Code
<b>SOCCS</b>	Satellite Operations Center Command System
<b>SONAR</b>	Simple Object Notification And Replication
<b>SQL</b>	Structured Query Language
<b>SV170</b>	M170 Programmable Read-Only Memory (PROM) chip
<b>TAG</b>	Technical Advisory Group
<b>TCP/IP</b>	Transmission Control Protocol / Internet Protocol
<b>TDD</b>	Test Driven Development
<b>TMC</b>	Transportation Management Center
<b>TRB</b>	Transportation Research Board
<b>UCD</b>	University of California Davis
<b>UI</b>	User Interface
<b>URL</b>	Uniform Resource Locator
<b>VDS</b>	Vehicle Detector Station
<b>VSL</b>	Variable Speed Limit
<b>WisDOT</b>	Wisconsin Department of Transportation
<b>x86</b>	commodity Intel, AMD, and compatible microprocessors



# 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 (TAG). The authors also acknowledge the dedicated efforts of the AHMCT development team members. Special thanks to Caltrans Headquarters Traffic Operations (Stan Slavin, Alan Benson, Maria Hionides, Jeff McRae, and Robert Copp), Caltrans Division of Research and Innovation (DRI) (Fred Yazdan, Roya Hassas, Sean Campbell, and Larry Orcutt), Caltrans District 10 (Dinah Bortner, Laurie Jurgens, John Castro, Mohammad Battah, Veronica Cipponeri, Arlene Cordero, Toni Moon, Joe Silvey, and James Collins), Caltrans District 1 (John Carson, Joe Dower, and Jim Sandford), Caltrans District 5 (Sherwyn Gilliland, Jacques Van Zeventer, Steven Gee, Julie Gonzalez, David Ybarra, Mike Keller), Caltrans District 12 Traffic Operations (Omid Segal and Morteza Fahrtash), Caltrans Information Technology (IT) (Larry Tjoelker, Margaret Frontella, Todd Larson, and Danial Peck), and finally, the outstanding Mn/DOT team (James Kranig, Doug Lau, Timothy Johnson, and Ralph Adair).





# Chapter 1

## Introduction

### 1.1 Document Purpose

This document is the final report of the Caltrans IRIS Demonstration study. It summarizes study results, lessons learned, and conclusions, and provides a direction for future research. The Open ATMS research study is a multi-year research project undertaken by the [Advanced Highway Maintenance & Construction Technology](#) (AHMCT) Research Center at the University of California, Davis.<sup>1</sup>

### 1.2 Study Objectives and Motivation

The objective of this research study was to dramatically reduce present and future ATMS life-cycle costs. From the original research proposal:

*The current Caltrans ATMS is composed of several proprietary single-source components. The lack of market competition for these single-source components leads to ever increasing one-time acquisition costs and yearly maintenance costs. Consequently, additional ATMS deployment is severely impacted.*

*Therefore, a robust and reliable ATMS can be designed with an open and modular architecture using open-source components and commodity hardware with standard interfaces. As a result, the one-time build-up cost and ongoing maintenance costs would be dramatically reduced. Now is the right time to investigate the use and deployment of an open-source ATMS throughout Caltrans.*

---

<sup>1</sup>For AHMCT see <http://ahmct.ucdavis.edu>

## 1.3 The Collaborative Approach

The terms *collaborative approach* and *open-source* are used interchangeably throughout this document. In general, the term open-source is used in many contexts and has multiple meanings. For our purposes it has three meanings:

1. Legal definition: the Open Source Initiative (OSI) is a non-profit organization that has legally defined the term *open source*.<sup>2</sup> For a software license to be considered open-source, it must meet certain legal requirements. Examples of popular open-source licenses include the [General Public License](#) (GPL), Berkeley Software Distribution (BSD), and Apache license. The legal terms of these licenses differ and are not necessarily interchangeable.
2. Software development methodology: seeks to leverage distributed development across multiple developers that freely share knowledge about a software project and seek to improve and refine it. This development methodology strongly encourages peer review, which is widely used in other engineering practices. The distributed nature of development also keeps costs low.
3. Approach for building knowledge communities: the legal requirements of open-source licenses (e.g. source code is freely available) results in the development of distributed knowledge communities. These communities cooperate and collaborate to improve the software. The more people that are involved with an open-source project, the more likely they will contribute, and the stronger the project grows. This positive feedback cycle benefits the entire knowledge community.

The open-source IRIS project uses the GNU GPL license. For a full discussion of open-source licenses such as the GPL and BSD licenses, see online sources.<sup>3</sup> In summary, the GPL license stipulates:

- Source code must be publicly available,
- The source code for derivative works (modified GPL applications) must be publicly available, and
- Derivative works are covered by the GPL.

## 1.4 Project History

In 2004, the Caltrans Division of Research and Innovation (DRI) initiated a search for innovative methods of reducing ATMS costs. Caltrans' existing ATMS system was effective when used within 7 of 12 districts:<sup>4</sup> 3, 4, 6, 7, 8, 11, and 12 (see Figure 1.1). However,

---

<sup>2</sup>For the Open Source Initiative (OSI) see <http://opensource.org>

<sup>3</sup>See <http://en.wikipedia.org/wiki/GPL> and [http://en.wikipedia.org/wiki/BSD\\_licenses](http://en.wikipedia.org/wiki/BSD_licenses)

<sup>4</sup>As of September 2009, the proprietary ATMS system is not yet installed in District 4.

Caltrans was seeking methods of reducing initial and ongoing ATMS costs of this system, which was built entirely with proprietary hardware and software components. The research proposal was granted in February of 2005 (scope 1).



Figure 1.1: Caltrans District Map

### 1.4.1 Project Scope

Throughout, the focus of the research study has been cost reduction. The study was re-scoped twice:

1. Original scope: development of an open-source ATMS, 02/2005: AHMCT proposed to develop a full open-source ATMS with a focus on cost reduction.
2. First re-scope: development of an open database, 02/2006: the goal was to create an open-source version of the existing proprietary ATMS system database. The research project was essentially on hold, except for some background literature search and reporting, between December 1st, 2005 and January 3rd, 2007 (13 months), while lawyers for the University of California and Caltrans attempted to reach agreement on the terms of the Non-Disclosure Agreement (NDA) that the researchers were required to sign to gain access to the proprietary ATMS source code. Ultimately, after 13 months of discussion, the attorneys could not reach an agreement.
3. Second re-scope: implementation and extension of an existing open-source ATMS, 07/2007: in May of 2007, Mn/DOT released the source code of their existing IRIS ATMS as open-source. This re-scope focused on determining the costs and benefits of implementing and extending<sup>5</sup> an existing open-source ATMS within Caltrans.

<sup>5</sup>Extend is used throughout the document and signifies the addition of new capabilities, improved performance, reliability, portability, etc.

Caltrans District 10 (D10) was selected for the demonstration because it did not have an existing ATMS system. Therefore, potential benefits provided by the IRIS ATMS would minimize overlapping functionality in existing systems, providing a net benefit to D10. Extension of IRIS included: 1) interfacing with existing hardware and software systems, 2) adding features the TMC deemed important for operations, and 3) fixing defects.

### 1.4.2 The History of Caltrans' Attraction to Mn/DOT's IRIS

*The information in this section was provided by Caltrans and Mn/DOT and edited by AHMCT.*

In March of 2006, Caltrans and Mn/DOT had discussions about the Operations Software used within the Mn/DOT TMC. This software (IRIS) had been developed internally by Mn/DOT. Over the years, Mn/DOT had received multiple requests from outside agencies about the possibility of making their IRIS ATMS available to other agencies. Approximately six months prior to the conversation with Caltrans, Mn/DOT had initiated an approval request for providing Intelligent Roadway Information System (IRIS) to other agencies.

Mn/DOT had an initial internal meeting to discuss how to proceed and all agreed that external distribution would be beneficial for Mn/DOT and provide a cost-effective way for other agencies to obtain a traffic management system. Mn/DOT developed several distribution and support scenarios and presented the plan to top management. Mn/DOT decided to license IRIS using the GNU [General Public License](#) (GPL).

Between January and June of 2007, several telephone conversations took place between Mn/DOT, Washington State Department of Transportation (DOT), Wisconsin DOT, and Caltrans. In addition, Mn/DOT was gathering information about the system capabilities of the existing Caltrans ATMS system for comparison with IRIS capabilities and requirements. Feature comparison was important because Mn/DOT was particularly interested in partnering with other DOTs for collaborative development of enhancements. For example, Caltrans could sponsor the development of a new functional module that would be used by both agencies.

The Caltrans ATMS system and IRIS had a number of complimentary features. For example, IRIS did not perform Incident Detection or generate and manage Incident Response Plans. However, IRIS supports the National Transportation Communications for ITS Protocol (NTCIP) standard (class A, B, and C) to communicate with Changeable/Dynamic Message Sign (CMS). Both the Caltrans ATMS and IRIS were installed in multiple geographic locations. In addition, IRIS was being used to remotely run a satellite operations center (St. Cloud)—a capability that was attractive to Caltrans for server consolidation.

In July of 2007, the University of California Davis (UCD) Open-ATMS study was re-scoped to implement and extend IRIS within Caltrans D10 to demonstrate the feasibility, benefits, and costs of this collaborative approach.

### **1.4.3 Internal Caltrans Project Demonstrations**

The Open ATMS project demonstration to upper management was in October of 2008, at the D10 headquarters in Stockton, California. The low bandwidth network capabilities of the IRIS client enabled a second remote demonstration of the D10 system in December of 2008 in Sacramento, California. This included the Chief of the Division of Traffic Operations and the Chief of the Division of Research & Innovation. The outcome was positive, resulting in subsequent maintenance funding and the initiation of the Caltrans IT Feasibility Study Report (FSR) process for subsequent deployment.

### **1.4.4 Additional Caltrans IRIS Deployments**

Based on the positive results from the D10 IRIS deployment, Caltrans decided to add IRIS test deployments in District 1 (D1) and District 5 (D5) to the demonstration study. IRIS was activated in D5 on August 12<sup>th</sup> 2009 and in D1 on August 26<sup>th</sup> 2009. See Section 5.6.4 for more information.

With the additional IRIS deployments, Caltrans sought to determine the usefulness of IRIS in additional districts, which potentially had different needs. For example, unlike D10, Districts 1 and 5 are largely rural. A key concern for Caltrans is reusability—avoiding highly customized single district solutions that are difficult or impossible for other districts to use. High reusability decreases costs and improves reliability. Deployment to these additional districts provided an opportunity to measure IRIS reusability. See Section 2.3.1.

### **1.4.5 Study Time-line**

Significant events during the course of research are noted in Table 1.1 below.

Table 1.1: Study Timeline

11/01/2004	Request for Proposal (RFP) from Caltrans.
02/01/2005	Research proposal granted.
10/01/2005	Study starts, initial scope was creation of an open-source ATMS.
11/30/2005	Meeting for scope revision, target replacing database with open-source.
12/01/2005	NDA negotiation process started between UCD lawyers and Caltrans lawyers.
02/23/2006	First revised scope: the open-database—create an open-source version of the existing ATMS system database.
12/08/2006	Project Task 2 report, ATMS literature review [6].
01/03/2007	After 13 months, the UCD and Caltrans lawyers fail to reach agreement over the NDA that AHMCT software engineers were required to sign to gain access to the proprietary ATMS source code.
05/25/2007	Mn/DOT releases IRIS source code, version alpha 1, using the GNU GPL.
06/13/2007	Initial project meeting with D10 in Stockton.
06/23/2007	IRIS client and server building using the alpha 1 code release.
07/17/2007	Revised project scope that details implementation of IRIS within Caltrans.
08/20/2007	First AHMCT contribution to the IRIS code-base.
10/02/2007	Requirements Definition Document complete [8].
10/04/2007	Key project team discussions about open-source licensing implications and issues.
10/10/2007	Mn/DOT IRIS alpha 2 code release.
10/24/2007	Initial Caltrans Traffic Operations and Research team meeting, downtown Sacramento.
12/20/2007	Researchers received 2 Model 170 (M170) controllers from Caltrans.
12/31/2007	Software Design Document complete [9].
02/04/2008	First AHMCT contribution in the mercurial repository.
05/20/2008	D10 IRIS Release 1: Developer release.
05/30/2008	D10 IRIS Release 2: Rudimentary CMS control, video monitoring, traffic monitoring.
06/18/2008	D10 IRIS Release 3: Internationalization, free form message entry, incidents displayed.
06/27/2008	D10 IRIS Release 4: CMS message via free-form and library entries.
07/10/2008	D10 IRIS Release 5: Travel time configured and ready for testing, Pan Tilt Zoom (PTZ) camera control.
07/21/2008	D10 IRIS Release 6: Free-form messages saved in library, numerous defects repaired.
07/23/2008	D10 IRIS Release 7: Fixed CMS message centering problem.
	<i>continued on next page</i>

	<i>Project timeline, continued</i>
09/20/2008	D10 IRIS Release 8: New Caltrans Automated Warning System (CAWS) driver, CMS control buttons.
10/06/2008	D10 IRIS Release 8.1: System Attributes, CMS on dial-up lines supported, Roadway tab configured.
10/17/2008	D10 IRIS Release 8.2: Traffic station map improved, support for model 520 CMS.
10/20/2008	D10 IRIS Release 8.3: Enhanced CAWS functionality, video presets, video reliability improved.
10/23/2008	Project Demonstration in Stockton.
11/12/2008	D10 IRIS Release 8.4: CAWS now displays which CMS are activated, mapping improved with street names, CMS preview capability.
11/21/2008	IRIS Demonstration to Division Chief, Traffic Operations.
11/24/2008	D10 IRIS Release 8.5: System Attribute enhancements, initial Quick Message library, new traffic stations, CAWS enhancements.
12/05/2008	D10 IRIS Release 8.6: Logging enhancements, real-time Keyhole Markup Language (KML) output for CMS.
12/09/2008	D10 IRIS Release 8.7: Easy access to cell modem re-initialization page, more descriptive error messages.
12/10/2008	Project Demonstration in Sacramento, including Division Chief, Research & Innovation.
01/07/2008	D10 IRIS Release 8.8: Ability to send a single message to multiple CMS, ability to blank multiple signs, enhanced CAWS functionality.
03/12/2009	D10 IRIS Release 8.9: SV170 3.4 support, enhanced logging, CAWS validation, context sensitive help system, numerous defects repaired.
06/16/2009	D10 IRIS Release 9.0: ability to send messages to multiple signs (ad-hoc and existing sign-groups), CAWS sign grouping, support for specifiable page on-time, completion of RMI to SONAR conversion, fonts specified on a per-page basis, smart alpha-numeric sorting, numerous other User Interface (UI) improvements, and defect repairs. Over 1000 change sets were added compared with R8.9.
06/21/09	D10 IRIS Release 9.0.1: enhanced logging, defect repairs.
06/24/09	D10 IRIS Release 9.0.2: new roles now have multiple permissions per role, bitmap images can now be sent to signs, defect repairs.
08/10/09	D10 IRIS Release 9.0.3: 50% faster map loading, defect repairs.
08/12/09	D5 IRIS Release 9.0.3. initial install in D5.
08/26/09	D1 IRIS Release 9.0.4: initial install in D1, defects repaired.
09/03/09	D1 IRIS Release 9.0.5: initial install in D10, defects repaired, update to IRIS V3.96.
10/28/09	D1, D5, D10 IRIS Release 9.0.6: defects repaired, permissions for different classes of users.
11/19/09	D1, D5, D10 IRIS Release 9.0.6a: defect repaired.

## 1.5 Project Documentation

For additional insight and background, the reader may consult the documents listed below and in the References section on page 70:

- Study reports and documents:
  - Project Task 2 report, ATMS literature review [6],
  - Project Task 3 report, D10 operations and equipment [7],
  - Project Task 4 report, functional requirements [8],
  - Project Task 5 report, IRIS review [9],
  - Project Task 6 report, Overview of Caltrans D10 IRIS Demonstration Design [10],
  - Implementation and Extension of the IRIS Open-Source Traffic Management System to Improve Organizational Performance, poster session at the Transportation Research Board (TRB), January 2009 conference [12],
  - IRIS Verification Plan Outline and Procedure for the California Department of Transportation D10 [11, 5].
  - Open ATMS proposal: Research & Development of Open-Source Advanced Traffic Management System Hardware and Software Components [2].
- Web Resources
  - This study's web site: <http://iris.ahmct.ucdavis.edu>,
  - IRIS JavaDoc source code documentation [17],
  - IRIS source code documentation: some modules contain additional documentation. See the doc sub-directory within each module's repository [18],
  - Caltrans D10 IRIS Maintenance Manual [4],
  - Caltrans D10 IRIS Source Repository [1],
  - Mn/DOT IRIS Source Repository [16].
- Mn/DOT Documentation
  - Intelligent Roadway Information System (IRIS) As Built System Design Document [15].
  - IRIS User Manual, April 2008.
  - IRIS Administrator's User Manual.
  - IRIS Open Source Software Release Research Project Review of Candidate Licenses, March 2007.
  - Minnesota Intelligent Roadway Information System (IRIS) High Level Summary.
  - IRIS Software Verification Program Concept Paper Draft, January 15, 2007.
  - IRIS Open Source Software Circulation and Licensing Feasibility Study, Final Report, October, 2007.
  - Open Source General Public License Traffic Management Software, Sharing TMC Software Using an Open Source General Public License Software. Paper for Intelligent Transportation System (ITS) America conference.



# Chapter 2

## Method

### 2.1 Study Management

This section discusses some of the general concepts, processes, and tools used to manage the study.

#### 2.1.1 Software Engineering

At the highest level, the development process was split into two stages, with slightly different requirements management processes. The first stage corresponds to Phases 0 - 4 in the ITS project life-cycle, shown in Figure 2.1. This consisted of requirements definition, design, and the initial implementation within each functional area.

The second project stage used a cyclic development model and corresponds with Phase 4 of the ITS project life-cycle (see Figure 2.2). Existing features were refined, defects were repaired, and new requirements were generated by users, developers, and management based on hands-on use of IRIS.

The two-stage approach was particularly effective—the initial set of requirements were high-level and defined prior to hands-on IRIS use. For developers and operators, the knowledge and experience gained through Stage 1 hands-on use of IRIS was invaluable for Stage 2 requirements definition—most of the Stage 2 requirements would have been very difficult (or impossible) to define during Stage 1.

#### 2.1.2 Project Coordination

Monthly meetings were used to demonstrate monthly progress in each functional area. This was particularly valuable during Stage 1 development. Weekly meetings were used to discuss TMC needs, new and existing requirements, ongoing defects, and the release

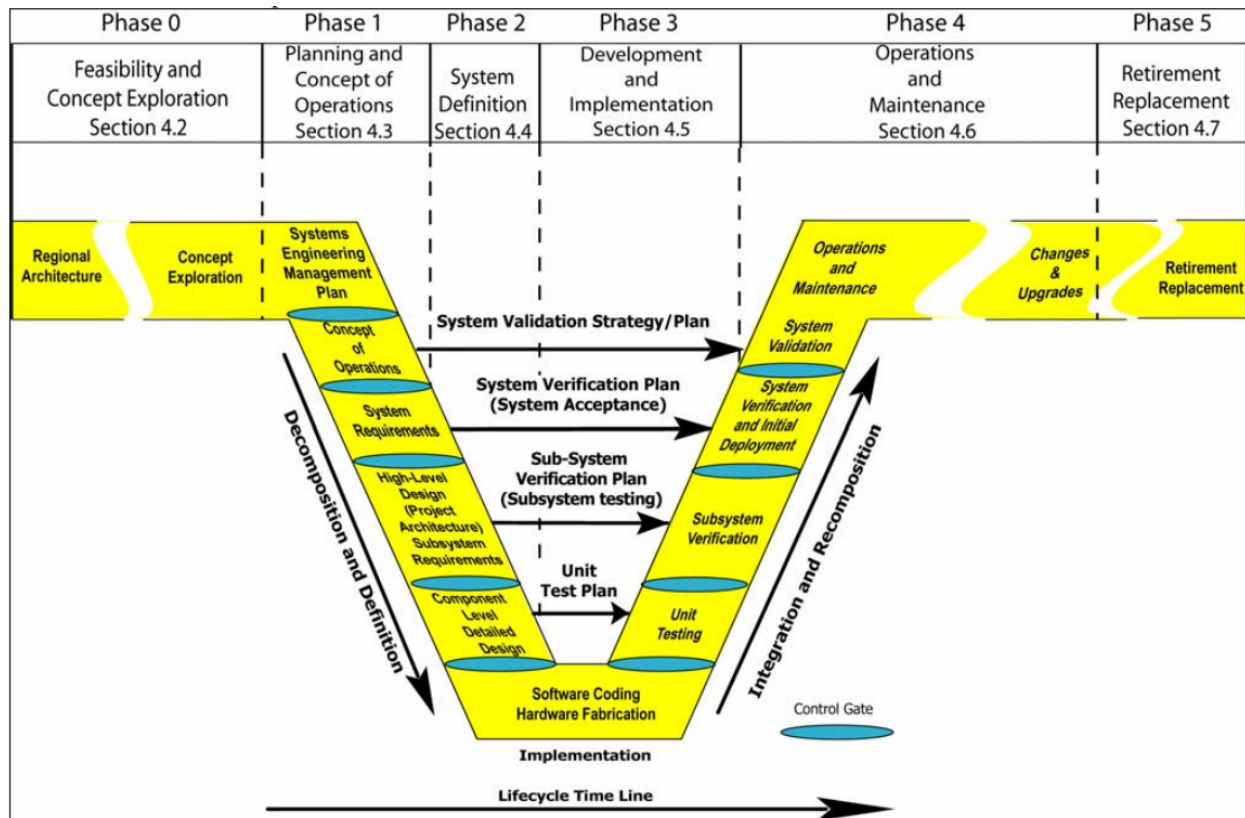


Figure 2.1: ITS Project Life-cycle Phases and Life-cycle Tasks [20]

schedule. Weekly meetings involved TMC administrative staff, Caltrans management, and development staff, and were particularly valuable during Stage 2 development.

### 2.1.3 Project Management and Communication Tools

The use of a project web site was indispensable for project communication, organization, and progress tracking. The project web site was implemented using a wiki<sup>1</sup>, which provided the ability to create both public and non-public web pages. Access to non-public pages was controlled with user names and passwords. Later in the project, a ticketing system<sup>2</sup> was used for managing defects, enhancements, and the testing process (see also section 3.4). Users created trouble tickets when defects were discovered. The wiki and ticket system provided:

- Access control to non-public content and documents.
- IRIS help system pages: a context-sensitive help system was added to IRIS that is accessible via the F1 key. Pressing the F1 key in IRIS loads context-specific pages from the wiki.

<sup>1</sup>See <http://www.mediawiki.org> for MediaWiki.

<sup>2</sup>See <http://trac.edgewall.org> for Trac.

- The IRIS Maintenance Manual: an on-line manual and consists of a series of wiki pages integrated with the maintenance manual.
- Document reference and storage: prior versions of documents were automatically saved. Procedures, technical documentation, reports, etc. were saved or referenced in the wiki.
- Creation and management of trouble tickets: using Trac, trouble tickets were created by end-users, administrators, and software engineers. Trouble tickets are linked with requirements and were used to generate and track new requirements. Trac was used to generate management reports and monitor overall project progress.
- Contact information for all project staff.
- Development progress: to-do lists, task completion lists, release notes, problem lists, etc.

### 2.1.4 Release Scheduling

Between May of 2008 (release 1) and January of 2009 (release 8.8) there were 16 releases, averaging a new release every 15 days. This short cycle time resulted in a rapid rate of improvement experienced by IRIS users, both in quality and functionality. This encouraged users to suggest improvements and increased their confidence in the system.

After the 9.0 release in June of 2009, the focus shifted from rapid improvement to maintenance, focusing on defect repair, stability, and increasing the predictability of releases and improvements. The research project team transitioned into a software Change Control Board (CCB) team, focused on improved software engineering processes.

### 2.1.5 Focus on Reliability

An explicit focus was placed on fixing defects first and rapidly. As development progressed, the defect discovery rate for application declined, which is shown in Figure 5.12 for the Sensor Server application. The development team encouraged users to immediately report suspected problems. User feedback was an important source of new feature suggestions. For example, any user confusion over the meaning of button labels resulted in relabeling. Rapid defect repair was also important for increasing user confidence in the new system.

Logging of anomalous conditions was invaluable for reconstructing subtle and intermittent problems, finding defects, detecting anomalous conditions, and verifying that code was executing as intended. The fast-fail approach for detecting and reporting internal problems was used extensively [22]. The design-by-contract approach using assertions, pre-conditions, and post-conditions was also used extensively during development. The combination of logging, fast-fail, and assertions enabled early problem detection.

## 2.2 Requirements Management

### 2.2.1 Stage 1: Deployment of Primary Feature Set

The first project stage corresponded with phases 0 - 4 in the ITS project life-cycle, shown in Figure 2.1. This consisted of requirements definition, design, and initial implementation within each functional area. The primary goal of this project stage was development and deployment of major system features within each functional area. For this stage, a top-down waterfall approach was used that defined an initial set of 100 coarse-grained functional and non-functional requirements[8]. These requirements were subdivided by prioritized functional area. Requirements definition was based on a survey of D10 systems [7] and limited knowledge of IRIS source code structure. The goal of these requirements was to loosely define functionality desired by IRIS users and TMC management. During this stage no attempt was made to add new requirements. A design for each of the major functional areas was created based on IRIS structure and consultation with Mn/DOT [9, 10]. This stage lasted through approximately release 4 in late June of 2008. This transition point can also be seen in Figure 5.12, in which the rate of change of code committed to the repositories slowed, as major features were completed.

### 2.2.2 Stage 2: Iterative Refinement and Extension

The second project stage used an Iterative and Incremental Development (IID) model to continuously refine and add features that were generated by users, engineers, and management from hands-on IRIS use (see Figure 2.2). This stage is characterized by a bottom-up approach using stakeholder involvement, elicitation, and acceptance.

The primary goal of this stage was user acceptance of IRIS as a replacement for the existing CMS control application (Satellite Operations Center Command System (SOCCS)). Achieving this goal was a function of: 1) developing feature parity with SOCCS, and 2) providing excellent reliability.

During this stage, 116 additional requirements were defined, 43 (37%) of which are future requirements, for use in subsequent project phases.

## 2.3 Cooperative Development Process

The cooperative software development process between AHMCT and Mn/DOT was a crucial aspect of the project for the following reasons:

- *Understanding the existing design:* communication with Mn/DOT was important for understanding the intent, future direction, and shortcomings of IRIS feature design.

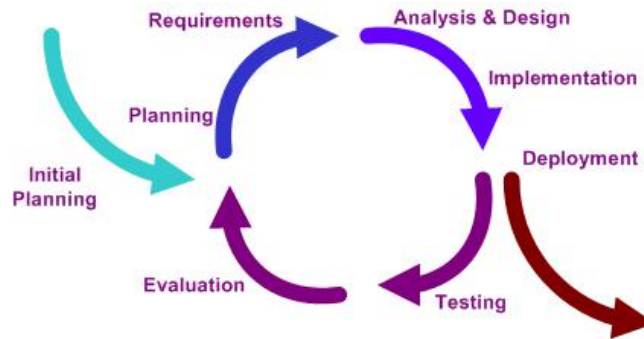


Figure 2.2: Iterative and Incremental Development Model (see Section 2.2.2)

- *Effective design of new features*: often required extending and generalizing existing IRIS functionality. Communication with Mn/DOT was essential for understanding their needs and creating enhancements that would satisfy Mn/DOT, Caltrans, and future users.
- *Resolving problems*: it was almost always more efficient to discuss problems with Mn/DOT before spending effort fixing them. Often, Mn/DOT was aware of the problem and had either 1) fixed it already, 2) anticipated that it would be fixed in subsequent releases due to a dependency that had been fixed, 3) were aware of the problem and were happy if AHMCT fixed it, or 4) were not aware of the problem. Typically, AHMCT attempted a repair in the last two cases.
- *Effortless code merges*: agencies contributing code to IRIS are strongly motivated to make the code merge process for Mn/DOT as effortless as possible. Intelligent and thoughtful coordination increased the probability that code merges would be effortless for Mn/DOT.

### 2.3.1 Single IRIS Code-base

Caltrans IRIS and Mn/DOT IRIS are built from the *same* code-base. It is crucial to understand that *all agencies share a single code-base*. Sharing a single code-base leverages limited resources for development and testing. Functional differences between IRIS running in different agencies is primarily due to configuration differences. The official IRIS release is maintained by the lead software engineers at Mn/DOT. However, each agency maintains their own (or multiple) source code repositories, as discussed in the next section.

The existence of one code-base does not imply that there is a single source code repository—in fact, a distributed source control system is used, in which many source repositories can be spread across multiple organizations or feature sets. This approach has numerous benefits (see below).

## 2.3.2 Distributed Source Repository Management System

The IRIS project uses a distributed source code repository system.<sup>3</sup> A *distributed repository* is structured so that each developer has an independent and complete repository. Fragmentation of the source code-base into separate and incompatible versions of IRIS does not happen because each IRIS developer that makes source code changes forwards those changes back to the lead engineers at Mn/DOT. Mn/DOT then incorporates these changes into their repository and subsequently releases new versions of IRIS to other developers.

A *change-set* is a discrete bundle of source code changes (which may span multiple files) and is typically associated with a single enhancement or defect repair. Developers typically 1) create enhancements and repairs, 2) commit them to the repository as change-sets, and 3) forward these change-sets to other developers through a repository *pull* process. The centralized repository concept of "checking-out" source files does not apply to the distributed source control model. The centralized model uses a cycle of: check-out, modify, and check-in. The distributed model uses a cycle of pull, modify, commit. The next section discusses the development process further.

## 2.3.3 Typical Development Sequence

The cooperative development process between AHMCT and Mn/DOT typically followed the steps discussed below and diagrammed in Figure 2.3.

- *Step A:* Development begins with AHMCT pulling (reading) from the public Mn/DOT IRIS repository shown as arrow A in Figure 2.3. The pull process imports all new change-sets that have been published by Mn/DOT since the last pull.
- *Step B:* AHMCT then adds new features, fixes defects, and enhances existing features. Each enhancement or defect repair is saved as a single change-set in the local AHMCT development repository. After enhancements are tested and judged to be complete, they are published in the public AHMCT IRIS repository. Typically this occurred approximately every two weeks. The development of enhancements and new features is substantially coordinated with Mn/DOT. This sometimes involves daily email and phone conversations discussing design decisions, problems, alternative approaches, etc. The AHMCT engineer's overriding objective is to produce code that is easy for the lead engineers at Mn/DOT to merge into the main IRIS repository (Step D below). This is easily achieved if enhancements are well designed, generalized, well tested, and conform to the existing IRIS code syntax.
- *Step C:* After a set of desired IRIS enhancements are complete, a new Caltrans IRIS release is 1) built, 2) tested by developers, 3) tested by users (acceptance testing), and 4) installed on the production system.

---

<sup>3</sup>See <http://www.selenic.com/mercurial> for the source management system used by IRIS.

- *Step D*: When Mn/DOT is ready, they pull (read) change-sets from the public AHMCT IRIS repository and merge it with their IRIS repository. High-quality code that conforms with or elegantly extended the existing IRIS design is typically easy to merge.
- *Step E*: When the lead engineers at Mn/DOT accumulate a significant number of enhancements, they release a new version of IRIS and publish it in their public IRIS repository.
- *Step F*: Mn/DOT installs a new version of IRIS in their TMC.

## IRIS Source Code Management

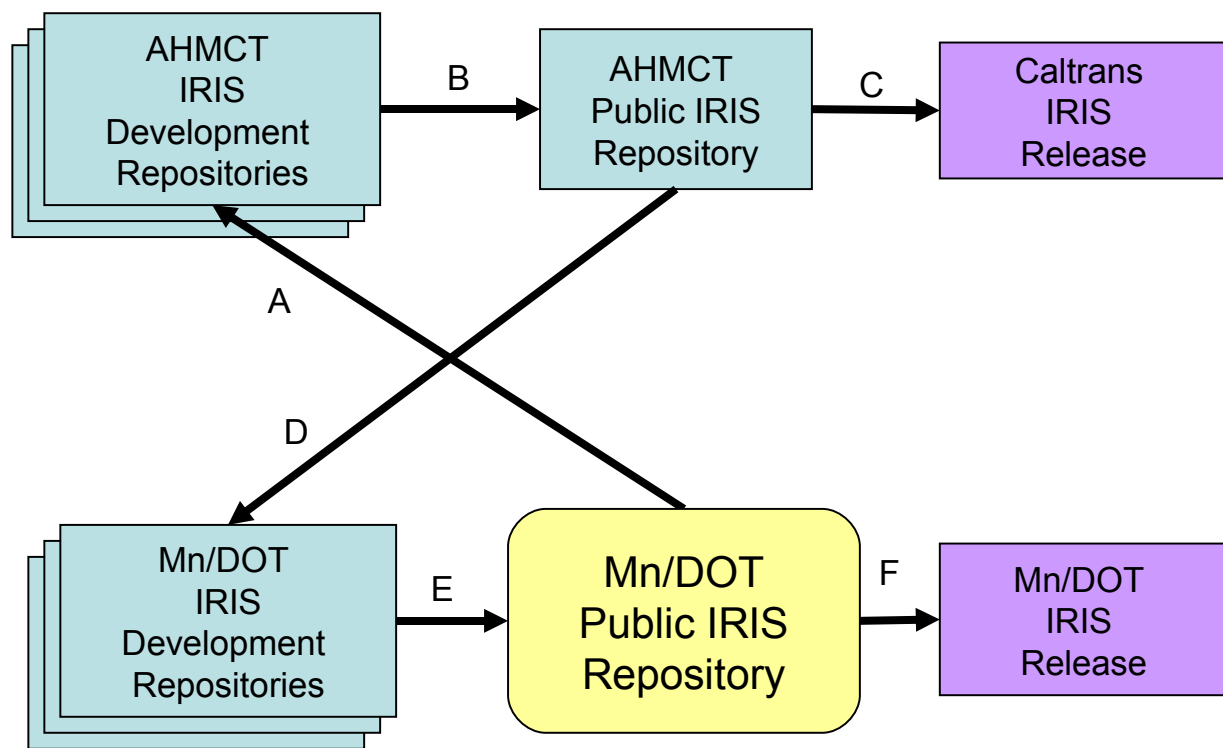


Figure 2.3: IRIS Source Code Management (see pg. 14)





## Chapter 3

# Testing, Validation, and Verification

Validation and verification were both performed. The *validation* process determines if the right enhancement was built (*are we building the right thing?*). Validation depends on user needs and is defined through discussion with users and requirements definition. Validation is performed against the defined requirements and through user acceptance tests.

The *verification* process determines if the enhancement is built according to specifications. Verification depends on requirements definition, unit tests, developer testing, and user acceptance tests. The ultimate form of verification is long-term use of the feature or system in the production system.

### 3.1 Test Plan

Testing and verification were both performed as follows:

- *Automated unit test cases*: were used. See Section 3.2.
- *Unit testing during development*: was used during development of new and enhanced features. This effort was part of the development process.
- *System testing during development*: was used extensively during development of new and enhanced features. CASPER was used to simulate multiple CMS (see Section 5.5.1 for CASPER). The real-time traffic and incident feeds were also used for testing. The production cameras were used for system testing. This effort was part of the development process.
- *Logging*: was used to detect anomalous conditions during development and in production. This effort was part of the development process. Logging was invaluable. See Section 2.1.5.
- *New-release system testing*: new releases were tested on the test and development machines. This effort was part of the development process. See Appendix B for

the procedure. In addition, a verification plan outline was developed for future use [5, 11].

- *Automated Warning System (AWS) verification*: substantial CAWS verification was performed. See Section 3.6.
- *Formal user acceptance*: test cases were used. See Section 3.3.

System and feature validation were performed as follows:

- *Requirements discussion*: before requirements were implemented, features were discussed with TMC administrators and end-users. This was particularly important for user-interface enhancements.
- *Feature prototyping*: some features (e.g. CAWS enhancements) were prototyped and shown to TMC staff. This was important for complex features with a strong user-interface dependence.
- *User acceptance*: the ultimate system validation was through user acceptance and use of IRIS. The existing CMS control application (SOCCS) was always available for use. The goal was voluntary user migration to IRIS as a result of enhanced reliability, features, and efficiency.
- *Formal user acceptance*: test cases were used. See Section 3.3.

Building and verifying a new IRIS production release typically follows these steps:

1. On the test or development server, performed by a software engineer: automated unit tests, ad-hoc integration testing, predefined test cases (e.g. CAWS test cases), and user acceptance test cases.
2. On the test server, by a software engineer: ad-hoc integration and unit tests which are typically a function of improvements in the new release.
3. On the test server, by a District user: end-user acceptance tests.
4. On the production server, by a software engineer or administrator: the procedure defined in the appendix—see Section B.
5. On the production server, by a software engineer: some tests may need to be executed which are difficult (or impossible) to execute on the test system.
6. On the production server, by a District user: some tests may need to be executed which are difficult (or impossible) to execute on the test system.

## 3.2 Automated Unit Test Cases

Automated test cases are used during development and provides a mechanism for software engineers to execute numerous test cases and automatically determine if each test

passed or failed. As of January 2010, the IRIS code-base had 481 automated test cases and the Caltrans IRIS applications (CASPER, Sensor Server, Trafserver), contain an additional 470 automated test cases. The development of these test cases was integral to the development of the associated code. This approach is often identified as Test Driven Development (TDD).

Presently, automated test cases are used primarily to unit test classes that provide basic functionality to other classes (convenience classes). Test cases were constructed for static methods. This was particularly effective for convenience classes because many other classes and methods depend on them for results. Automated unit test cases are also useful for regression testing.

IRIS uses the JUnit<sup>1</sup> test case framework for automated tests. These test case are run by software engineers during the development process. The example below shows two test cases related to the NTCIP Dynamic Message Sign (DMS) protocol. The first test case should fail, and the second test case should pass. If either of these test cases does not perform as expected, the system will halt automated testing.

```
assertFalse(MultiString.isEquivalent("[fo2]LINE1", "[fo1]LINE1"));
assertTrue(MultiString.isEquivalent("LINE1[nl][nl]", "LINE1"));
```

### 3.3 User Acceptance Tests

User acceptance tests were used for verification and validation. Test cases were developed by software engineers and executed by both district users and developers. Test cases were developed for each defect or enhancement. User acceptance of a new release was dependent on all test cases for a release successfully passing. The total time required for all personnel for user acceptance testing, per release, was typically less than 12 hours. A sample user acceptance test case is shown here:

- *Ticket Number:* UNR\_270
  - *Ticket description:* Defect, sensorserver is returning the wrong error message when the remote phone line is busy. Operators need to know the current status of field hardware so problems can be diagnosed.
  - *IRIS Version:* 9.0.6
  - *Test case description:* verify error message is correct when the field modem is busy.
  - *Test procedure:*
    1. Have an admin configure IRIS to call a local phone that is off the hook (so it will be busy).
    2. Send a get-message request to the CMS.

---

<sup>1</sup>For JUnit, see <http://www.junit.org>

3. At the completion of step #2, verify that the error message returned states that the remote modem is busy. This will be in the 'Current Operation' field.
- *Test case executed:*
    - \* MD pass, 10/27/2009, 9.0.6, software engineer
    - \* MB pass, 10/29/2009, 9.0.6, D10
    - \* JD pass, 11/2/2009, 9.0.6, D1

## 3.4 Defect Tracking

Defect tracking and logging evolved significantly over the course of the project. Initially, defects were loosely tracked via a task log on a project wiki page. This method was used for approximately 12 months. During this time approximately 200+ significant code defects were repaired. An equal number of minor defects and problems were fixed.

Starting in March of 2009, a software ticketing system was used (Trac<sup>2</sup>) to log defects, enhancements and tasks. This ticketing system is maintained by Caltrans and facilitates communication within Caltrans and between Caltrans and AHMCT. This ticketing system tracks Caltrans-specific tasks, requirements and defects.

Starting in June of 2009, an additional ticketing system (also Trac) was deployed. It is used primarily to facilitate communication between IRIS software engineers in different agencies. For example, between AHMCT and Mn/DOT. The use of these ticketing systems substantially improved defect and task tracking, reporting and communication.

## 3.5 Testing and Verification Results

The end result of the strong focus on reliability, combined with testing, verification, and defect repair was a reliable mission critical ATMS. For CMS operations, during 9 months of 24x7 TMC operation, there were 5 service outages. One was due to a configuration error, three were due to D10 networking problems, and one was due to a Sensor Server defect.

## 3.6 Automated Warning System Verification

CAWS is an automated warning system developed by Caltrans, that uses data from RWIS (wind speed, visibility), combined with speed sensor data, to automatically generate CMS messages. IRIS sends these messages to CMS on 30 second intervals. The following test process was used to verify IRIS CAWS functionality:

---

<sup>2</sup>See <http://trac.edgewall.org> for Trac.

1. Automated unit tests: were used for testing at the method level.
2. Lab testing of CAWS functionality: a test matrix of 10 different message types (100+ test cases) was used to test message overwrite functionality. For example, one test case verifies that an operator message does not overwrite a CAWS generated message. Developer effort to execute these test cases was approximately 6 hours, per release.
3. End-to-end verification: a test-bed was used to compare actual IRIS message output with the real-time CAWS generated messages over the course of 4 weeks. Approximately 1400 CAWS messages were sent to simulated signs. The CASPER field controller simulator and the watchdog application (see Section 5.5.6) were used for this purpose. Logged messages were then compared with expected messages. As a result of the validation process, one subtle defect was discovered and fixed before it occurred in production. Total testing and development effort for this verification was approximately 4 weeks.
4. Incremental IRIS control: IRIS was initially used to control a single CMS in the TMC for approximately one week. This provided operators with an opportunity to gain experience with the new UI and administrators the opportunity to verify the new CAWS functionality was operating as expected.
5. Full IRIS control: all CAWS CMS were enabled for full IRIS control.
6. Operator acceptance tests: were developed late in the project and verified by IRIS administrators and users. These involved generated test messages deployed on simulated CMS.

### 3.7 IRIS Scalability Testing

To quantify IRIS scalability, testing was performed with 50 simultaneous IRIS clients. Memory and processor utilization were monitored every 60 seconds during the test. The results are shown in Figure 3.1.

Excluding the mapping issue discussed below, there are no known scalability issues that would limit IRIS use in small, large, or very large transportation agencies. Based on the test results below, a very large transportation agency could effectively run IRIS on a \$1,000 commodity server with no performance degradation.

The one known IRIS scalability issue limits client map sizes to approximately 5 Megabytes, which is a relatively small map. This limits the number of lines on the map, not the size of the geographic area the map covers. The work-around for this limitation is to simplify the map (using line simplification) to eliminate unnecessary line segments, which reduces the map size. This limitation is due to the use of a non-scalable data structure in the IRIS client code. Mapping enhancements have been discussed and are desired by Caltrans and Mn/DOT.

The scalability testing used a \$1,000 commodity x86 server with modest capabilities: an Intel Core Duo (2 cores), running at 2.4 GHz, with 2 Gigabytes of memory. In March

of 2009, a new machine with similar capabilities costs less than \$1,000. The server was executing:

- Apache web server,
- CASPER field simulator,
- Sensor Server,
- IRIS server,
- IRIS video servlet,
- Lightweight Directory Access Protocol (LDAP) directory server,
- PostgreSQL database,
- Apache Tomcat server,
- Traffic Server.

Total memory usage across all applications with 50 simultaneous users was 811 MBytes. Memory usage within the IRIS server application with 50 users was less than 140 MBytes and was proportional to the number of users ( $O(n)$ ). Total processor utilization was very modest at less than 20 percent. From these results, it appears that a single relatively low-end IRIS server will easily support substantially more than 50 simultaneous users.

### 3.8 Future Testing and Verification

Comprehensive testing and verification are crucial for defect discovery, both in the development process and during verification before deployment. Looking towards the future, enhancing IRIS' test and verification capabilities is crucial, particularly as more agencies become involved. Improving IRIS testing and verification capabilities would be very beneficial 1) during developing, 2) for verification before deployment, and 3) during deployment. The following testing-related features are desirable:

- *Automated end-to-end test cases*: would enable automatic testing of existing functionality by executing end-to-end operations (e.g. sending a CMS message to a simulated CMS) and verifying the results. This would be useful for developers following completion of new features and enhancements and merging code from other developers. New features would require new or modified test cases. Release builds could be regression tested prior to release. Presently, end-to-end testing is performed manually by developers and user acceptance tests.
- *Simulated traffic*: the ability to feed IRIS with simulated traffic would enable testing of traffic dependent functionality, such as travel time generation, Vehicle Detector Station (VDS) status, and ramp meter functionality. This would enable regression testing in these areas. The ability to "capture" live traffic and feed it into the traffic simulator would also be useful for debugging problems and possibly for operator training.

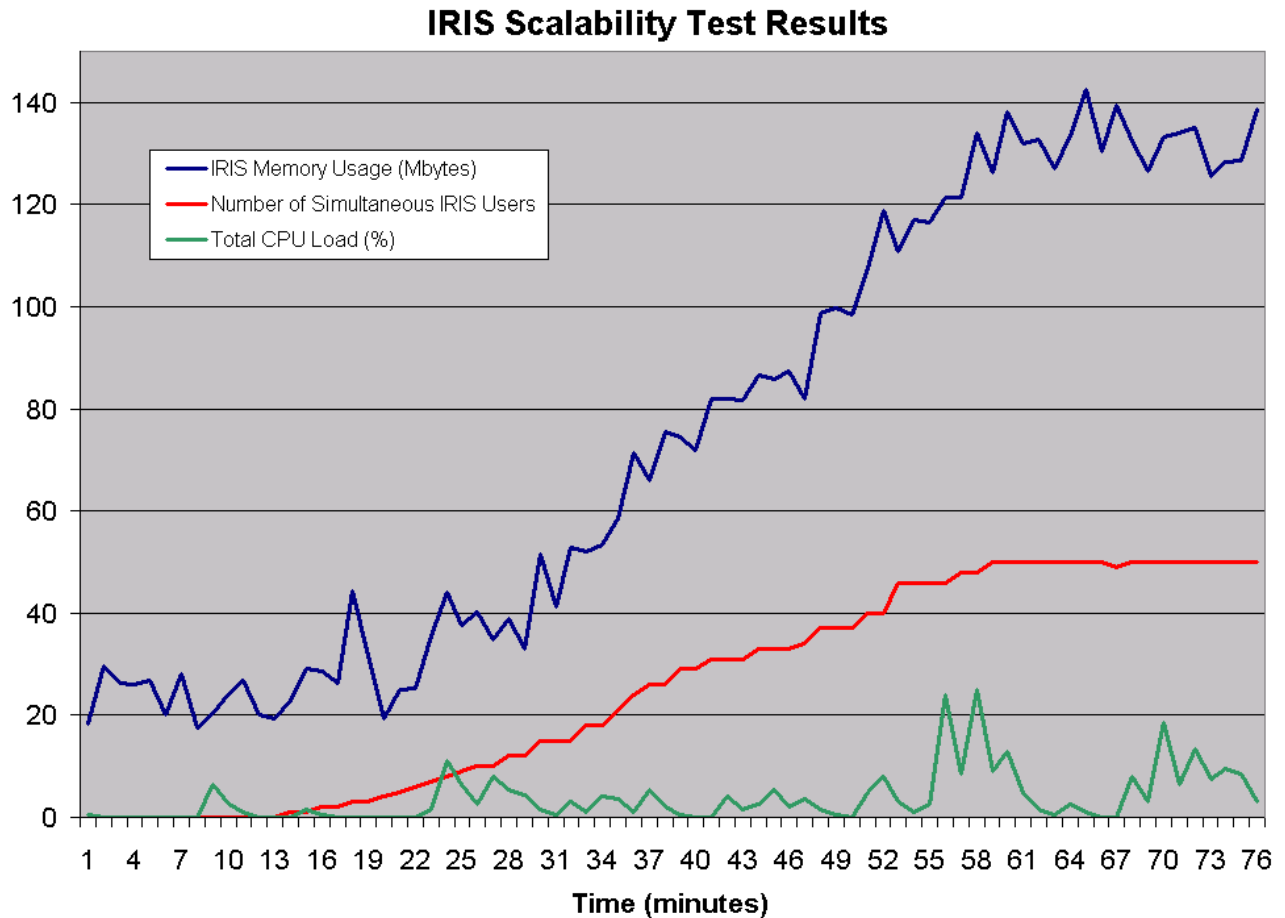


Figure 3.1: IRIS Scalability Test Results (see Section 3.7)

- *Simulated incidents:* may be useful for supporting future features such as incident management.
- *Real-time verification:* would be useful for notifying operators of critical system failures in real-time.
- *District 3 NTCIP sign testing:* was performed but results were inclusive. District 3 has a small number of CMS that support the NTCIP communications standard (or a variant). One of these signs was briefly tested with IRIS and results were inconclusive: either the CMS in question does not support the NTCIP class A, B, or C standards, or there was a serial server communication problem. Further testing is required.





## Chapter 4

# IRIS Portability

### 4.1 Relevance of Portability

Implementing an open ATMS in a transportation agency at low cost is a function of portability. A high degree of portability keeps implementation costs low. Software is *portable* to a new agency to the extent that the effort to transport and adapt it is less than the cost of redevelopment [19].

### 4.2 IRIS Modularity

Modularity is an important aspect of portability. Modular designs localize dependencies and facilitate testing, defect repair, and portability. IRIS binary modules are shown in Figure 4.1. Each binary module shown in the future is created from a set of source code modules (name-spaces). For example, the binary IRIS client module consists of many source code modules such as: camera, detector, dms, incidents, roads, security, widget, etc.

### 4.3 IRIS Clients Written In Other Software Languages

The IRIS server uses the Simple Object Notification And Replication (SONAR) communication protocol to communicate with the client. IRIS clients using the SONAR protocol can be written in other programming languages. For example, for testing purposes Mn/DOT developed a Python SONAR client. In the future, the development of IRIS clients in other languages on other platforms may be useful.

SONAR is a communication protocol developed by Mn/DOT as a replacement for Remote Method Invocation (RMI), the original communication protocol used in IRIS.

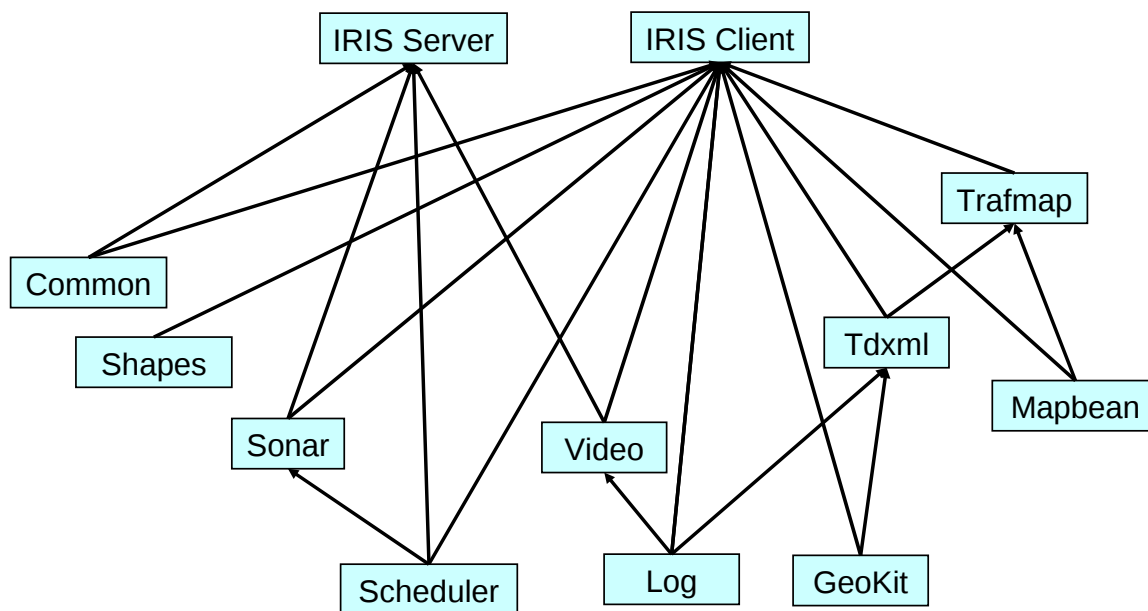


Figure 4.1: IRIS Binary Modules

SONAR was developed to address reliability, configurability, security, and performance issues with RMI. The Mn/DOT conversion of all IRIS RMI code to SONAR code was complete in June of 2009 (version 9.0) and was an important milestone for IRIS. See also Section 5.5.4 and the time-line in Section 1.4.5.

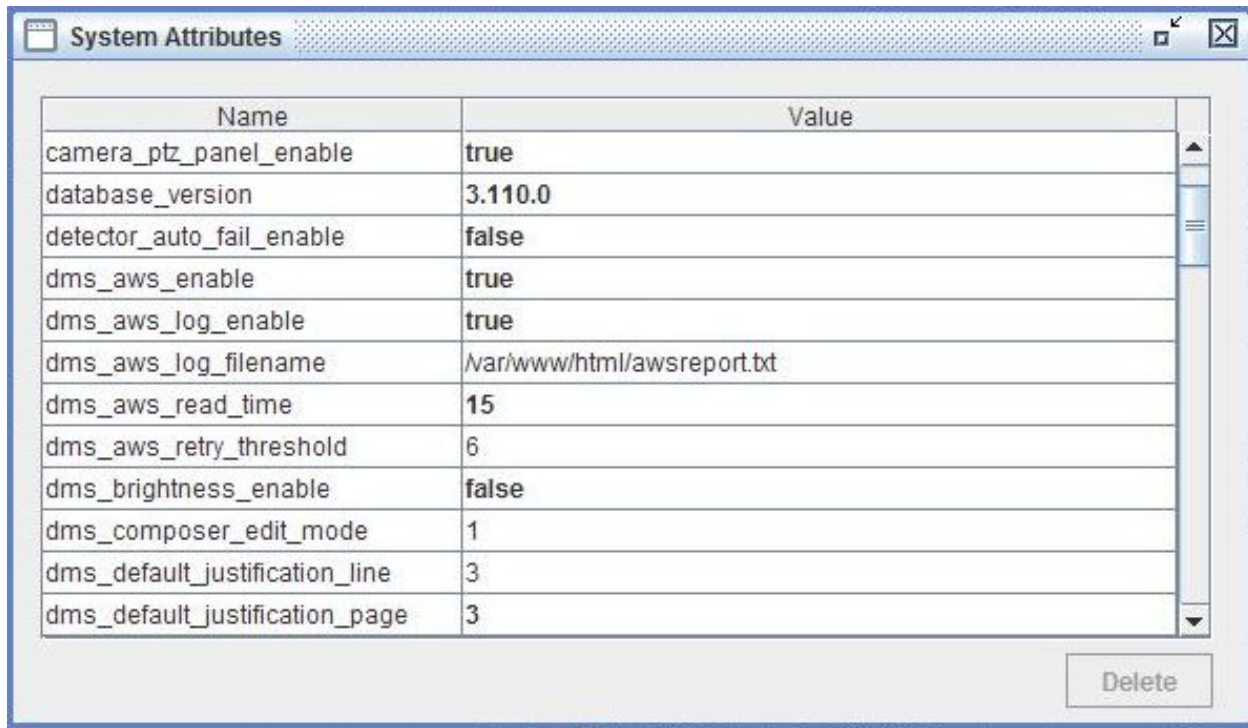
## 4.4 Configuring IRIS for an Agency

Configuring IRIS behavior and appearance for an agency is achieved with system attributes, internationalization, properties files, user permissions, and device drivers. The help system may also be customized per agency. Ideally, IRIS source code contains no agency-specific code (or as little as possible), with customization and feature availability controlled with settings and attributes.

### 4.4.1 System Attributes

System attributes are the primary means of tailoring IRIS. Attribute values are stored in the database and editable at run-time using the editor shown in Figure 4.2. Attributes are edited by administrators and have a specified name, type, value, and description. A type may be an integer, floating point, boolean, string, or enumerated type. Examples

of attributes are communication time-out values, file locations, the HyperText Transfer Protocol (HTTP) address of the AWS message file, etc. An overall design goal is that all agency-specific behavior and configuration be specified with system attributes.



Name	Value
camera_ptz_panel_enable	true
database_version	3.110.0
detector_auto_fail_enable	false
dms_aws_enable	true
dms_aws_log_enable	true
dms_aws_log_filename	/var/www/html/awsreport.txt
dms_aws_read_time	15
dms_aws_retry_threshold	6
dms_brightness_enable	false
dms_composer_edit_mode	1
dms_default_justification_line	3
dms_default_justification_page	3

Delete

Figure 4.2: IRIS Attribute Editor Form (see Section 4.4.1)

## 4.4.2 Internationalization

Agency-specific nomenclature is defined in a single file that maps each standard term (e.g. 'DMS') to the agency-specific term (e.g. 'CMS'). Figure 4.3 shows a menu that contains the agency-specific term 'CMS' in the Caltrans release of IRIS. In the Mn/DOT release, the same menu shows the Mn/DOT term ('DMS'). Another example is 'alert' versus 'AMBER Alert'. The current message bundles define 50 agency-specific strings.

## 4.4.3 Properties Files

Properties files define agency specific run-time constants. These are specified by developers during the build process. They may also be modified at run-time by system administrators. For example, the Internet Protocol (IP) address of the LDAP server.

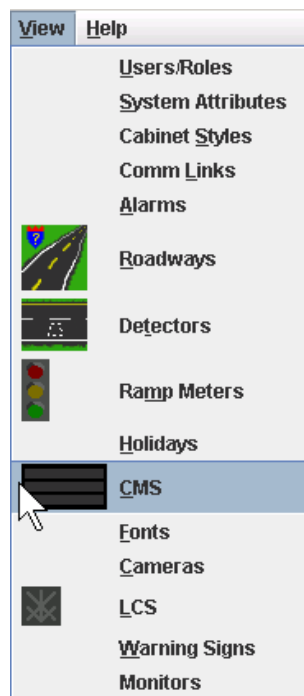


Figure 4.3: Internationalized IRIS Menu (see pg. 27)

#### 4.4.4 User Permissions

User permissions control access to features. Access is controlled through read, write, delete, and create privileges. The role configuration form is shown in Figure 4.4. Defined roles are assigned to each user.

 A screenshot of a web application window titled 'Users and Roles'. It has three tabs: 'Users', 'Roles', and 'Connections'. The 'Roles' tab is active, displaying a table with columns: Name, Enabled, Pattern, Read, Write, Create, and Delete. The table lists several roles and their associated permissions.
 

Name	Enabled	Pattern	Read	Write	Create	Delete
admin	<input checked="" type="checkbox"/>	*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
operator	<input checked="" type="checkbox"/>	dms/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
view	<input checked="" type="checkbox"/>	r_node/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	sign_group/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	sign_message/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	sign_group/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	sign_text/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	dms_sign_group/*	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	<input type="checkbox"/>	camera/*ptz	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
	<input type="checkbox"/>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

 At the bottom of the window, there are two buttons: 'Delete Role' and 'Delete Privilege'.

Figure 4.4: IRIS User Permissions Configuration Form (see pg. 28)

## 4.4.5 Interfacing IRIS with Devices and Software Systems

The driver interface connects IRIS with external hardware and software systems. The user interface is shown in Figure 4.5. IRIS contains open-source drivers for some common hardware devices, such as the Wavetronix SmartSensordriver [23], SmartSensor HD, and NTCIP class A, B and C drivers. These were developed by Mn/DOT. For a discussion of integrating IRIS with proprietary protocols, see Section 5.3.

Interfacing with external software systems is also possible through the driver interface. For example, IRIS integration with the CAWS system and with the proprietary Caltrans CMS protocol were implemented as drivers. See Section 5.7 for effort estimates for hardware drivers.

Comm Link	Description	URL	Status	Protocol	Timeout
L001	V1	iris.ahmct.ucdavis.edu:50001		DMS Lite	750
L002	CAWS	http://iris.ahmct.ucdavis.edu/d10cms.txt		NTCIP Class B	750
L003	V2	iris.ahmct.ucdavis.edu:50002		MnDOT 170 (4-bit)	750
L004	V3	iris.ahmct.ucdavis.edu:50003		MnDOT 170 (5-bit)	750
L005	Traffic1	iris.ahmct.ucdavis.edu:8111		SmartSensor 105	1500
L006	V4	iris.ahmct.ucdavis.edu:50004		Canoga	1400

Controller	Location	Drop	Active	Status	Error Detail	Version
ctl_92325	I5 SB S of French Camp Rd	1	<input checked="" type="checkbox"/>			Vx.xx
			<input type="checkbox"/>			

Figure 4.5: IRIS Device Driver Configuration Form (see pg. 29)

## 4.4.6 Help System

The help system loads a context-sensitive web page when the F1 key is pressed. Agencies specify which Uniform Resource Locator (URL) is loaded for each client form. This mechanism provides the ability for agencies to share generic help system pages with other agencies or develop customized help pages, depending on context. For example, the help page for the System Attribute form is likely to be agency specific, with instructions for administrators.

## 4.5 IRIS Operating System Portability

The IRIS client is a Java application and executes on computers running the Microsoft Windows and Linux operating systems. The Java Virtual Machine (JVM) must be installed. The IRIS client should run on Apple Mac machines without problems; however, to date, this has not been tested.

The IRIS server is a Java application and executes on computers running the Linux operating system. To date, Red Hat's Fedora<sup>1</sup>, Red Hat Enterprise Linux (RHEL)<sup>2</sup>, and CentOS<sup>3</sup> have been used. The Ubuntu<sup>4</sup> server distribution was tested early in the project—minor database incompatibilities were discovered. Using Ubuntu or other Linux distributions would require testing and (likely) some development effort. The PostgreSQL database, an integral component, runs on Windows and other platforms, and does not inherently limit IRIS portability.

---

<sup>1</sup>For Fedora see <http://fedoraproject.org>

<sup>2</sup>For Red Hat Enterprise Linux (RHEL) see <http://redhat.com>

<sup>3</sup>For CentOS see <http://centos.org>

<sup>4</sup>For Ubuntu, see <http://ubuntu.com>

# Chapter 5

## Results

The study implemented, extended, and enhanced the IRIS open-source ATMS within Caltrans D10. IRIS was developed and released as open-source by Mn/DOT in May of 2007. The Caltrans implementation is the first outside of Minnesota. The D10 TMC used IRIS in parallel with the existing CMS control applications (e.g. SOCCS) and supporting traffic management software. Results are discussed below.

### 5.1 TMC Operational Safety Enhancements

Safety enhancements for TMC operations are substantial and are discussed below.

#### 5.1.1 CMS Safety Enhancements

IRIS provides the following enhancements to TMC safety that the existing system (SOCCS) did not.

- *Periodic CMS polling*: IRIS automatically polls CMS on a periodic basis. The polling frequency is presently every 10 minutes and is specified with a system attribute. This time is expected to be decreased as D10 optimizes their wireless communication budget. This feature enables operators to determine at a glance what messages are presently on the signs. If a CMS power or communication failure occurred, operators would know something was wrong (within the polling frequency time) by glancing at the CMS status within IRIS. This increases TMC awareness and responsiveness to problems.
- *CMS communication network health reports*: three reports were developed to track network health over time on the wireless and dial-up networks. D10 personnel use these reports to monitor and diagnose network problems. The first report is a

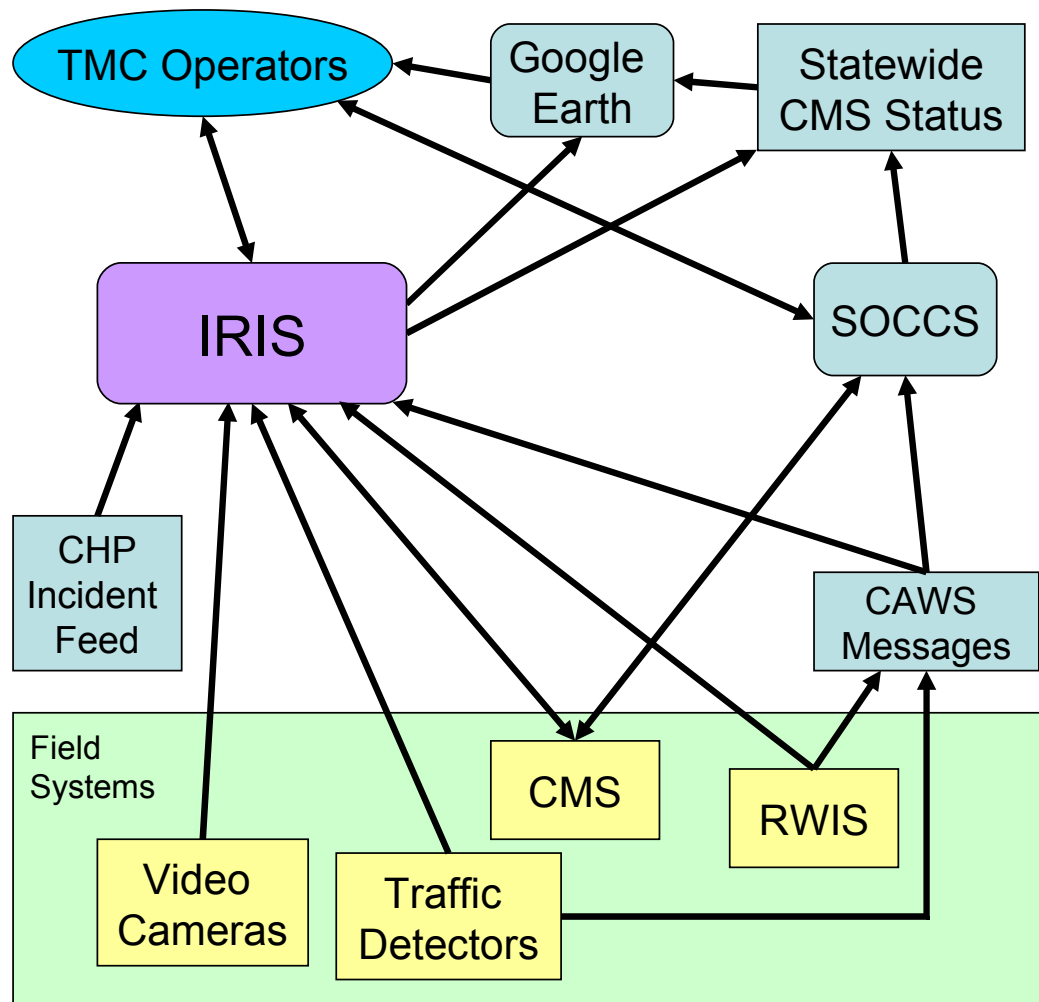


Figure 5.1: Caltrans District 10 Simplified Data Flow Diagram

graphical report (SignScope) and is shown in Figures 5.2 and 5.3. The second report (tabular) tracks communication operations in detail. The third report (tabular) tracks cumulative network error statistics over time.

- *Enhanced AMBER Alerts:* IRIS provides the ability for operators to send AMBER alert messages that optionally do not overwrite existing operator messages. This feature is implemented using the IRIS message priority scheme, in which lower priority messages do not overwrite higher priority messages. For example, operator messages have a higher priority than travel time messages, but lower than CAWS messages. CAWS and operator messages always overwrite alert messages. In the existing SOCCS system, when an operator sends a single sign message to multiple signs, it always overwrites existing sign messages.



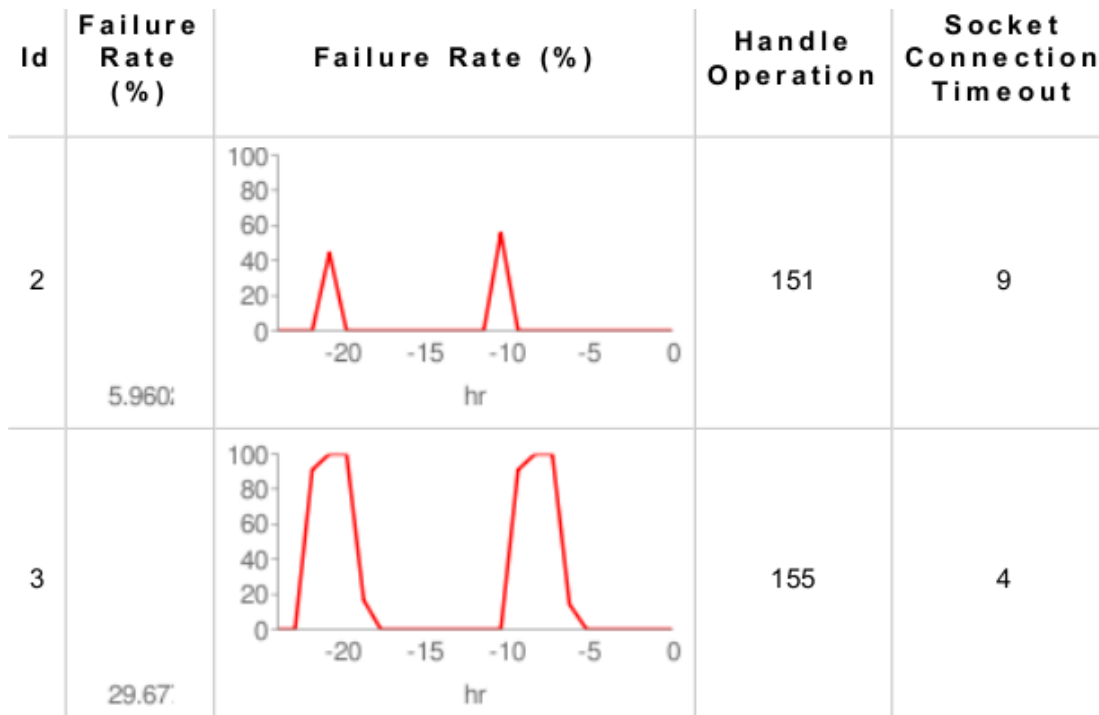


Figure 5.2: A Portion of the CMS Communication Network Health Report (SignScope)

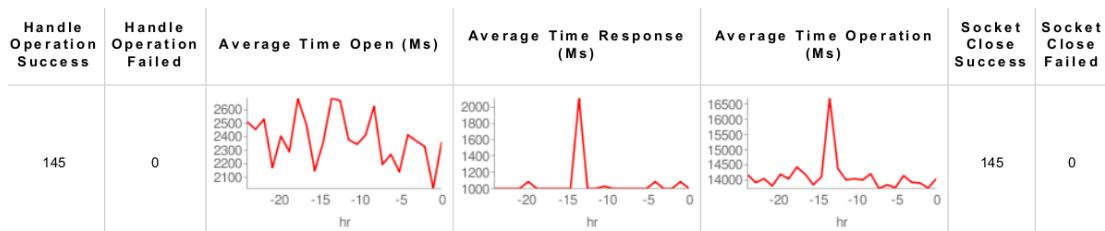


Figure 5.3: A Portion of the CMS Communication Network Health Report (SignScope)

### 5.1.2 Automated Warning System Safety Enhancements

IRIS integration with CAWS provides significant safety enhancements for D10 operations via a message priority scheme, user interface enhancements, failure notification, and testing.

#### IRIS Message Priority Safety Enhancements

The default IRIS behavior for overwriting deployed CAWS messages was changed compared with the existing D10 CMS control application (SOCCS). SOCCS allows operators to overwrite deployed CAWS messages without a warning message or other safety mechanism to alert the operator that they are replacing a CAWS message. For further information on the designed behavior of the SOCCS system see [3].

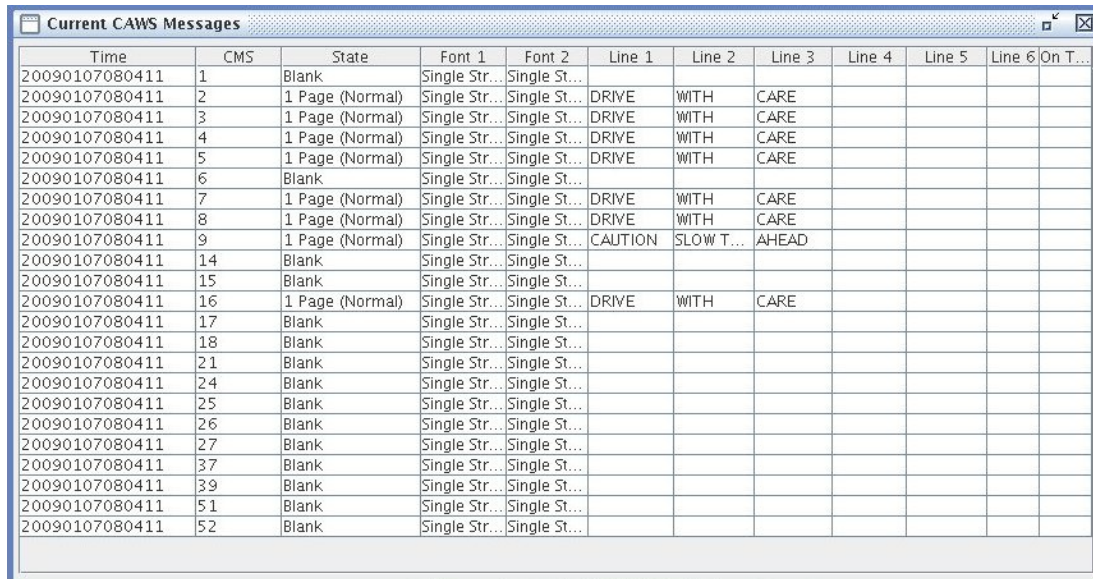
IRIS handles CAWS and operator messages differently than the SOCCS system. IRIS CAWS support was implemented using a message priority scheme that assigns a higher priority to CAWS messages than operator messages. This insures that new CAWS messages will overwrite existing operator messages and operator messages do not overwrite existing CAWS messages by default. An operator may only overwrite a deployed CAWS message if the following actions are taken:

1. The CMS must be deactivated from CAWS control using a check-box,
2. The CMS must be blanked, and
3. The new operator message must be sent to the CMS.

## CAWS User Interface Safety Enhancements

User interface enhancements were added to the IRIS client to provide visual safety information for operators:

1. CAWS toolbar: displays a list of operator deactivated CMS which is visible to all operators.
2. Current CAWS messages form: displays the current CAWS messages that should be displayed on activated CMS (see Figure 5.4). This form enables administrators and operators to verify that the correct CAWS messages have been successfully deployed (or blanked). The form is accessible via a button on the CAWS toolbar.



Time	CMS	State	Font 1	Font 2	Line 1	Line 2	Line 3	Line 4	Line 5	Line 6	On T...
20090107080411	1	Blank	Single Str...	Single St...							
20090107080411	2	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	3	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	4	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	5	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	6	Blank	Single Str...	Single St...							
20090107080411	7	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	8	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	9	1 Page (Normal)	Single Str...	Single St...	CAUTION	SLOW T...	AHEAD				
20090107080411	14	Blank	Single Str...	Single St...							
20090107080411	15	Blank	Single Str...	Single St...							
20090107080411	16	1 Page (Normal)	Single Str...	Single St...	DRIVE	WITH	CARE				
20090107080411	17	Blank	Single Str...	Single St...							
20090107080411	18	Blank	Single Str...	Single St...							
20090107080411	21	Blank	Single Str...	Single St...							
20090107080411	24	Blank	Single Str...	Single St...							
20090107080411	25	Blank	Single Str...	Single St...							
20090107080411	26	Blank	Single Str...	Single St...							
20090107080411	27	Blank	Single Str...	Single St...							
20090107080411	37	Blank	Single Str...	Single St...							
20090107080411	39	Blank	Single Str...	Single St...							
20090107080411	51	Blank	Single Str...	Single St...							
20090107080411	52	Blank	Single Str...	Single St...							

Figure 5.4: List of Real-time Automated Warning System Generated Messages

3. CMS categorization: the user interface provides a list of CMS that are marked as CAWS controlled and another list of all CMS that contain deployed CAWS messages. This enables the operator to rapidly determine the state of CAWS CMS, error states, and message state.

4. CAWS CMS map integration with real-time traffic: CAWS-activated CMS are displayed on the map, along with real-time traffic conditions. This is important for verifying traffic-related CAWS messages. Figure 5.5 shows a CAWS CMS activated by traffic, with the congested traffic detectors visible beyond the CMS. Figure 5.6 shows a CAWS CMS activated by incident-related traffic. The incident is visible on the map as the circle (near the CMS).

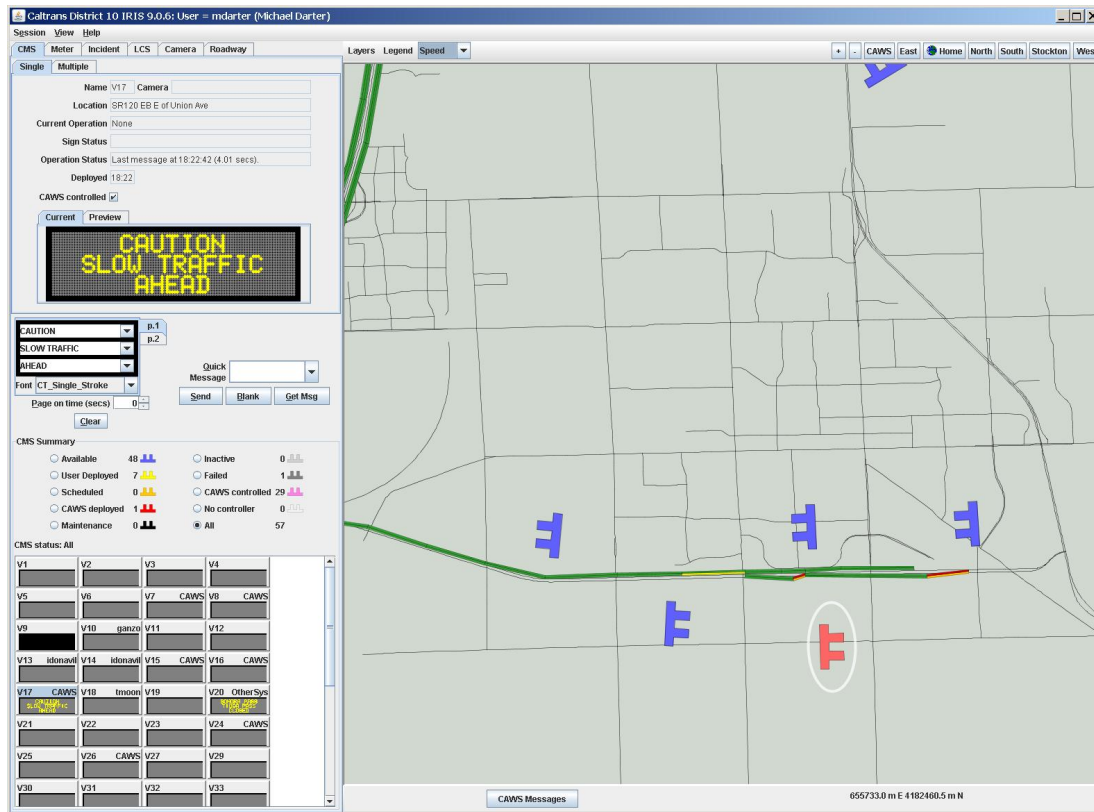


Figure 5.5: Verification of CAWS CMS Activated by Mapped Traffic Congestion

## CAWS Failure Notification and Logging Safety Enhancements

If IRIS is unable to send a CAWS message to a CMS, an email is generated and sent to a predefined list of recipients. TMC staff are therefore aware of transient failures due to communication, configuration, or other problems. IRIS also logs failures, successes, and expected CAWS messages. The SOCCS system does not perform these actions.

## CAWS Testing Safety Enhancements

Significant testing-related enhancements were made to improve CAWS safety. See Section 3.6.

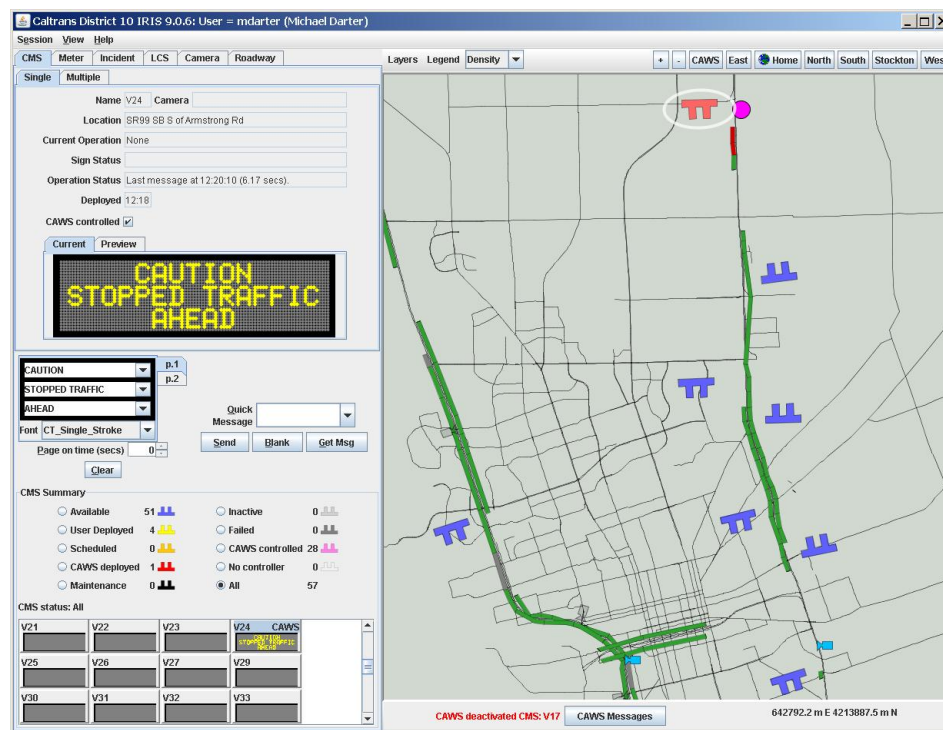


Figure 5.6: Verification of CAWS CMS Activated by Mapped Incident

### 5.1.3 Functional Integration Safety Enhancements

The map-based integration of CMS, real-time traffic, incident locations, video cameras, and RWIS stations (future), can provide operational benefits, enabling operators to see functional relationships that would otherwise be difficult to realize using separate applications (see Figure 5.7).

## 5.2 Functional Enhancements to TMC Operations

The IRIS implementation in D10 provided the TMC with the following operational enhancements:

### 5.2.1 General Enhancements to TMC Operations

- User authentication using an LDAP authentication application server. The authentication server may be shared across all IRIS installations within Caltrans, or installed per district. Alternatively, an existing LDAP server may be used.
- User permissions: IRIS supports read, write, view, and delete permissions that are specified for each functional area (CMS, cameras, system attributes, etc.). See Figure 4.4. D10 uses permissions to enable operator administrator editing of message

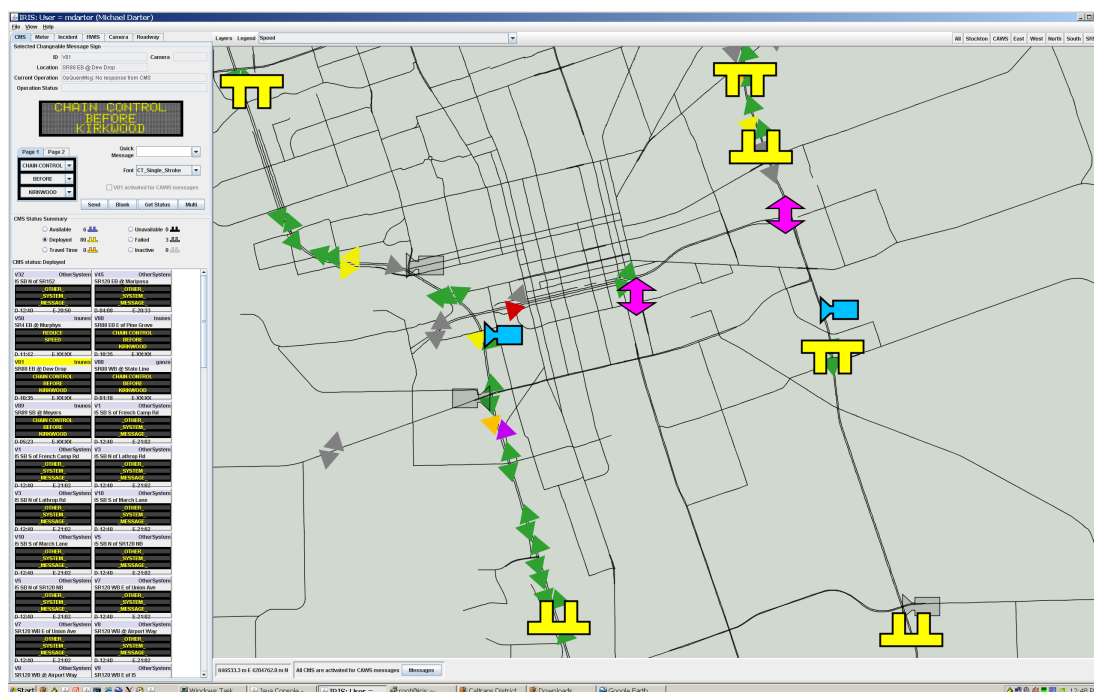


Figure 5.7: IRIS integration of incidents, CMS, camera positions, and traffic (see pg. 36)

libraries, but enable read-only access for operators. Also, permissions are used to assign read-only IRIS login accounts, which are used by non-operators to view the status of district traffic operations.

## 5.2.2 Enhancements to TMC Traffic Monitoring

Prior to IRIS, D10 used Freeway Performance Measurement System (PeMS) to monitor real-time traffic.

- Map integrated monitoring of real-time traffic speed, density, and flow. Other map elements include CMS, incidents, cameras, and roads. This enables operator monitoring of relationships between traffic and CAWS messages and incidents.

## 5.2.3 Enhancements to TMC CMS Monitoring and Control

Prior to IRIS, D10 used SOCCS and the legacy SignView application to monitor and control CMS.

- Generation of travel time messages supporting multiple destinations and message display formats.
- CMS integrated with mapping and other functional areas.

- Support for multiple editable fonts and characters.
- Ability to send a CMS message to predefined groups of CMS.
- Ability to save free-form CMS messages to a message library.
- A prioritized CMS message hierarchy, e.g., travel time messages do not supersede operator messages.
- Ability to perform CMS operations (blank, send, query message) on predefined and ad-hoc groups of signs.
- Identification of the CMS message author.

## 5.2.4 Enhancements to TMC Video Monitoring and Control

Prior to IRIS, D10 used Windows Media Player to monitor and control video cameras.

- Ability to view camera streams, integrated with mapping and other functional areas.
- PTZ control of cameras, including joystick control at each client.

## 5.2.5 Enhancements to TMC Reporting

Prior to IRIS, D10 used SOCCS for CMS reporting. However, D10 had no way of determining who made changes or when they were made. D10 also did not have the ability to view log files using Graphical User Interface (GUI) tools—accessing and viewing the log files was difficult, cumbersome, and unreliable. In addition, the message author was logged simply as 'TMC,' not the name of the operator who sent the message.

- CMS activity reports:
  - Sorted by time period (last hour, last day, last week, last month, etc.),
  - Sorted by message author (last hour, last day, last week, last month, etc.),
  - Sorted by sign number (last hour, last day, last week, last month, etc.),
  - Automatically generated report displaying calculated number of messages per month. Previously, this was a manual counting process.
- CMS network communication health reports. See Figures 5.2 and 5.3. Previously, D10 had no way to monitor communication network performance or outages.
- Ability to view IRIS application logs using a GUI tool. See Figure 5.8.
- Comprehensive logging: who activated and deactivated CMS from CAWS control, operator log-in history, etc.



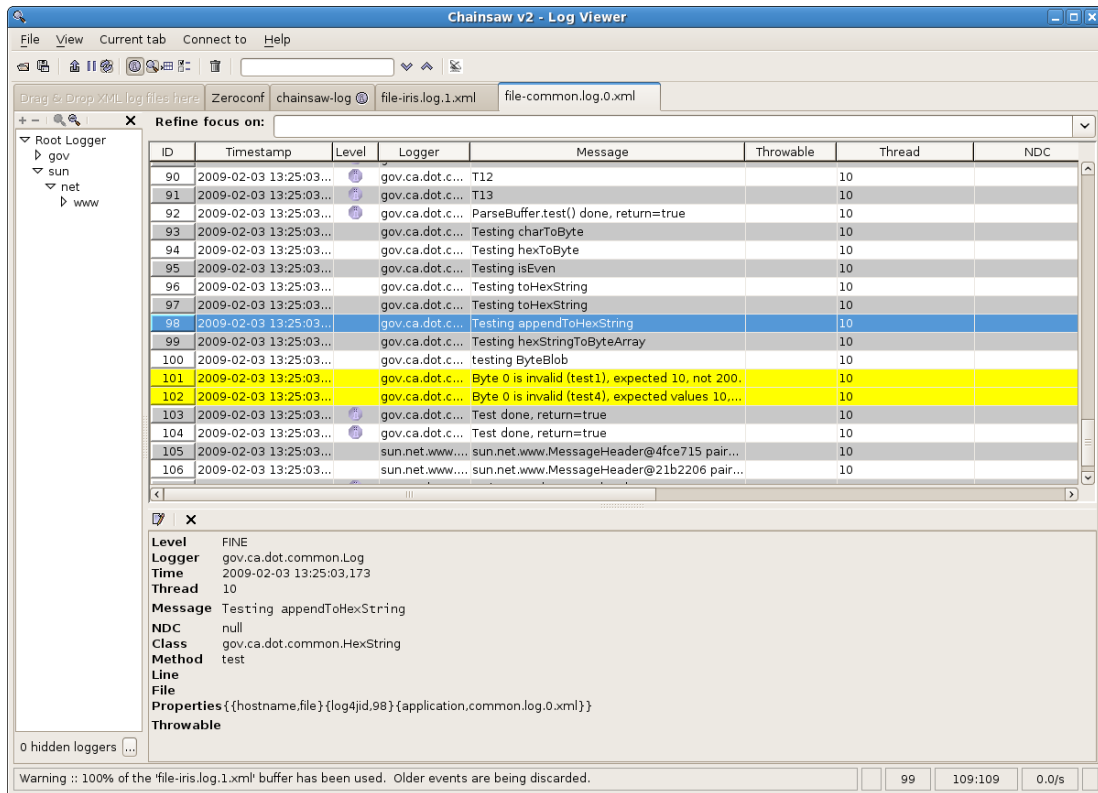


Figure 5.8: Screenshot of the Chainsaw Application Log Viewing Tool (see pg. 38)

### 5.3 IRIS Integration with Proprietary Protocols

The collaborative open-source approach requires integrated software code to be publicly available [21]. This may be problematic for proprietary software that can not be publicized for licensing, security, or competitive reasons. For example, the Caltrans San Diego Ramp Metering System (SDRMS) and CMS protocols both have issues which preclude releasing them as open-source.

A solution to this integration problem is the creation of a stand-alone server that interfaces with IRIS using an open-source protocol, and with field elements using the proprietary protocol. This architecture is shown in Figure 5.9. Integration with future proprietary protocols should use this same approach. The Sensor Server application is designed to support multiple protocols, with each protocol localized in its own name-space. The framework consists of 18 Java classes that would be reused by subsequent protocols. The new open-sourced DMSLite protocol was added to IRIS for communication between IRIS and the Sensor Server.

The effort to develop a stand-alone application and additional protocol is larger than the effort to integrate a single existing protocol with IRIS. The availability of the Sensor Server framework for integration of subsequent proprietary protocols is anticipated to reduce the effort. See Section 5.6 for a discussion of associated costs.

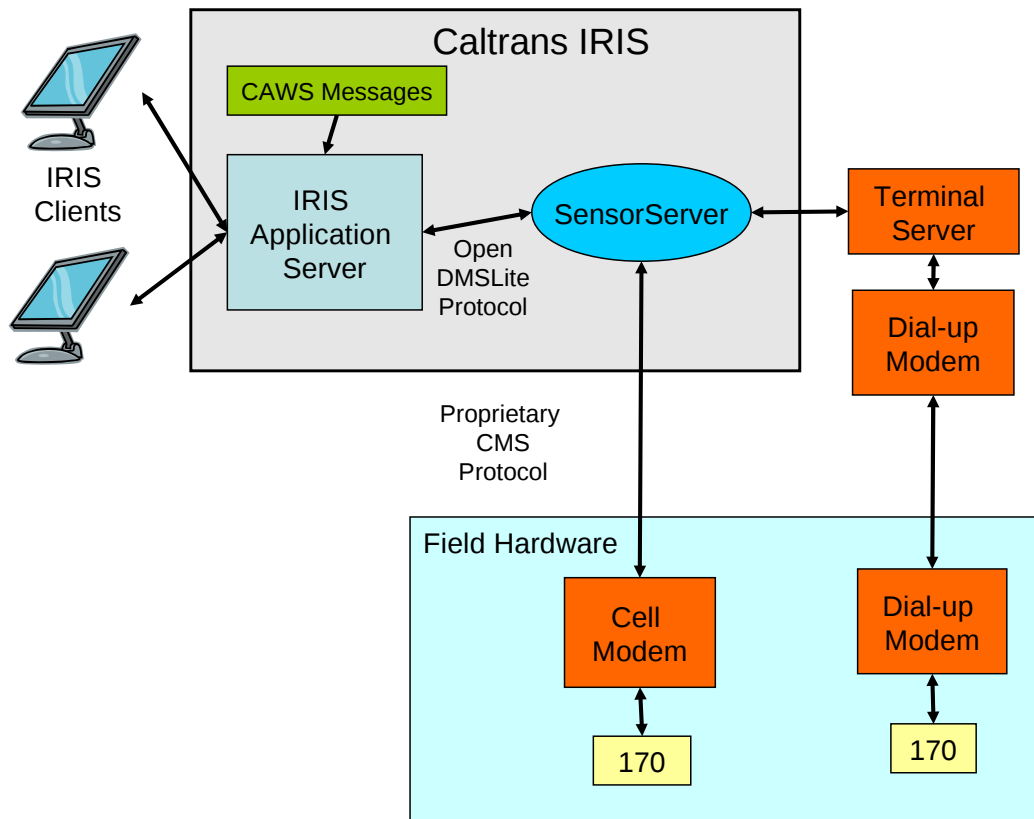


Figure 5.9: Sensor Server Architecture and Relationship with IRIS (see pg. 39)

## 5.4 Project Contributions to IRIS

The software code developed during the study was contributed back to the open-source IRIS project. These contributions benefit other agencies using IRIS, such as Mn/DOT, Wisconsin Department of Transportation (WisDOT), and Montana Department of Transportation (MDT). Typically, contributions are automatically merged into the existing IRIS code base by the source control system, provided they are 1) of high code quality and 2) elegantly extend the IRIS design (see Section 2.3.3). The value of study contributions to the IRIS project is discussed below 1) quantitatively, 2) qualitatively, and 3) from Mn/DOT's perspective.

### 5.4.1 Quantitative Contributions to IRIS

Table 5.1 shows development hours, Source Lines of Code (SLOC), and *cumulative SLOC*<sup>1</sup> for each developed application and a calculated SLOC estimate for AHMCT IRIS contributions, based on cumulative SLOC extracted from AHMCT source code repositories.

<sup>1</sup>Cumulative SLOC is the total number of lines of code including all prior modifications.



Figure 5.10 illustrates cumulative SLOC history for all of the IRIS repositories. SLOC was calculated as the number of new lines of code in each repository change set.

Figure 5.11 shows the cumulative repository contributions by Mn/DOT and AHMCT to the IRIS repositories.<sup>2</sup> This figure shows the increased contribution rate (line slope) to IRIS, as a result of collaboration. Additional agencies are anticipated to provide a higher rate of improvement.

Figure 5.12 shows cumulative SLOC for the applications developed by AHMCT. The large increase in cumulative SLOC for Sensor Server near the end of the project is largely due to code reorganization to increase modularity.

The authors hope that the values shown here for hours, SLOC, and cumulative SLOC can serve as a guide for agencies in the future that are estimating schedules and resources. For reference, AHMCT development staff had no experience with IRIS at the onset of the project.

Table 5.1: Quantified Project Results by Component (see Section 5.4.1)

<b>Component</b>	<b>Component Function</b>	<b>Development Hours</b>	<b>SLOC</b>	<b>Cumulative SLOC</b>
casper	Controller simulator	40	2,000	5,333
sensorserver	Sensor Server	2,200	10,300	50,815
common	Common library	190	5,600	13,423
trafserver	Traffic server	160	1,500	23,600
watchdog	AWS validation	40	318	486
iris	AHMCT IRIS contributions	2200	14,000 (est.)	54,200
Total AHMCT		4830	33,718	147,857

## 5.4.2 Qualitative Contributions to IRIS

Project contributions to IRIS fall into three categories: 1) stand-alone applications developed specifically for D10, 2) IRIS enhancements, and 3) new IRIS features.

- *Stand-alone Applications Developed by AHMCT:*

- CASPER: an application that simulates multiple field controllers. This was used for testing and verification. See Section 5.5.1.
- Sensor Server: an application that communicates with CMS field elements using a proprietary protocol. See Section 5.5.3.
- Traffic Server: an application that interfaces with existing D10 systems and provides real-time traffic data to IRIS. See Section 5.5.5.
- Watchdog: an application that performs AWS verification. See Section 5.5.6.

<sup>2</sup>These statistics were generated from the D10 8.9 release.

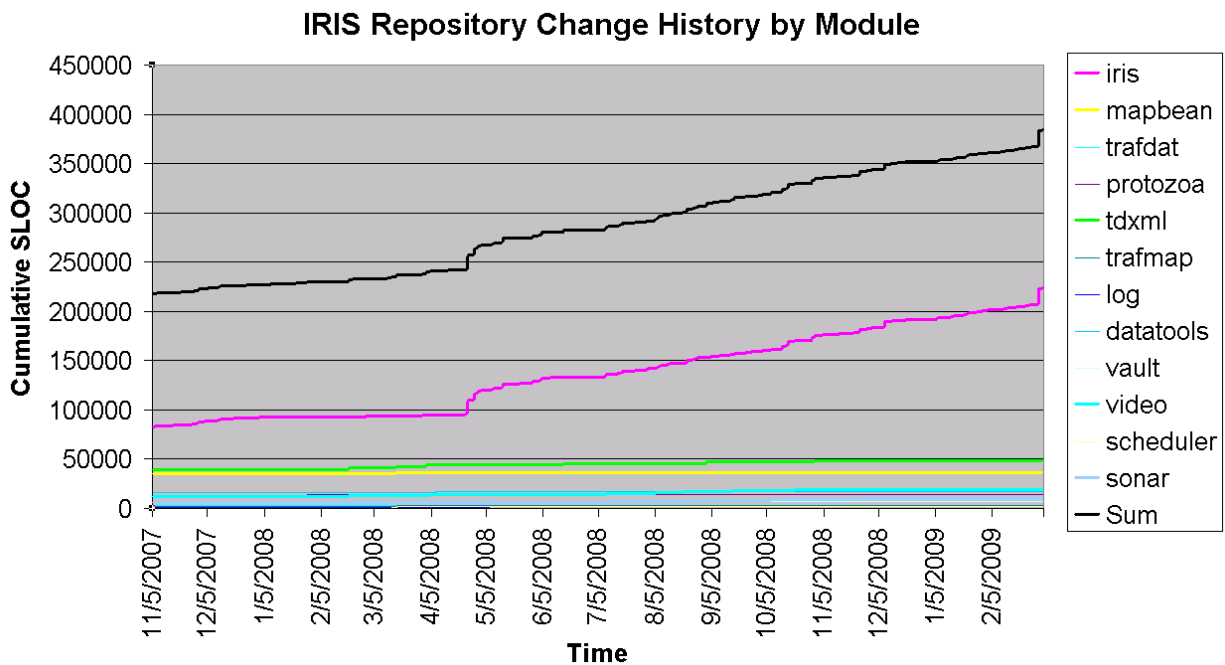


Figure 5.10: Cumulative SLOC for all IRIS Modules  
(see pg. 40)

- *IRIS Enhancements Contributed by AHMCT:*

- Build process: the IRIS build process was enhanced with build-all and clean-all functionality and module version verification.
- RPM Package Manager (RPM) functionality: the build process optionally creates RPM files which greatly simplify installation of IRIS and all stand-alone applications. IRIS applications can be installed or uninstalled with a single operating system command.
- Free-form CMS message entry: this feature was requested by D10, and the initial implementation was performed by Mn/DOT. There were subsequent enhancements by AHMCT and Mn/DOT. This feature is used in both TMCs.
- Incident generalization: incident functionality was generalized, making it easier for other agencies to extend incident functionality.
- Video interface enhancements: PTZ controls and preset buttons were added to the video interface.
- Defect discovery and repair: AHMCT detected and repaired a number of defects, which resulted in increased reliability. See Section 5.5.4.

- *New IRIS Features Contributed by AHMCT:*

- Internationalization: provides the ability for agencies to override default naming for on-screen menu items, button text, and labels. This is a significant contribution, as it allows agencies to configure and customize aspects of IRIS without modifying the code-base. See Section 4.4.2 and Figure 4.3.

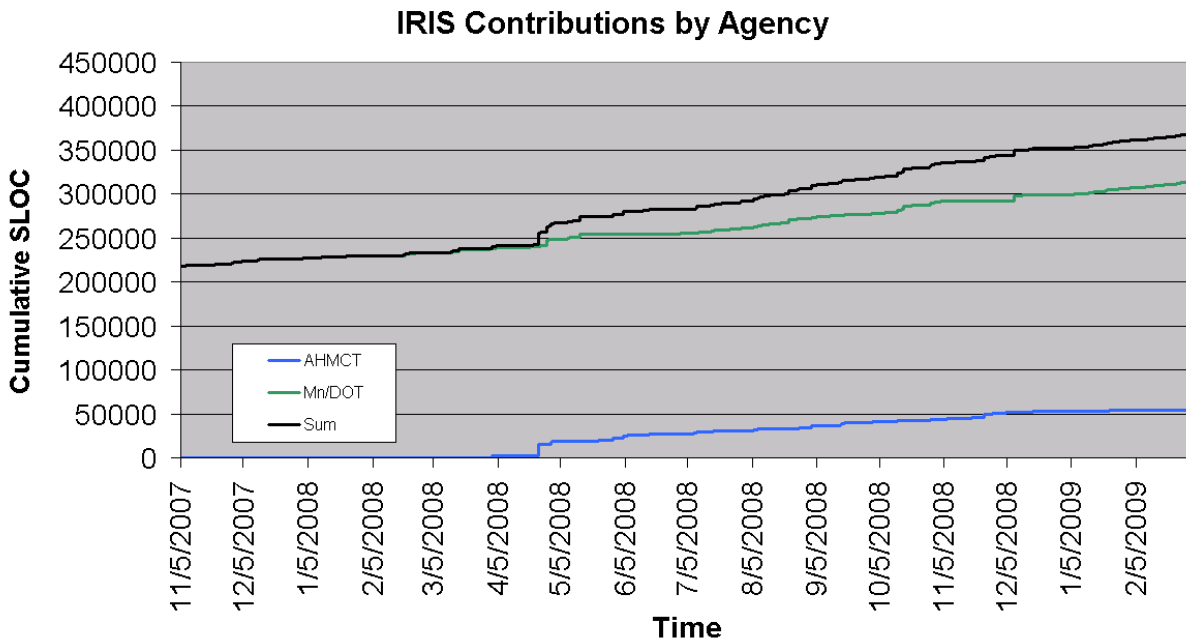


Figure 5.11: Collaborative Development: Cumulative IRIS Contributions by Agency

- System Attributes: system-wide settings are stored in a database and editable by administrators. See Section 4.4.1 and Figure 4.2.
- Quick Message Library: the ability to specify a predefined CMS message by ID was added. This capability allows operators to quickly specify complete predefined messages with associated fonts, and send the message.
- AWS support: AWS functionality was added to IRIS to provide D10 with integrated support of their existing CAWS system. This functionality was subsequently enhanced by Mn/DOT. Presently only D10 is using this functionality. With additional development (e.g. integration with RWIS), other agencies may find it valuable.
- Automated unit tests: were added to the build process. This enables software engineers to easily execute numerous unit tests during development. See Section 3.2.
- Google Earth support: the IRIS server periodically writes a static KML file for use by Google Earth clients. This enables the display of real-time messages on CMS embedded in the Google Earth map. KML output for additional functional areas is possible.
- Help System: context-sensitive help system functionality was added to IRIS. When the user presses the F1 key, a web browser is launched that displays a web page specific to the user's current form. The help system uses HyperText Markup Language (HTML) URLs that are specified in the agency-specific Internationalization (I18N) message files. This enables agencies to develop their own help pages or leverage existing pages from other agencies.
- Up-time Logging: the IRIS server logs critical system information every 60 sec-

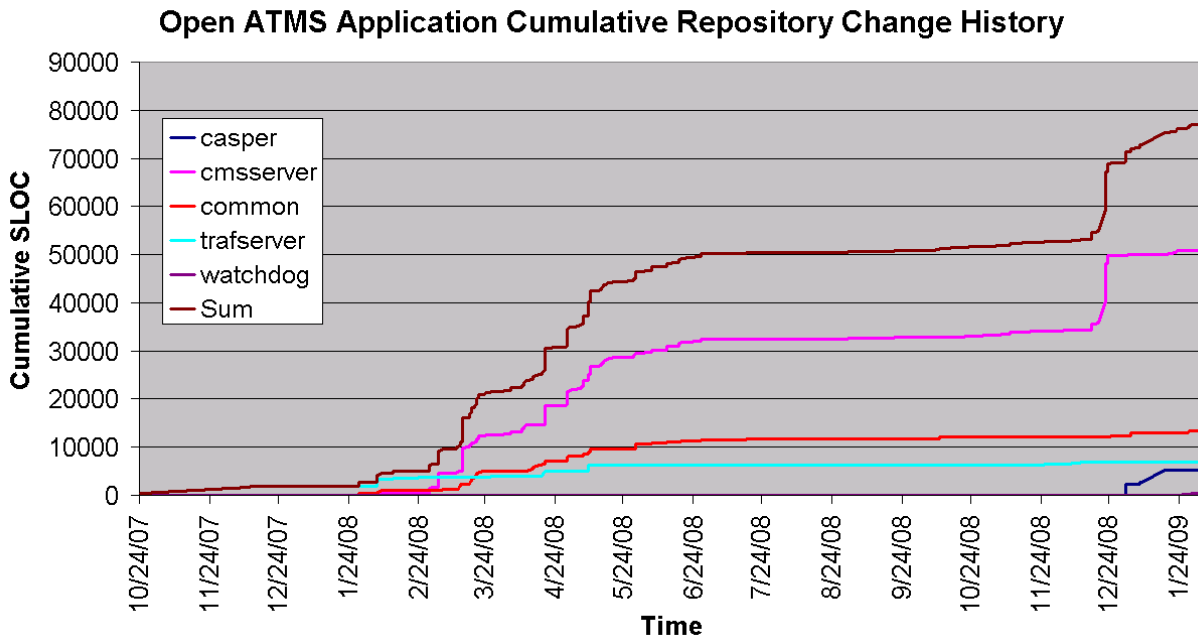


Figure 5.12: Cumulative SLOC for AHMCT-Developed Open ATMS Applications (see pg. 40)

onds. This log is used to monitor processor usage, memory usage, the number of users, etc. It is also used to evaluate the effects of system changes across new versions of IRIS, the database, JVM, etc.

### 5.4.3 Mn/DOT's Perspective on Study Contributions

From Mn/DOT's perspective, what were the advantages and disadvantages of the study? Mn/DOT reported that...

This collaboration was the first effort to deploy IRIS in a non-Mn/DOT setting, following the decision to make IRIS available on an open-source [General Public License](#) (GPL) basis. Mn/DOT needed to make significant modifications to the IRIS code, in close cooperation with the AHMCT staff, to make IRIS more suited to adaptation to different traffic management systems, field devices and geographic locations.

The IRIS code modification effort required a fairly significant amount of Mn/DOT staff time where these resources are very much in demand for ongoing development and support of our Freeway Management Operations. The result was a temporary disadvantage of having to postpone desired enhancements of IRIS. However, once completed, the effort directly resulted in immediate benefits to the IRIS code by making it easier to maintain, easier to develop enhancements, easier to install in new locations, and generally more

powerful and stable.

Therefore, rather than being a disadvantage, the IRIS modification efforts were essentially an investment in IRIS that will reap future benefits from the collaboration of multiple entities in further development of the IRIS software. It is now easier to add features and fix problems. Also, having other people and agencies familiar with the code base gives us additional confidence that IRIS can be maintained and continually improved into the future. This is essentially the goal of all open-source software efforts.<sup>3</sup>

## 5.5 Other Study Products

Study products not covered elsewhere are discussed here. This includes software applications that are not a part of the Mn/DOT IRIS distribution but are a part of the Caltrans IRIS distribution, a ticket system shared by developers, and others.

### 5.5.1 CASPER Field Controller Simulator

The Controller Array Simulator for Performance and Enhanced Reliability (CASPER) field controller simulator was developed for simulating multiple field controllers using multiple protocols. This enabled end-to-end system testing of IRIS CAWS functionality. It also enabled testing of CMS functionality that otherwise would be untestable, such as AMBER alerts.

CASPER is used for simulating CMS on both dial-up modems and cell modems. It is also used to simulate various error conditions (e.g. returning incorrect messages, not responding, etc.). The ability to reliably and repeatedly test using these conditions is invaluable for improving the reliability of developed code.

CASPER supports multiple protocols. Code for each protocol is placed in a dedicated name-space. Each simulated field controller is associated with a single Transmission Control Protocol / Internet Protocol (TCP/IP) port. CASPER supports any number of simultaneous field controllers, each on a different port. The architecture supports the development of different types of field controllers (not just 170s).

### 5.5.2 IRIS Developer Ticket System

The researchers installed and currently maintain a ticket tracking system for use by IRIS developers across all agencies.<sup>4</sup> The goals of this system are to facilitate information flow among IRIS developers and provide:

---

<sup>3</sup>Reported by Mn/DOT in January of 2010.

<sup>4</sup>For the IRIS developer ticket system see <http://iris.ahmct.ucdavis.edu>

- Defect tracking and history.
- Enhance request tracking, history, discussion, plans.
- Link repository change sets with detailed ticket histories.
- Provide a resource for others to evaluate IRIS reliability, progress, development pace, etc.

The contents of the IRIS software engineer ticket tracking system are covered by the GNU Free Documentation License (GFDL), and the system is presently hosted by AHMCT. It is anticipated that the ticket system will move to Mn/DOT's IRIS server.

The Trac ticket system is being used. There is the possibility of future interoperability with the Caltrans Trac system and other agencies that have their own ticket tracking system. During the course of the project, the Caltrans IRIS Trac system was used to track hours, requirements, tasks, and defect repairs that were specific to the Caltrans implementation of IRIS. Caltrans' pioneering use of a ticketing system was extremely beneficial to the project and served as an example of effectiveness for the IRIS developer ticket system. Agencies adopting and contributing to IRIS should adopt an internal ticket tracking system for agency-specific development and issues.

### 5.5.3 Sensor Server

The Sensor Server is a stand-alone application developed to interface IRIS with CMS field controllers. It uses a generalized architecture with the goal of supporting additional proprietary protocols. Sensor Server was developed as a stand-alone application for licensing reasons (see Section 5.3). The open-source DMSLite protocol is used between IRIS and the Sensor Server, and a proprietary CMS protocol is used to directly communicate with field controllers. See Section 5.3 for a discussion of proprietary protocols. See Figure 5.9 for the CMS architecture.

#### Sensor Server Reusability

How reusable is Sensor Server code for another proprietary protocol, such as ramp metering? Approximately 50% of Sensor Server code is protocol-specific, and 50% is general framework code. A rough estimate for developing a new protocol using the Sensor Server framework would then be about 50% of the development effort of the original code. In terms of additional effort, some retrofitting of the original framework code would inevitably be necessary.

### 5.5.4 Defect Discovery and Repair

The collaborative approach had a positive effect on defect discovery and repair. In general, it was found that developing and executing IRIS in multiple environments by multi-

ple developers enhanced reliability by exposing defects and nuisance problems that otherwise would have been difficult to detect. Reliability was also enhanced by multiple developers viewing and discussing the same code. Examples of types of defects that were encountered are discussed below. The intent is to document different classes of typical problems.

- Database compatibility issues: initially, AHMCT was using two Linux distributions<sup>5</sup> for development (Ubuntu and Fedora). Minor database compatibility problems in Structured Query Language (SQL) statements were uncovered when running IRIS on Ubuntu. Generalized SQL statements were created that worked with IRIS running on both platforms, which enhanced reliability.
- External file dependency: the IRIS client required the existence of an optional file to successfully start. This problem can be classified as a configuration nuisance problem. AHMCT produced a patch that enabled the client to start without availability of the external file. This is also an example of *failing badly*<sup>6</sup>.
- Missing property file values: this nuisance problem can also be classified as failing-badly. Optional values within property files were incorrectly required to have values, or the server would not start.
- Null Pointer Exceptions (NPEs): a number of NPE errors were fixed. These were due to differences in configuration files and maps and were therefore not visible to Mn/DOT. The repaired defects increased reliability.
- RMI-related problems: these nuisance problems were visible to both Mn/DOT and AHMCT. Mn/DOT's long-term development schedule included elimination of these defects by replacing all client-side RMI code with SONAR code. This was completed in June of 2009 with release 9.0. Due to the nature of the repair (the SONAR conversion), AHMCT was dependent on Mn/DOT to resolve the problem. This is an example of a dependency that complicates collaborative development.
- Video reliability problems: the reliability of video streams on slower networks was impacted due to assumptions in the video code about network delays. Resolution involved adjusting a timeout value and verifying performance with lab testing of various simulated network delays.

### 5.5.5 Traffic Server

The traffic server is a stand-alone application that receives real-time traffic data from the existing D10 infrastructure (see Figure 5.13). The data is re-formatted, queued, and sent to the connected IRIS server using the Wavetronix SmartSensor protocol. The IRIS server reads the traffic data using its open-source Wavetronix driver. IRIS then displays the real-time traffic data on a map. The Traffic Server can also be used to feed simulated traffic to IRIS.

<sup>5</sup>For *Linux distribution* see [http://en.wikipedia.org/wiki/Linux\\_distribution](http://en.wikipedia.org/wiki/Linux_distribution)

<sup>6</sup>For *failing badly* and *failing well* see [http://en.wikipedia.org/wiki/Failing\\_well](http://en.wikipedia.org/wiki/Failing_well)

## 5.5.6 Watchdog

Watchdog is a stand-alone application that was developed to perform CAWS testing and verification. It compares new messages generated by the CAWS system with CAWS messages actually sent to the CMS. Discrepancies are flagged as potential defects. Several subtle and difficult defects were detected using this application that otherwise would have been extremely difficult to detect.

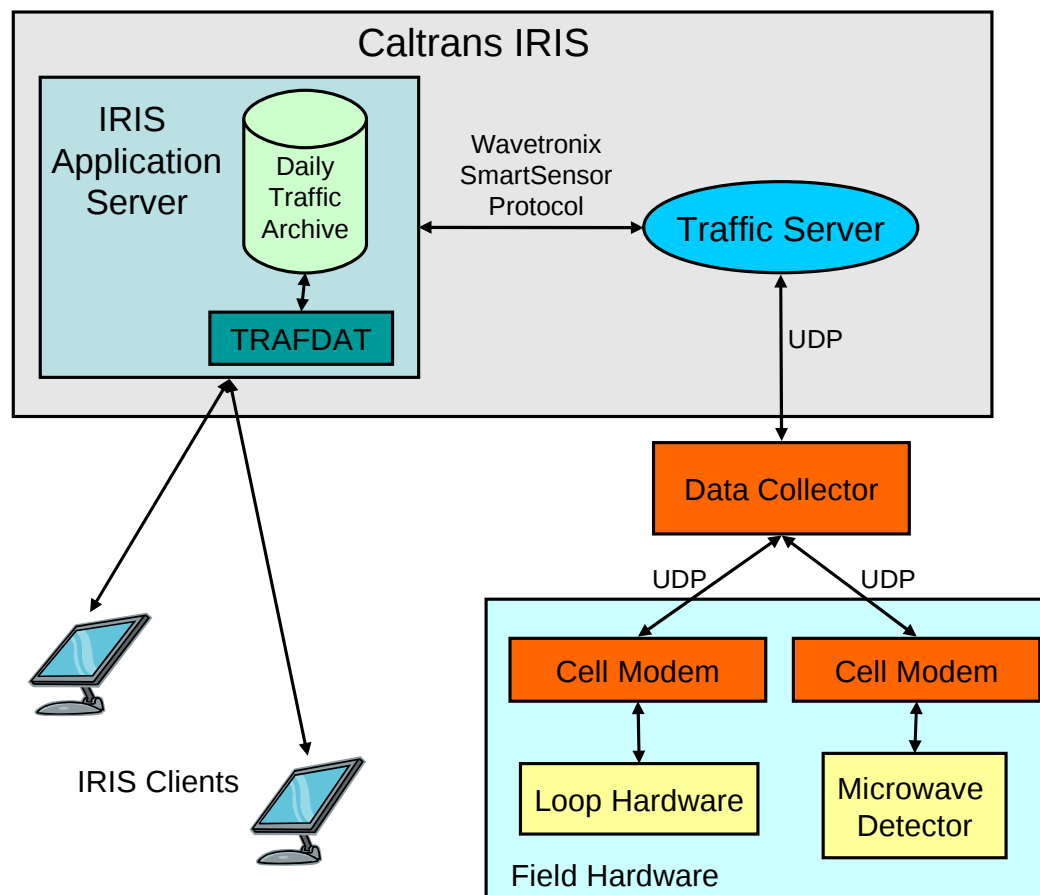


Figure 5.13: Traffic Server Relationship with IRIS (see pg. 47)

## 5.6 Costs

This section covers effort and cost issues from different perspectives. The total dollar value of IRIS and developed project components are estimated based on the number of lines of software code. A component cost breakdown is provided to facilitate estimates for future projects of similar magnitude. IRIS maintenance costs are estimated, and finally, a high-level cost comparison between the open collaborative approach and proprietary



approach is made.

### 5.6.1 Estimating the Free Economic Value of IRIS

IRIS can be downloaded for free by anyone. What is the economic value in dollars of a free download? In other words, if IRIS was developed from scratch, what would be the total incurred development cost? This question can be answered using the COConstructive COSt Model (COCOMO) model on existing software projects. The SLOCCount tool<sup>7</sup> uses the COCOMO model to estimate hours and development costs for existing software, using developed Source Lines of Code (SLOC) as an input. The calculated estimates for casper, sensorserver, common, trafserver, and IRIS are shown in Table 5.2. The dollar values are based on a U.S. Bureau of Labor Statistics average salary of \$75,662 for a U.S. software developer in July 2008, with an assumed overhead rate of 2.4. The values in Table 5.2 represent estimates of the effort and cost required to develop these applications, using industry averages. *The values in Table 5.2 do not represent actual hours or costs to Mn/DOT or Caltrans.*

A key conclusion from the results in Table 5.2 is that the collaborative open-source model of ITS software development can result in agencies receiving much more software economic value than they spend. Caltrans gained a total of \$4,239,300 in software from the Open ATMS project and spent approximately \$400,000 to get it. In other words, *Caltrans gained approximately ten times what they spent in terms of software dollars.* Using the traditional proprietary model, agencies must spend \$1 to gain \$1 of software.

Table 5.2: Estimating the Free Economic Value of IRIS

Application	Application Function	SLOCCount SLOC	SLOCCount Estimated Hours (not actual hours)	SLOCCount Estimated Cost (not \$ spent)
casper	Controller simulator	1,296	539	\$47,682
sensorserver	Sensor server	5,801	2,651	\$230,037
common	Common library	3,377	1,486	\$130,340
trafserver	Traffic server	936	384	\$33,881
iris	IRIS client and server	83,792	43,493	\$3,797,360
<b>Total</b>		95,202	48,553	\$4,239,300

### 5.6.2 Component and Functional Area Effort Breakdown

Table 5.3 shows the effort breakdown by functional area and application.

<sup>7</sup>For SLOCCount information and assumptions see <http://www.dwheeler.com/sloccount>

### 5.6.3 Estimated IRIS Maintenance Costs

At a minimum, maintenance for transportation districts or agencies should include the items below:

1. Support:

- (a) Problem resolution,
- (b) Answering questions from district administrators and maintenance personnel,
- (c) Updating the IRIS Maintenance Manual in subsequent releases,
- (d) Updating the ticket tracking system and preparing any reports,
- (e) Training for new features in subsequent releases.

2. Development:

- (a) Synchronization with new Mn/DOT IRIS releases,
- (b) Defect diagnosis and repair,
- (c) Consultation with Mn/DOT and other developers as needed,
- (d) Development of minor enhancements,
- (e) Updating the ticket tracking system and new feature documentation,
- (f) Collaboration with Mn/DOT, forwarding code changes.
- (g) Building, testing, installing new releases.

The support effort described here is expected to scale to some degree—that is, the required effort (per district) to support multiple districts would be less than the required effort to support one.

Table 5.3: Component and Functional Area Effort Breakdown (see pg. 49)

Component	Description	Hours	Percent
casper	Field controller simulator	64	1
common	Common module	300	4
sensorserver	Sensor Server application (proprietary protocols only)	1,107	16
trafserver	Traffic server application	160	2
watchdog	Verification application	40	1
IRIS	Traffic extensions	80	1
IRIS	CMS extensions	1,369	20
IRIS	CAWS extensions	673	10
IRIS	Video extensions	225	3
IRIS	Incident extensions	160	2
IRIS	Logging	80	1
IRIS	Mapping	440	6
IRIS	General	382	6
<b>Development Total</b>		<b>5,080</b>	<b>73</b>
<b>Everything else</b>	Reports, user manuals, presentations, training, support	<b>1,838</b>	<b>27</b>
<b>Total hours</b>		<b>6,918</b>	<b>100</b>

Item 2d, "Development of minor enhancements" is important to elaborate. Because IRIS is an open-source project, contributing some amount of features and defect repairs back into the IRIS source code is important. An open-source product like IRIS does not remain static. The rate of improvement is high (see Figure 5.10). Ongoing maintenance would require some level of consultation with Mn/DOT for problem resolution and discussion of issues with new releases. Professional etiquette requires that if any agency is using Mn/DOT's scarce developer time, something should be contributed back to the IRIS code-base. This is particularly important considering that any transportation agency receives updates and enhancements from Mn/DOT *for free*. Contributing back to the IRIS code base will help assure Mn/DOT's continued active and valuable development of IRIS.

The estimated effort to provide maintenance to a Caltrans district the size of D10 is expected to range from one-half to one Full Time Equivalent (FTE), depending on the desired level of contributions back to the IRIS project.

#### 5.6.4 High-level Cost Comparison of Open and Proprietary ATMS

This section provides a cost comparison between two ATMS development approaches, based on actual costs incurred by Caltrans for the IRIS system and the existing proprietary ATMS used in 7 of 12 districts. Each of the following life-cycle cost components is discussed below:

- Acquisition costs (Table 5.4),
- Installation, configuration, and customization costs (Tables 5.5 and 5.6),
- Annual maintenance costs (Table 5.7), and
- Total five-year cost estimate which incorporates the above three costs (Table 5.8).

The goal of this cost comparison is to outline differences between the open-source collaborative approach and the proprietary approach presently used by Caltrans and other DOTs. No attempt is made to compare the feature sets of these systems, which are somewhat different and complimentary. For example, the existing Caltrans ATMS system supports adaptive ramp metering, incident detection, and incident response, while IRIS supports travel time generation, Variable Speed Limits (VSLs), lane control signs, and visualization of historic traffic data (TRAFDAT).

##### Acquisition Costs

Acquisition costs show a 98% cost reduction for the open ATMS approach. Acquisition costs are shown in Table 5.4. Table notes are listed below.

1. Server hardware acquisition cost: for IRIS, a single commodity x86 server with a redundant drive array. The 2007 cost was \$4,500 for an 8-core Hewlett-Packard (HP)

system with 4 GBytes of memory and a 1 TB drive array. This included 5 years of 24x7 on-site hardware support. Note that in terms of hardware, this system is over-specified. IRIS will run very well on a low-priced desktop commodity x86 machine (see Section 3.7). For the ATMS system, the numbers for acquisition costs in Table 5.4 are based on actual Caltrans costs [2].

2. Backup server hardware acquisition cost: same as above.
3. Client acquisition costs are for 5 workstations. The use of existing commodity x86 workstations, quite feasible at least for IRIS, would eliminate this cost item.
4. Server software license cost: this represents the cost for all the software licenses required for the application and database server. With their existing proprietary ATMS system, Caltrans typically prefers to use separate machines for the application and database server, which may increase hardware and software costs further.
5. Software system support: this is an optional cost item and was not purchased for D10. It represents the cost for 24x7 operating system support provided by either a consultant or vendor.
6. Backup server software license cost: this represents the cost for all the software licenses required for a backup application and database server.
7. Single software development license cost: this is the cost of software licenses to support a single development environment. For multiple districts, this cost item may be shared. For example, Caltrans shares 3 development environments across 7 districts. For the Open ATMS, the acquisition costs for development and run-time software for IRIS are \$0. This includes the database (PostgreSQL), server operating system (Linux), run-time environment (Java, Apache), and development tools (OpenJDK, Ant, Mercurial). For the proprietary ATMS system, these costs were estimated based on discussions with vendors and Caltrans estimates [2].

Table 5.4: Open and Proprietary ATMS Acquisition Costs For One Agency

Note	Acquisition Cost Components	Open ATMS	Proprietary ATMS
1	Server hardware acquisition	\$4,500	\$150,000
2	Backup server hardware acquisition	\$4,500	\$150,000
3	Client workstations (5) acquisition	5x \$1,500	5x \$1,500
4	Server software license cost	\$0	\$239,000
5	Software system support, unlimited, 24x7	\$1300	included
6	Backup server software license cost	\$0	\$239,000
7	Single software development license cost	\$0	\$269,000
	<b>Total Acquisition Cost</b>	<b>\$17,800</b>	<b>\$1,054,500</b>
	Cost Reduction (%)	98%	

## Installation, Configuration, and Customization Costs

Estimated installation, configuration, and customization costs for two Caltrans districts are shown in Table 5.5. This table is *not* meant to provide a comparison of these costs between the open and proprietary approach. This cost category is not expected to differ substantially between the two approaches. The researchers stipulate that approximately the same effort would be required to add a new feature to both systems. There may be a slight cost advantage for the open-source approach due to 1) the high code quality and design encouraged by the open-source approach, and 2) low barriers to entry, encouraging competitive bidding from many firms and the elimination of sole-source contracts. Table notes are listed below.

Table 5.5: Installation, Configuration, and Customization Costs for Open and Proprietary ATMS (see pg. 53)

	Installation, Configuration, Customization	Open ATMS	Proprietary ATMS
1	D4 installation, configuration, customization	Not applicable	\$750,000
2	D10 installation, configuration, customization	\$350,000	Not applicable

1. District 4 (D4) proprietary ATMS installation, configuration, and customization: for installation of the existing Caltrans ATMS system within District 4, which is a large urban district. The cost represents significant enhancements made to the ATMS system. The existing ATMS system contains functionality not available in IRIS, such as adaptive ramp metering, incident detection, and incident response.
2. D10 IRIS installation, configuration, and customization: cost is the effort required to install and configure the existing IRIS applications in Caltrans District 10. The numerous enhancements completed are the subject of this report.

## Costs for Installation and Configuration Only

Table 5.6: Costs: Installation and Configuration Only, i.e. When No Customization Needed for Open ATMS (see pg. 53)

	Installation, Configuration	Open ATMS
1	D1 installation and configuration	\$6,000
2	D5 installation and configuration	\$6,000

For future planning purposes, Table 5.6 provides the approximate cost for installation and configuration of IRIS in two Caltrans districts. These rural districts were configured for the CMS portion of IRIS; they were not configured for and did not use the other components of IRIS. Table 5.6 serves as a cost guide for entities that can use existing drivers in IRIS, i.e. districts or agencies that would need no customization. Of course, configuration costs will vary based on a number of factors, including: urban vs. rural, number

and diversity of field elements, and existing state of configuration information. However, the experience in the current research project for configuring the two Caltrans districts provides a solid basis to assume that the general configuration costs can be expected in the same order of magnitude as those indicated in Table 5.6. Table notes are listed below.

1. D1 IRIS installation and configuration: cost is the effort required to install and configure the existing IRIS applications for CMS use in Caltrans District 1, a small rural district. The ongoing IRIS installation in D1 is a trial and for evaluation purposes. Activated functionality included CMS (approximately 20), mapping, and defined roads. This was a configuration effort and required no modification or customization to IRIS, other than the repair of defects discovered related to dial-up modem use.
2. D5 IRIS installation and configuration: cost is the effort required to install and configure the existing IRIS applications for CMS use in Caltrans District 5, a small rural district. The ongoing IRIS installation in D5 is a trial and for evaluation purposes. Activated functionality included CMS, mapping, and defined roads. This was a configuration effort and required no modification or customization to IRIS.

### Annual Maintenance Costs

Annual maintenance costs are shown in Table 5.7. Annual costs are projected to be reduced 68% - 86%. Table notes are listed below.

Table 5.7: Annual Maintenance Costs For One Agency for Open and Proprietary ATMS (see pg. 54)

	<b>Annual Maintenance Cost Components</b>	<b>Open ATMS</b>	<b>Proprietary ATMS</b>
1	Server hardware maintenance	\$200	\$30,000
2	Software license cost	\$0	\$60,000
3	Maintenance cost, personnel	.5 - 1 FTE	2.5 - 3 FTE
	<b>Total Annual Maintenance Cost</b>	<b>\$75,200 - \$150,200</b>	<b>\$465,000 - \$540,000</b>
	Cost Reduction (%)	68% - 86%	

1. Server hardware maintenance: for the commodity x86 8 core HP system running IRIS was \$500 for 5 years, and was included with the initial system purchase. This included 24x7 support and hardware replacement. For the proprietary ATMS, these costs were estimated based on Caltrans estimates [2].
2. Annual software license costs: for the IRIS system is \$0. This includes the database (PostgreSQL), server operating system (Linux), run-time environment (Java, Apache, LDAP server), and development tools (OpenJDK, Ant, Mercurial). For the proprietary ATMS, this includes the Oracle database server, Tibco SmartSockets, Sun's Java, GenSym G2, and R/T.

3. Maintenance cost for personnel: for purposes here, one FTE is assumed to be \$150k per year, which includes overhead.<sup>8</sup> The annual personnel maintenance costs are the most difficult to estimate, particularly because IRIS and the proprietary ATMS system have different feature sets. The intent here is to give an order-of-magnitude estimate for maintenance costs, and discuss comparison points. A key consideration is that IRIS software maintenance costs are shared with Mn/DOT and other agencies. Maintenance costs include the repair and detection of defects, which are shared by multiple contributing agencies. This network effect is anticipated to reduce maintenance costs, particularly as more agencies actively contribute to IRIS. For comparison, Mn/DOT has 2 FTE maintaining 1 production IRIS server, which is terms of size, is equivalent to the largest Caltrans district. Very little time is spent by these engineers performing system administration. The bulk of their time is spent 1) adding new features, and 2) fixing defects. The free availability of the source code eliminates vendor *lock-in*<sup>9</sup> and is anticipated to keep maintenance costs competitive. See Section 5.6.3 for more information.

### Five Year Costs

A 5-year cost comparison is shown in Table 5.8, with a 72% cost reduction for the open-source approach. This conservative comparison assumes the same customization costs for both approaches, and a high estimate for the annual personnel cost.

Table 5.8: Five-Year Acquisition, Configuration, & Maintenance Costs of Open and Proprietary ATMS For One Agency (see pg. 55)

Cost	Open ATMS	Proprietary ATMS
Acquisition	\$17,800	\$1,054,500
Configuration and customization	\$350,000	\$350,000
Year 1 maintenance	\$150,200	\$540,000
Year 2 maintenance	\$150,200	\$540,000
Year 3 maintenance	\$150,200	\$540,000
Year 4 maintenance	\$150,200	\$540,000
Year 5 maintenance	\$150,200	\$540,000
<b>Total 5 Year Cost</b>	<b>\$1,118,800</b>	<b>\$4,104,500</b>
Cost Reduction (%)	72%	

### Effort For Additional Caltrans IRIS Deployments

Late in the project, Caltrans requested that IRIS be deployed for testing in additional districts. IRIS was activated in D5 on August 12<sup>th</sup>, 2009 and in D1 on August 26<sup>th</sup>, 2009.

<sup>8</sup>The U.S. Bureau of Labor Statistics average salary for a U.S. software developer in July of 2008 was \$75,662. An overhead rate of 100%, would result in approximately \$150k.

<sup>9</sup>Lock-in can subsequently require non-competitive fees for continued use of the system.

The same IRIS code-base is used for all Caltrans districts (and Mn/DOT). No customized code was developed for these two additional IRIS deployments. Installation, configuration, and training time was less than two weeks per district. These deployments presently use CMS functionality, with no video or traffic monitoring capabilities. Illustrating the flexibility of IRIS, the D5 IRIS server is remotely located in the District 3 (D3) Rancho Cordova TMC. Deployments in additional districts, using similar capabilities are expected to require less than two weeks of effort per installation.

## 5.7 Effort Estimates for New IRIS Implementations

For new IRIS implementations, the following rough effort estimates may be useful. The effort estimate assumes engineer familiarity with the code-base, database, development tools, and operating system:

- Effort for a new IRIS installation: 1 to 3 weeks of configuration time. This includes server installation, configuration, map building, field element configuration, setting up user permissions, etc. This does not include any development effort. The installation and configuration of IRIS within Caltrans D1 and D5 required approximately 2 weeks per district.
- Effort for developing new hardware device drivers: 1 to 3 weeks of development and testing time per protocol. This assumes that the new protocol is well-defined and no UI enhancements are required.

Crucial questions for new implementations:

- How many new hardware devices need to be integrated with IRIS?
- How many existing software systems need to be integrated with IRIS?
- How many user interface enhancements need to be made? Are they extensive?

Integrating with hardware or software systems that use a proprietary protocol needs special consideration. Approximately 50% of the code in the developed Sensor Server application is reusable for the implementation of other proprietary protocols, e.g. proprietary ramp metering such as SDRMS. See Section 5.3 for more information.

## 5.8 Future Enhancements

Through the course of the study, desirable requirements were identified. Of 216 total requirements, 43 were identified as future requirements to be completed in a subsequent phase. See Appendix A for a complete list of requirements. A few of these are discussed below:



- *Highway Advisory Radio (HAR)*: integration of HAR functionality with IRIS is a high priority. Requirements 160 and 96-105 are specific to HAR. HAR integration is anticipated to provide a dedicated tab, showing the locations of stations on the map. The IRIS UI would provide the ability to activate stations and view their status. The hours estimate is presently 2-6 weeks of effort.
- *Remote Weather Information System / Roadway Weather Information System (RWIS)*: integration of RWIS functionality with IRIS is anticipated to provide a dedicated RWIS tab, with stations located on the map. Real-time station data would be provided by the UI. This data would be used by operators to confirm the accuracy of CAWS messages. For example, when a "high wind advisory" message is displayed, operators could view real-time RWIS wind speeds. See requirements 75-82 and 147.
- *AWS message generation*: both Caltrans and Mn/DOT anticipate the need to generate AWS messages within IRIS as a function of RWIS inputs (precipitation, wind speed, visibility, etc.), traffic (speed), time of day, etc.
- *Reading traffic data directly from field elements*: D1, D5, and D10 have expressed interest in reading traffic data directly from field elements. This data would then potentially be forwarded to other systems that need the data, such as PeMS.
- *Real-time verification*: requirement 217 discusses ongoing verification of correct system behavior, especially as related to CAWS functionality. A real-time application would alert operators if critical failures occurred (in real-time), such as a failure to read from the CAWS, or network failures.
- *Improved mapping*: has been discussed by Caltrans and Mn/DOT. Additional layers, the ability to label layers, display raster images, improved scalability, etc. would all be beneficial.
- *Improved cut-over to backup machine*: the ability of administrators to switch operations to the IRIS backup machine in less than 30 seconds. Presently, this is a 10 minute effort.
- *Automated end-to-end test cases*: would augment testing and be particularly useful during development to detect regression problems.
- *TMCAL*: requirement 195 discusses integration with the TMCAL system.
- *Traffic Signals*: requirement 97 discusses IRIS integration with traffic signals using a Caltrans protocol.
- *Ramp Metering*: requirements 64-74 discuss IRIS integration with the Caltrans SDRMS ramp metering protocol. IRIS uses Mn/DOT's ramp metering protocol.
- *Moving Picture Experts Group 4 (MPEG-4)*: requirement 57 discusses adding MPEG-4 support to IRIS video functionality.
- *Monitoring and Control of Portable CMS*: requirement 274 would add IRIS support for monitoring and control of portable CMS.
- *Center-to-Center Functionality*: requirement 278 targets adding center-to-center functionality to IRIS. This would provide the ability for IRIS servers to communicate between each other, exchanging information that users in other (adjacent) districts would find useful. For example, field element status (e.g. CMS, traffic, RWIS, incidents, etc.), CMS message scheduling, travel times, map updates, etc. Remote

control of field elements such as CMS would also provide a certain amount of redundancy. Ideally, if IRIS servers were deployed across Caltrans districts, an operator located in any district could view and possibly control field elements in other districts if necessary.

- IRIS integration with existing ATMS reporting functionality was explored and initial work was performed. This is anticipated to commence in the future.

# Chapter 6

## Conclusions and Recommendations

### 6.1 Strengths of the Collaborative Approach

Over the course of the study, benefits and strengths of the open-source collaborative approach were observed. Many of these benefits are well known in the open-source community and not specific to the IRIS project. Some strengths include:

- *Enhanced defect discovery*: a key benefit of the collaborative approach. Numerous defects were discovered and repaired by both Mn/DOT and AHMCT over 18 months. Defect detection and repair was roughly doubled. The use of two ticketing systems was crucial for tracking problems across time and between agencies. This increases reliability (see Section 5.5.4).
- *Lower Legal Friction*: the freely-available source code nearly eliminated legal friction. Future IRIS developers are not required to sign NDA agreements or submit to other forms of legal entanglement. This lowers costs and barriers to entry. This was experienced first-hand by the research team—in the first project scope, lawyers from the University and Caltrans failed to reach an agreement over the proprietary ATMS NDA terms *after 18 months of discussion*, approximately the amount of time spent on AHMCT IRIS development. See Section 1.4.5.
- *Low barriers to entry*: the freely-available source code lowers barriers to entry. Any engineer in any organization can contribute enhancements and repairs. The research team used a total of five individuals to contribute enhancements to the project. It is extremely unlikely this would have been possible if a restrictive NDA would have been required. If Caltrans staff from any division or consultants want to contribute to the code-base, there is nothing standing in their way. This lowers costs, increases innovation, and contributes to the rate of improvement. Moreover, the nature of the GPL license assures contributors that their enhancements will remain freely-available to others and reduces (or eliminates) information hiding [14].
- *Ease of integrating research findings*: the combination of freely-available source code and zero initial software costs lowers entry barriers for transportation researchers.

Moreover, the ease with which enhancements can be merged into the open-source project and subsequently deployed in production encourages researchers to develop production-quality deployable research. For university research, this increases transportation student interest and involvement with real-world projects, and may increase innovation across the transportation field as a whole.

- *Development process transparency*: the availability of the source code increases the transparency of the development process. This can have a positive impact on budget projections and cost estimates.
- *Increased ability to customize*: the availability of the source code enables customization to meet specific needs. The desire to have source code enhancements merged into the IRIS source tree encourages software engineers to create generalized features. These tendencies—specialization and generalization—are both encouraged by the open-source development process. Over time, this process tends to encourage *user-driven* enhancements that fit within a generalized design. This can benefit portability and reliability.
- *Zero licensing costs*: enables an organization to reallocate resources for adding new features or integrating other systems. This increases agency responsiveness to public transportation needs.
- *Multiple collaborative scenarios*: are possible. The current Open ATMS study used the development expertise of a third party (AHMCT) to provide integration and enhancement work that was guided both by the customer (D10) and Caltrans headquarters. The involvement of contributions of private contractors, multiple transportation agencies, universities, and Federal government organizations is possible. Given the internationalization enhancements to IRIS, international agency and researcher involvement is also possible. The nature of the GPL license assures stakeholders that contributions will be available to everyone in the future.
- *The network effect*: as more people become involved with open-source projects, a network effect develops, in which a single contribution benefits more and more people. This has the potential of reducing development costs further.

## 6.2 Obstacles and Concerns with the Collaborative Approach

Concerns and obstacles were encountered during the course of the project. In particular:

- *Understanding open-source*: may be a primary difficulty. Understanding the legal obligations of the GPL may be difficult to communicate to management. Also, explaining other differences from the traditional model, such as the importance of building an open-source community, can be a challenge.
- *Development dependencies*: may arise which are unavoidable. Critical path dependencies that involve other agencies should be carefully considered and planned for.
- *Coordination and planning*: become even more crucial when coordinating activities across several agencies. Three to four features were implemented twice, due to

engineers in different agencies knowingly working on the same (or related) code, resulting in 20% to 30% additional effort for that feature. In all other cases this was avoided through schedule coordination. However, if multiple agencies have overlapping deadlines that depend on modifying the same code, schedule coordination is crucial to avoid extra effort. In general, multiple engineers working simultaneously on the same code was found to work well *if* minor changes were being made. If one engineer was making major changes, other engineers usually had to manually merge their changes. In general, frequent code merging between agencies lowers these risks.

- *Caltrans dependence on other agencies due to a single IRIS source code-base:* IRIS was developed by Mn/DOT for Mn/DOT. During the course of the study, many agency specific dependencies were discovered and generalized. Agency specific dependencies included assumed constants, URLs, file names, device terminology (e.g. DMS versus CMS), protocols, algorithms, etc. All of these agency specific dependencies were eliminated, as part of the development process. This resulted the *generalization* of the source code, capabilities, and configuration options. This beneficial process is expected to continue as other agencies adopt IRIS.
- *Caltrans dependence on other agencies to fix defects due to a single IRIS source code-base:* during the course of the study, if a critical defect was discovered, the availability of the source code enabled AHMCT to fix all problems. Consultation with Mn/DOT facilitated solving complex problems. Repair of low priority defects was often performed by either AHMCT or Mn/DOT.
- *Limitation of Liability:* was an issue identified by Mn/DOT prior to open-sourcing IRIS. The GPL license explicitly states that the software covered by the license has no warranty and the developers have no liability.<sup>1</sup> However, these are legal issues and legal council should be sought to address these concerns.<sup>2</sup>
- *Integration with existing systems:* should be carefully considered if the communications protocol is proprietary. Proprietary code covered by NDAs, patents, etc., probably can not be directly integrated into IRIS. This type of integration probably needs to be implemented using stand-alone servers communicating with inter-process communication. The current research effort developed general methodologies and specific implementations for this approach.
- *Scarcity of IRIS engineers:* may be an issue for some agencies. Presently, there are approximately a total of four active IRIS developers. Increasing the number of engineers with deep IRIS knowledge was one of the factors motivating Mn/DOT to open-source IRIS. However, the researchers do not see this as a substantial concern—three developers were trained during the course of this project and the effort required is modest. The researchers also provided support for four other developers from other agencies (a DOT, a university, and a consultant), in addition to providing extensive online documentation for other developers. These training and support

---

<sup>1</sup>For the GPL see <http://www.gnu.org/licenses/gpl-2.0.txt> and <http://www.gnu.org/copyleft/gpl.html>

<sup>2</sup>For information on how GPL V3 differs from V2, see <http://www.gnu.org/licenses/quick-guide-gplv3.html>

efforts were not a burden and considered part of the collaborative open-source development model. The researchers think that the free availability of the source code and the collaborative nature of open-source development reduce these risks relative to the same risks on a proprietary software project, particularly for those covered by restrictive NDAs. In addition, hiring engineers on the open-market with Linux and Java experience is common.

## 6.3 Lessons Learned

- *Planning is crucial:* the two-stage development process was found to be effective (see Section 2.2). In terms of planning, it is straightforward to write user level functional requirements. Estimating the design and implementation effort involved is difficult without deep hands-on knowledge of the existing code. It's difficult to gain this knowledge without first doing development. Considerable skepticism is in order for initial effort estimates *if* the existing code-base and design are not well understood. As knowledge of the existing software increases, effort estimates improve. The use of a ticketing system that tracks hours is highly recommended for keeping effort estimates predictable. A go-slow approach may be the most efficient, where targeting a series of goals for completion is first achieved before long-term overarching plans are made. During the course of development, many new needs and areas that can be improved will be discovered. At this point, a more guided, top-down process may be beneficial and superior for planning. See 5.7 for effort estimates.
- *Excellent quality contributions:* from others deserve excellent contributions in return.
- *End-user feedback:* Frequent and detailed feedback from end-users (TMC operators) is crucial for: discovering defects, adjusting UI elements to eliminate confusion or problems, and determining the relative importance (priority) of enhancements. This was crucial for the software engineers and D10 TMC management. The software engineers found that on-site visits with TMC personnel are crucial for effective feedback. If the feedback process is effective, operators are more likely to suggest improvements and identify problems because they have confidence that improvements will be made.

## 6.4 IRIS Strengths and Areas for Improvement

Over the course of 18 months, IRIS' strengths and areas that would benefit from improvement were identified. Some noteworthy IRIS strengths are:

- *Rate of improvement:* since May of 2007, the rate of improvement has been high. For example, the first version (alpha1) did not build (due to recursive build problems), and portions of the user interface were fragile. These problems have been solved

and subsequently many new features have been added, all RMI code has been removed (boosting performance and reliability), and many implicit Mn/DOT-specific configuration assumptions have been generalized. The rate of improvement is expected to remain high, with additional contributions from future agencies contributing to the rate of improvement.

- *Ability to tailor*: the availability of the source code enables agencies to customize and tailor IRIS for their specific needs. System attributes, internationalization, the help system, and property files all provide configuration flexibility. Code enhancements and modifications that are contributed back into the code-base are more valuable if they are generalized and can be used by other agencies.
- *Scalability*: testing results indicate that IRIS can support many simultaneous users (see Section 3.7).
- *Reusability*: is high as a result of 1) a single source code-base, 2) an emphasis on configurability, and 3) use and testing by multiple agencies. High reusability decreases costs and improves reliability.
- *Reliability*: is high. Over the course of 18 months, the D10 IRIS server did not fail. Networking and configuration problems are much more likely to be the source of any concern, due to this high reliability.
- *Strength of lead agency*: Successful open-source projects have strong leadership. As demonstrated on this project, Mn/DOT has a very strong commitment to IRIS quality, continued development and improvement. The IRIS team at Mn/DOT was particularly responsive, flexible, highly-skilled, motivated, and demonstrated strong leadership during the project.
- *The IRIS driver interface*: is flexible and well-designed. This facilitates interfacing IRIS with hardware and external software systems.

Areas that would benefit from improvement are:

- *Mapping*: IRIS would benefit from enhanced map functionality. This includes: support for large map files (see Section 3.7), support for raster layers, attribute labeling, spatial search, and other enhancements.
- *Data loader*: the ability for new agencies to load infrastructure data into IRIS in bulk would reduce the initial system configuration effort. Without bulk data loading, IRIS is configured and loaded with data through the client user interface. However, this is a one-time effort and presently can be done through manual database commands, if necessary.
- *End-to-end automated test cases*: both the development process and the reliability of deployed releases would benefit from development of automated end-to-end test cases. The automated test cases would be executed after each new enhancement (or defect repair) was incorporated into the code-base. This would help ensure that new features don't break existing features. If many agencies are using IRIS, this capability may become crucial. See Section 3.8.

- *User interface refinement:* The IRIS client would benefit from user interface refinements that increase consistency and reliability: for individual IRIS users, the client occasionally freezes, requiring that individual user to restart the client. When this happens, no other users are effected; i.e. only the one client freezes, and the server is unaffected. The level of reliability experienced on the IRIS server has been very high, with almost no down time in more than 19 months. Achieving and maintaining this same level of reliability on the client is important.

## 6.5 Conclusions

The goal of this study was to dramatically reduce initial and ongoing ATMS life-cycle costs. A collaborative approach was used that implemented an existing open-source ATMS within Caltrans District 10 (D10), using commodity x86 hardware.

Compared with the use of a proprietary ATMS, this approach was found to be extremely effective at providing an ATMS system with substantial capabilities at minimal cost. D10 acquisition costs were 98% less than the existing proprietary ATMS (see Section sec:acquisitioncosts). Annual maintenance costs are anticipated to be 68% to 86% less than the existing proprietary ATMS. Annual personnel maintenance costs are estimated to be 1/5 to 1/3 of the current system, primarily due to 1) the ability to share maintenance efforts across contributing agencies, and 2) the unrestricted availability of source code, which is anticipated to keep maintenance costs competitively low by lowering barriers to entry and eliminating the need for sole-source contracts. A total 5-year cost comparison for a single agency shows a 72% cost reduction for the open-source approach.

There is no evidence that these cost savings would result in inferior software, features, reliability, or innovation for systemic reasons. The evidence is strong that the collaborative open-source approach would strengthen results in these dimensions.

IRIS was shown to be very scalable, easily supporting 50+ simultaneous users on a single, modest, commodity server, priced at less than \$1,000.

Server reliability has been high over the last 19 months, with zero service outages due to IRIS server defects. Networking and configuration problems were responsible for any service outages. The client defects resulting in occasional freezing and premature client closure need to be addressed and do not appear to be a result of systemic problems with the collaborative approach.

There is strong evidence that the open-source approach will strengthen innovation, providing a means for researchers to deploy research directly into a production system with minimal overhead. These innovations would be immediately available to all agencies using IRIS.

Districts 10, 5, and 1 have shown a strong ability and willingness to transition to providing their own routine IRIS problem resolution and support. The potential of sharing this technical information and know-how with others in the IRIS community encourages



the incubation of internal technical expertise.

Agencies are faced with a difficult choice: nurture in-house software development expertise, with associated risks and without economies of scale, *or* benefit from economies of scale by purchasing access to a proprietary ATMS and giving up internal technical expertise *and* control. The open-source approach cuts a middle path by enabling DOTs to nurture and develop in-house expertise while leveraging the benefits of shared development and economies of scale with other organizations. It also enables the possibility of using third parties to develop and maintain the system, while keeping bidding competitive. This project used the development expertise of a third party (AHMCT) to perform integration and enhancement work.

The combination of unhindered access to existing and future source code (stipulated by the GPL license) provides a new pathway for public transportation agencies to interact with the private sector. Enhancements to IRIS source code (derivative works) must be publicly shared, enabling a public agency to hire a private consultant for integration and enhancement tasks, free of legal entanglements and NDAs. This mechanism enables preservation of public ownership and allows the private sector to contribute with transparency.

The positive results from this study and the dynamic rapidly evolving nature of the ITS field are strong indicators for future positive developments. For ongoing work and progress in these areas, see the project web site.<sup>3</sup>

## 6.6 Recommendations

This report has discussed results and conclusions from a multi-year research demonstration study that implemented and extended an existing open-source ATMS within Caltrans District 10.

Software can be viewed as a product or a type of knowledge [6]. Viewing software as a type of knowledge naturally prompts questions of how best to extend, share, preserve, and expand that knowledge. Encouraging and supporting collaboration is crucial—if collaboration is not easy, people will not do it. The benefits observed in this study are the result of low-friction collaboration.

As the benefits observed during the course of this research study are attractive, the following recommendations may be useful. They are intended to achieve the benefits discussed above.

- Technical recommendations:
  - Promote and encourage the use of software development languages that have strong cross-platform capabilities. For example, Java applications can be de-

---

<sup>3</sup>For the project web site see <http://iris.ahmct.ucdavis.edu>

veloped on and moved between many different operating systems. Scripting languages such as Python, Perl, and PHP have strong cross-platform capabilities. Platform specific languages like C# are less portable. Portability provides a path for server applications to be migrated to open-source operating systems, which lowers costs.

- Use end-to-end encryption for communication with security-sensitive field elements, such as CMS.
- Eliminate the use of proprietary protocols, which:
  1. Are potential security risks, if protocol secrecy is used to provide security, otherwise known as “security through obscurity.” This approach was discredited in 1883 [13].
  2. Encourage information hoarding, which reduces innovation.
  3. Increase lock-in and lack of choice, which increases costs.
  4. Cannot be open-sourced.
- For projects that would benefit from inter-agency cooperation and the shared development model:
  - Take concrete steps to support collaboration, such as using open-source licenses for software code.
  - Collaborate with other agencies.
  - If possible, use and extend existing open-source transportation projects, rather than starting from scratch. This lowers development risk.
- For existing proprietary projects built with proprietary software:
  - Investigate replacing individual components with open-source versions. For example, replacing a proprietary database with MySQL or PostgreSQL. There are also proprietary products built on top of open-source products. For example, EnterpriseDB offers Oracle application compatibility for an order of magnitude less cost than Oracle.
  - Investigate building new components with open-source applications. The open-source applications would communicate with proprietary applications using inter-process communication, as was done on this project.
  - Investigate open-sourcing existing proprietary applications. Other transportation agencies may be interested in your application and be willing to contribute.
- For new projects:
  - Consider using an open-source license, particularly with consultants. This increases transparency, and guarantees that derivative works remain open-source, regardless of who works on them.
  - Monitor ongoing project progress via the source code repository. The contents of the repository reveal much about development progress (see Figure 5.10, Figure 5.11, and Figure 5.12).
  - Use bug tracking software, e.g. Trac or Bugzilla, from the outset of the project.

- Avoid using hardware that uses proprietary protocols. This increases the difficulty and complexity of interfacing the hardware with an open-source application.
- For agencies considering IRIS:
  - Start with small-scale enhancements and increment based on success. The accuracy of effort estimates should increase over time.
  - Select the highest-priority functional area, with a goal of getting IRIS up and running.
  - Discuss short and long-term plans with IRIS software engineers and get feedback and ideas from them.
  - Download the IRIS source and try it out—there is zero risk.
  - Ensure (perhaps contractually) that enhancements made to IRIS will be coordinated with the IRIS lead engineers (Mn/DOT). If this is done correctly, the enhancements will be effortlessly incorporated into the main IRIS code-base by Mn/DOT. This collaborative process requires extra effort during the design and implementation phases, perhaps 10%-30%. However, ensuring that new enhancements are incorporated into the main IRIS code-base has these benefits:
    1. Enhancements must meet quality standards to be incorporated into the main IRIS code-base.
    2. Enhancements incorporated into the code-base are much more likely to be generalized features, if possible.
    3. Enhancements incorporated can be directly used by other agencies, increasing the likelihood they will be used, which encourages subsequent enhancements from other agencies.
    4. Enhancements incorporated into the code-base do not have to be manually re-merged with each subsequent IRIS release, which lowers software maintenance costs.



# References

- [1] AHMCT, University of California. Caltrans IRIS Repository. URL <http://iris.ahmct.ucdavis.edu/hg>. Accessed Mar, 2009.
- [2] AHMCT, University of California, Davis. Research & Development of Open-Source Advanced Traffic Management System Hardware and Software Components. Technical Report TS-517, AHMCT, February 2005.
- [3] AHMCT, University of California, Davis. Caltrans SOCCS Design Description, . URL [http://iris.ahmct.ucdavis.edu/SOCCS\\_Design\\_Description.pdf](http://iris.ahmct.ucdavis.edu/SOCCS_Design_Description.pdf). Accessed March, 2010.
- [4] AHMCT, University of California, Davis. AHMCT Open ATMS Project Wiki, . URL <http://iris.ahmct.ucdavis.edu/mediawiki>. Accessed Mar, 2009.
- [5] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. IRIS Verification Procedure for the California Department of Transportation District 10. Technical report, AHMCT, March 2000. URL <http://www.ahmct.ucdavis.edu>.
- [6] 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. URL <http://www.ahmct.ucdavis.edu>.
- [7] 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. URL <http://www.ahmct.ucdavis.edu>.
- [8] 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. URL <http://www.ahmct.ucdavis.edu>.
- [9] 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. URL <http://www.ahmct.ucdavis.edu>.

- [10] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. Overview of Caltrans District 10 IRIS Demonstration Design. Technical Report UCD-ARR-07-12-31-02, AHMCT, December 2007. URL <http://www.ahmct.ucdavis.edu>.
- [11] M.T. Darter, S.M. Donecker, K.S. Yen, B. Ravani, and T.A. Lasky. IRIS Verification Plan for the California Department of Transportation District 10. Technical report, AHMCT, March 2009. URL <http://www.ahmct.ucdavis.edu>.
- [12] M.T. Darter, T.A. Lasky, D. Lau, C. Gregory, and B. Ravani. Implementation and extension of the iris open-source traffic management system to improve organizational performance. In *Transportation Research Board 88th Annual Meeting Proceedings*, Washington, D.C., 2009.
- [13] Auguste Kerckhoff. La cryptographie militaire. *Journal des Sciences Militaires*, (5): 5–38, January 1883. URL <http://petitcolas.net/fabien/kerckhoffs/>.
- [14] Josh Lerner and Jean Tirole. Some Simple Economics of Open Source. *The Journal of Industrial Economics*, 50(2):197–234, March 2003.
- [15] Minnesota Department of Transportation. Intelligent Roadway Information System (IRIS) As Built System Design Document. Technical report, Minnesota Department of Transportation, June 2007.
- [16] Minnesota Department of Transportation. Mn/DOT IRIS Repository. URL <http://iris.dot.state.mn.us/hg>. Accessed Mar, 2009.
- [17] Minnesota Department of Transportation IRIS Project. Mn/DOT IRIS JavaDoc, December 2007.
- [18] Minnesota Department of Transportation IRIS Project. Mn/DOT IRIS source code, December 2007.
- [19] J. D. Mooney. Strategies for supporting application portability. *Computer*, 23(11): 59–70, November 1990.
- [20] California Department of Transportation. Systems Engineering Guidebook For ITS. Technical report, Caltrans, February 2005.
- [21] Open Source Initiative. The Open Source Definition. URL <http://opensource.org/docs/osd>. Accessed Jan, 2009.
- [22] Jim Shore. Fail fast [software debugging]. *Software, IEEE*, 21(5):21–25, September 2004. URL [http://ieeexplore.ieee.org/xpls/abs\\_all.jsp?arnumber=1331296](http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1331296).
- [23] Wavetronix LLC. SS105 SmartSensor Data Protocol V2.02, February 2004. URL <http://www.wavetronix.com/>.

# Appendix A

## Functional and Non-functional Requirements

This appendix lists tickets in the Caltrans IRIS ticket system. This includes tasks, defect tickets, and functional and non-functional requirements. See Section 2.2 on page 12 for a discussion of requirements. This list was generated in January of 2010. For an updated list, see the ticket system.

Table A.1: Tasks, Functional, and Non-functional Requirements.

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
1	Traffic	Traffic-related functionality will be used as-is within the IRIS client.	closed
2	Traffic	The IRIS client will display a D10 map.	closed
3	Traffic	D10 IRIS will receive UDP/IP packets from a D10 router, containing real-time traffic data from inductive loop and microwave traffic detectors.	closed
4	Traffic	D10 IRIS will format real-time traffic data received from a D10 router for input into IRIS.	closed
5	Traffic	IRIS traffic-related functionality shall operate as-is.	closed
6	Traffic	Real-time traffic data from microwave and inductive loop detectors will be used by IRIS.	closed
7	Traffic	(optional) The IRIS TRAFDAT applications will be implemented.	new
8	Traffic	If implemented, the IRIS TRAFDAT applications will operate as-is.	new
9	Traffic	If TRAFDAT is implemented, D10 IRIS-specific TRAFDAT reports and plots are anticipated, depending on sub-task priority. For example, TRAFDAT reports specific to CAWS.	new
10	Traffic	D10 IRIS is not anticipated to impact existing D10 hardware or software performance.	closed
11	Traffic	D10 IRIS will use the existing IRIS database schema as-is.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
12	Traffic	D10 IRIS will use existing IRIS data formats whenever possible.	closed
13	Traffic	Developed traffic functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	closed
14	Traffic	AHMCT and D10 will test developed functionality both in the laboratory and in the field.	closed
15	Traffic	D10 IRIS traffic functionality will operate in parallel with existing D10 traffic management software.	closed
16	Traffic	D10 IRIS and existing D10 TMC functionality are not interdependent in any way.	closed
17	Traffic	D10 IRIS will use, and be dependent on existing IRIS and D10 security features.	closed
18	Traffic	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	closed
19	CMS	(optional) D10 IRIS will indicate to TMC operators the origin of CMS messages (see the background section for potential message origin).	closed
20	CMS	D10 IRIS shall use existing IRIS functionality, such as support for the NTCIP protocol.	closed
21	CMS	D10 IRIS shall support CMS on dial-up lines.	closed
22	CMS	D10 IRIS shall support CMS connected to the TMC via D10s cell modems.	closed
23	CMS	D10 IRIS will not use existing IRIS serial tunneling functionality.	closed
24	CMS	D10 IRIS shall implement the CMS protocol in an application separate from IRIS.	closed
25	CMS	D10 IRIS shall support CMS that use the Caltrans CMS protocol.	closed
26	CMS	D10 IRIS shall interface with CMS on dial-up lines through a terminal server, as in the existing D10 SOCCS system.	closed
27	CMS	The D10 IRIS UI shall function as is (see the Background section above).	closed
28	CMS	D10 IRIS shall receive and process D10 traffic management software generated messages (CAWS) and display them on the CMS.	closed
29	CMS	Developed CMS applications and functionality shall comply with the existing protocol, to the extent possible.	closed
30	CMS	Developed NTCIP applications and IRIS functionality shall comply with the existing protocol, to the extent possible.	closed
31	CMS	Developed CMS functionality will support a subset of the full Caltrans CMS protocol.	closed
32	CMS	Developed CMS functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	closed

Continued on Next Page...



Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
33	CMS	AHMCT will test developed functionality both in the laboratory and in the field.	closed
34	CMS	AHMCT will develop a formal internal CMS protocol document using information from 1) verbal communication with Caltrans engineers, 2) from viewing previously developed SOCCS source code, 3) from internal Caltrans documentation, and 4) from testing of AHMCT-developed code.	closed
35	CMS	Existing D10 traffic management software and CAWS applications are not expected to change.	closed
36	CMS	Existing D10 CMS control (the SOCCS system) will act as a backup CMS control system.	closed
37	CMS	Existing CMS code will not be shared outside of AHMCT or Caltrans.	closed
38	CMS	The CMS protocol will not be shared outside of AHMCT or Caltrans.	closed
39	CMS	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	closed
40	CMS	Existing CMS code is copyrighted by Caltrans.	closed
41	CMS	Existing CMS code is not subject to the GNU's General Public License (GPL) copyright.	closed
42	CMS	Newly-developed code that uses the CMS protocol will not be released outside of AHMCT or Caltrans.	closed
43	CMS	Newly-developed code that uses the CMS protocol shall not be copyrighted with the GPL.	closed
44	Incidents	D10 IRIS shall use existing IRIS mapping UI functionality.	closed
45	Incidents	D10 IRIS shall convert the California Highway Patrol (CHP) eXtensible Markup Language (XML) file into a data format supported by IRIS.	closed
46	Incidents	D10 IRIS shall periodically read real-time CHP incident data.	closed
47	Incidents	CHP incident data will be read from a web server as an XML file via HyperText Transfer Protocol (HTTP).	closed
48	Incidents	D10 IRIS incident mapping functionality shall operate as-is.	closed
49	Incidents	D10 IRIS will perform geographic coordinate translation between supplied CHP mapping coordinate system (Thomas Brothers XY) and the coordinate system used by IRIS (Universal Transverse Mercator (UTM)).	closed
50	Incidents	Developed incident mapping functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	closed
51	Incidents	IRIS incident mapping is independent of any existing D10 systems and operations.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
52	Incidents	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	closed
53	CCTV	The IRIS user interface will be used as is.	closed
54	CCTV	D10 IRIS shall support video monitoring of D10 IP-based cameras.	closed
55	CCTV	D10 IRIS shall support TMC-based camera control of D10 IP-based cameras that support motion control.	closed
56	CCTV	(optional) D10 IRIS shall support video monitoring of the existing D10 channel City of Stockton analog system.	closed
57	CCTV	(optional) D10 IRIS may support Moving Picture Experts Group 4 (MPEG-4) encoding, pending other priorities.	new
58	CCTV	D10 IRIS shall use primarily M-JPEG or MPEG-4 for digital video encoding.	closed
59	CCTV	Developed video functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	closed
60	CCTV	AHMCT will test developed functionality both in the laboratory and in the field.	closed
61	CCTV	Any developed D10 IRIS video monitoring functionality providing City of Stockton video monitoring is anticipated to operate in parallel with existing D10 City of Stockton systems.	closed
62	CCTV	Any developed D10 IRIS video control functionality providing control of existing City of Stockton cameras is anticipated to operate in parallel with existing D10 City of Stockton systems.	closed
63	CCTV	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	closed
64	Ramp Metering	D10 IRIS enhancements to IRIS supporting multiple ramp meter protocols will be modular to the extent possible, anticipating the addition of future ramp meter protocols to the IRIS project.	new
65	Ramp Metering	D10 IRIS shall use IRIS ramp metering functionality as is, with the exception of the communications protocol.	new
66	Ramp Metering	D10 IRIS may support a subset of IRIS ramp metering functionality, depending on the priority of other Requirements.	new
67	Ramp Metering	D10 IRIS shall use the SDRMS protocol.	new
68	Ramp Metering	The in-field controller shall support and conform to the SDRMS protocol.	new
69	Ramp Metering	Implemented modules or applications within D10 IRIS shall support and conform to the SDRMS protocol.	new

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
70	Ramp Metering	Developed ramp meter functionality will support a subset of the full SDRMS protocol.	new
71	Ramp Metering	Developed ramp meter functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	new
72	Ramp Metering	AHMCT will test developed functionality both in the laboratory and in the field.	new
73	Ramp Metering	D10 IRIS ramp meter testing will involve lab testing, field testing, and TMC testing.	new
74	Ramp Metering	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	new
75	RWIS	Weather Stations and Sensors: D10 IRIS UI enhancements to IRIS will be consistent with the existing UI.	new
76	RWIS	D10 IRIS will receive weather station data from an existing D10 traffic management software server.	new
77	RWIS	D10 IRIS shall display the locations of weather stations on the System map.	new
78	RWIS	The D10 IRIS map shall indicate current wind speed and visibility measurements.	new
79	RWIS	Developed weather station functionality will use existing IRIS features to monitor reliable operation (e.g. the event log database).	new
80	RWIS	AHMCT will test developed functionality both in the laboratory and in the field.	new
81	RWIS	Existing D10 weather station functionality will not be dependent on D10 IRIS.	new
82	RWIS	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	new
83	Reports	Event Logging: D10 IRIS logging shall be consistent with the existing UI.	closed
84	IRIS Reports	D10 IRIS shall log trigger-based weather station events, e.g., visibility and wind speed reaching trigger values.	new
85	Reports	D10 IRIS shall log other sensor events that are relevant to D10, according to their relative sub-task priority.	closed
86	Reports	D10 IRIS shall use the existing IRIS log database schema.	closed
87	Reports	D10 IRIS shall add additional events and devices specific to D10 as needed.	closed
88	Reports	Logging functionality will use existing IRIS features to monitor reliable operation (e.g. assertion reporting, the event log database).	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
89	Reports	Logging functionality is used in most functional areas and will therefore be tested across those functional areas, both in the laboratory and in the field where appropriate.	closed
90	Reports	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	closed
91	HAR	HAR and EMS: (deferred) D10 IRIS UI enhancements to IRIS will be consistent with the existing UI.	new
92	HAR	(deferred) D10 IRIS shall interface to the Quixote DR2000 SIM module.	new
93	HAR	(deferred) D10 IRIS shall interface to the Quixote DR2000 SIM module over the TMC local area network.	new
94	HAR	(deferred) D10 IRIS shall display HAR and extinguishable message sign locations on the IRIS system map.	new
95	HAR	(deferred) D10 IRIS shall display current HAR messages for each HAR unit. The message display may be symbolic or literal. For example 'WAV Msg 14' or 'Slow Speed Ahead'.	new
96	HAR	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	new
97	HAR	(deferred) D10 IRIS shall support non-realtime monitoring of traffic signals using the Caltrans Central Signal Control System (CTNet) protocol.	new
98	HAR	(deferred) D10 IRIS shall support real-time monitoring of traffic signals using the CTNet protocol.	new
99	HAR	(deferred) D10 IRIS shall support real-time control of traffic signals using the CTNet protocol.	new
100	HAR	Software source code will be maintained in a source control repository to support ongoing and future software maintenance.	new
101	CMS	Ability for operators to easily create and send free-form CMS messages.	closed
102	CMS	Ability for operators to send messages to CMS connected via terminal server and modem i.e. dial-up.	closed
103	CAWS	Operators need existing D10 CAWS functionality integrated with IRIS. IRIS should read the real-time CAWS messages (via HTTP) and send the messages to CMS.	closed
104	CMS	Operators need the ability to view CMS history in report(s).	closed
105	CMS	Operators need the ability to specify a font per CMS message.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
106	CMS	Operators need the ability to quickly specify existing library messages by ID and send them to the CMS. This functionality will populate the message line fields and font field using existing messages from a quick message library.	closed
107	CMS	Ability for operators to easily send a single CMS message to an ad-hoc grouping of signs. The message can be 1) quick message, 2) free-form, or 3) from message line library.	closed
108	CMS	Each CMS message that is sent needs to be archived in a database for future reference and reporting.	closed
109	CMS	Easy access to CMS message history (via queries) for various time periods. For example, last 24 hours, last 7 days, last 30 days, etc.	closed
110	CMS	Ability to generate a CMS activity report sorted by CMS, for CMS activity over a fixed time frame.	closed
111	CMS	Display CMS messages with identity of originating system (SOCCS, IRIS).	closed
112	CMS	Instant View of CMS status including failed.	closed
113	CMS	Spreadsheet type view of CMS	closed
114	CMS	Somewhat easy to Add, Modify, Delete, CMS including all configuration, including whether part of CAWS.	closed
115	CMS	Ability to view and edit which CMS belong to which sign groups within a single dialog box: e.g. show/edit which CMSs belong to Star Campaign sign group.	new
116	CAWS	Easily identify if a CMS is in CAWS.	closed
117	CMS	Add the ability for operators to see easily if a message is mis-sized before it is sent to the sign.	new
118	CMS	Ability to re-initialize Model 170 controller.	closed
119	CMS	Ability to re-Initialize cell modem.	closed
120	CMS	Handles Model 520	closed
121	CMS	Easy update (view) of CMS message	closed
122	Maps	Improve IRIS mapping	new
123	CMS	Allow selection of pre-existing group of CMSs easily to send same message (Custom or Library).	closed
124	CMS	Message simulation / preview ability without sending the message to the CMS.	closed
125	CMS	Extend message preview ( #127) to show selected font	closed
126	CMS	CMS Message Width Warning.	new
127	User Accounts	Easy to add new User and specify permissions.	closed
128	CMS	Ability to edit Library/CMS group name; Or to copy group with new name	new
129	CMS	Ability to spell check messages, including abbreviations.	new
130	CMS	Ability to display network health over time.	closed
131	CMS	Contains Message Library that is easy to edit	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
132	CMS	Save free form messages into message library.	closed
133	Amber Alerts	Ability for AMBER Alerts to overwrite operator messages already on the sign.	closed
134	Maps	Ability for clicked map element to provide ID information.	closed
135	CMS	Multiple users can simultaneously use CMS functionality.	closed
136	Maps	Display persistent map labels that can be turned on and off. Presently, users must hover over streets to determine the street name via a popup balloon.	new
137	Travel Time	Travel time calculation uses traffic history if detector data unavailable	new
138	CCTV	Video plays indefinitely	closed
139	CCTV	Provide user option to play video indefinitely or specified time.	closed
140	User Accounts	Provide LDAP administration capabilities via IRIS, so admins can add new user names remotely.	new
141	CCTV	Ability to display video on video wall.	closed
142	RWIS	Display RWIS real-time data on RWIS map tab.	new
143	Ramp Metering	SDRMS ramp metering support.	new
144	Travel Time	D10 traffic detector display working reliably in IRIS, so that travel time message generation works reliably.	closed
145	Incidents	Display IRIS incidents reliably. Presently there appears to be some kind of parse problem within IRIS of the CHP XML feed.	closed
146	User Accounts	IRIS configured with different capabilities for different users.	closed
147	Maps	Display street names in balloon user hovers over with the mouse.	closed
148	Travel Time	Perform MITTENS vs IRIS travel time comparison.	new
149	IRIS UI	Display symbolic roadway information: on/off ramps, VDS, etc.	closed
150	CMS	CMS text messages need to be vertically centered.	closed
151	CCTV	Users need preset views for camera positions: 4 minimum, 6 preferred. This includes zoom settings.	closed
152	CCTV	High speed flashing issue w/ cameras.	closed
153	CAWS	Add ability for operator CMS messages to overwrite existing CAWS messages.	closed
154	Travel Time	Calculation and display of travel times.	closed
155	HAR	Users needs to locate HAR stations on the map, monitor status, activate messages.	new
156	EMS	Users need to locate EMS (extinguishable message sign) station on the map, monitor status, activate signs.	new
157	Traffic	Users need to monitor status, upload parameters, locate signals on map.	new
158	General	IRIS Server verifies database version is correct on startup.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
159	CMS	Ability to type CMS messages in free form.	closed
160	CMS	Add the ability for CMS free form messages to display message completion possibilities as a function of what the user has typed and available entries in the libraries. This is often called "type ahead ability" or "predictive text".	new
161	WIM	Add integration with Weigh in Motion system.	new
162	CAWS	D10 needs to know IRIS CAWS functionality has been tested and validated (see also #103).	closed
163	CMS	Determine if the error message from the 170 'Invalid Response Code' should be handled differently than it presently is.	closed
164	CMS	Ability to change CMS polling time w/o rebuilding IRIS.	closed
165	Travel Time	Activate message start/stop time for travel time messages for a CMS w/ SV170 V3.4.	Requirements
166	General	Increase the robustness of the IRIS client and server to networking problems.	closed
167	Maps	Improve existing maps used in IRIS.	closed
168	General	Ability to run the IRIS client on the IRIS server machine.	closed
169	Incidents	Users need to know IRIS has been validated for incident reliability compared with existing systems. See also #145.	closed
170	CMS	Add vertical space between 2 line CMS message lines. Is there a statewide standard?	closed
171	Traffic	Enhance IRIS lane numbering so that it conforms to Caltrans convention, rather than Mn/DOT convention.	new
172	CMS	Add ability for other districts to see IRIS placed messages via the DRI Google Earth state-wide system (Sean Campbell's system).	new
173	Maps	Add ability for operators to search for street names.	new
174	CMS	Add queueing for modems to sensorserver. This is a reliability feature. For example, sending an AMBER alert to all CMS on dial-up should queue on the modem line.	closed
175	Documentation	D10 take over Requirements–D10 needs sufficient documents and knowledge transfer to maintain IRIS, diagnose problems, etc.	Requirements
176	Maps	Display latitude and longitude, or UTM X, Y coordinates of mouse pointer in a toolbar.	closed
177	Maps	Ability for operators to see content of the Lakes and Streams layers in IRIS.	new
178	Amber Alerts	Provide ability for AMBER Alerts to be sent to a predefined list of CMS. Each CMS could be included or excluded from this list.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
179	CAWS	Add a new CMS message category 'CAWS' and CMS radio button in DMSTrafficManager that would group together all CMS w/ deployed CAWS messages.	closed
180	CMS	Provide the ability for an operator to leave a note, per CMS, that other operators could see.	new
181	CMS	Provide recent event history, per CMS, inside the IRIS client.	new
182	CAWS	Provide a toolbar which lists all CMS that have been removed from CAWS control.	closed
183	CAWS	Have deactivated CAWS CMS automatically reactivated after a certain time period.	closed
184	CMS	Add the ability for IRIS users to specify a font per message page in multipage messages. Presently a single font is used for all pages.	closed
185	Incidents	Add ability for operators to see incidents from the restricted incident "media feed" rather than public CHP feed.	new
186	Incidents	Use incident feed from future CAD system, which will be available in 3-5 years.	new
187	CAWS	Add existing CAWS traffic management software functionality to IRIS	new
188	Traffic	Add the ability to read real-time traffic data directly from detectors	new
189	CMS	IRIS connects to TMCAL for logging.	new
190	CMS	Add a new report that shows the number of CMS messages sent per month	closed
191	CMS	Produce a periodic report that is readable in GoogleEarth and Google Maps (kml/kmz files).	closed
192	IRIS Client	The IRIS client needs to fit on a smaller screen	closed
193	CMS	CMS reporting that stores tabular report indefinitely.	closed
194	CAWS	Add auto-action 1: if a CMS is removed from CAWS control (deactivated) and if the sign is sent a blank message, IRIS automatically reactivates the sign.	closed
195	CAWS	Add auto-action 2: if IRIS has a CAWS message to send and the sign is deactivated and blank, IRIS automatically reactivates the sign and IRIS sends the CAWS message.	closed
196	CMS	Ability to generate a report showing the contents of the IRIS Quick Message Library, sorted by ID.	closed
197	CMS	Validate IRIS communications reliability compared with SOCCS. Users need to know IRIS communication reliability relative to SOCCS.	closed
198	CAWS	Add ability for administrators to specify the number of retry attempts for CAWS failures and regular message retry attempts.	closed

Continued on Next Page...



Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
199	CMS	Ability for operators to specify a page display time per page for multi-page messages in the quick message library. Presently the default page display time (2.5 secs) is being used for all messages.	closed
200	CMS	Add message preview capability for 1) Amber alerts and 2) Quick Message Library. Also, add the ability to see the 2nd page of message previews.	closed
201	CMS	Users need an easy way to clear all CMS message comboboxes on all pages.	closed
202	IRIS Client	Users need ready access to an online help system based on the current screen (context).	closed
203	IRIS Client	Enhance user permissions to be group based, rather than role based (like in the ATMS system). Groups would contain multiple roles.	closed
204	IRIS Client	Add ability to easily open controller page in web browser. This will be done via adding a button to the controller dialog box that opens the URL of the controller in a web browser.	new
205	IRIS	Add ability to track administrative changes: a new log would indicate who, when, and what administrative change was made. E.g. for tracking when CAWS was enabled/disabled.	new
206	CMS	Ability for IRIS administrators to change CMS field controller IP addresses within the IRIS client.	new
207	IRIS	Quantitatively track IRIS server uptime and scalability. Origin of request: AHMCT, Mn/DOT, Jeff Mcrae, scalability charts for final report.	closed
208	Reports	Users need the ability to view IRIS logs from sensorserver, trafserver, iris, casper, etc. in a GUI based log viewer that provides filtering and sorting capabilities. This facilitates IRIS administration and problem determination and resolution.	closed
209	IRIS UI	Add menu item in IRIS that when clicked links to the Trac management system.	closed
210	Reports	Format all IRIS logs so they can be read by a GUI log viewing tool such as Apache Chainsaw.	closed
211	CAWS	Add the ability for IRIS to perform real-time validation of critical system functions such as validating that CAWS messages were sent, the message file was received, etc. A watchdog application would alert operators if critical failures such as a failure to read the CAWS message file.	new
212	CMS	Add a simple user interface to the sensorserver application, that would contain a combobox for cms selection, buttons for sending messages, reinitializing, getting status, etc. The focus is in providing an application for use in the field for testing CMS.	new
213	User Accounts	Create New User	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
214	User Accounts	Enhance logging to display login information and reason for failure	closed
215	User Accounts	Create Trac User George Anzo	closed
216	CCTV	Operator asked if there was a problem with video—couldn't see video feed	closed
217	General	Resolve boot configuration issues with D10 IRIS server	closed
218	CMS	R8.9 sensorserver cumulative comm report problem	Requirements
219	CCTV	TMC can't see video stream from cameras	closed
220	CMS	fix sensorserver SV170 parse defect related to timeouts	closed
221	CMS	In sensorserver, fix zero-f defect.	closed
222	CMS	Create one or more simulated CMS for testing and operator training	closed
223	Reports	Create IRIS uptime log	closed
224	CMS	Updating sensorserver to support SV3.4	closed
225	CMS	Fix unusual timeout defect between IRIS and sensorserver	closed
226	CAWS	Enhance caws log to include blank messages.	closed
227	CAWS	Fix defect: skipping of sending CAWS messages at 4AM.	closed
228	IRIS Client	Add context sensitive help system to the IRIS client	closed
229	IRIS Client	Add new help menu item that loads the IRIS Trac system web page	closed
230	SONAR	Merge w/ IRIS 3.80-88 (SONAR conversion)	closed
231	CMS	Change operation timeout from 10sec to 20sec in Iris server	closed
232	General	TMC operator can't find posted message in the log	closed
233	CMS	cms 81 not on google map	closed
234	CMS	Icons out of place	new
235	Maps	Adjust CMS sign icon locations	closed
236	CAWS	Identify CAWS CMS by triggers; eg weather, speed, or both	new
237	Maps	Zoom and/or move all Map views	closed
238	CMS	Adjust sensorserver timeout	closed
239	User Accounts	ADD NEW USER	closed
240	Traffic	Resolve VDS detector force fail problem	closed
241	CAWS	CAWS validation on CMS V39	closed
242	IRIS	IRIS client disconnect from server due to SSL MAC exception	closed
243	IRIS	IRIS client disconnect from server due to client/server communication mismatch	closed
244	CMS	Can't create a message with a 3rd line on the 2nd page	closed
245	CMS	Ad-hoc multi-selection in IRIS 9 is more difficult	new
246	Maps	Operators can't zoom the map	closed
247	CMS	IRIS client crashes while performing CMS selection.	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
248	CAWS	CMS that have been blanked by CAWS are incor- rectly classified	closed
249	CMS	CMS message author incorrectly changes to 'Oth- erSystem'	closed
250	IRIS	Users can't log in	closed
251	IRIS Client	Client window Size and location persistence	new
252	Reports	CAWS blank messages being double-logged	closed
253	Incidents	Integration of IRIS with TMCAL	new
254	CMS	Load new quick message library that contains page on times	closed
255	CCTV	Video from cameras not visible in IRIS client	closed
256	Maps	Map street name hover text not visible in client	new
257	IRIS	Merging with IRIS 3.90.0	closed
258	IRIS	GetStatus requests are generating extra log mes- sages	closed
259	IRIS	Merging with IRIS 3.91.0	closed
260	IRIS UI	Message preview not correctly displaying quick messages	closed
261	CMS	Design improvement for code that calculates CMS text position	closed
262	IRIS	Merging with IRIS 3.92 and 3.93	closed
263	CMS	Editable CMS line comboboxes should have stan- dard click behavior	closed
264	IRIS Client	Zoom button on the toolbar permanently become gray and unclickable	closed
265	IRIS Reports	No Active Cameras 0 / 7	new
266	Reports	Reduced size of D10 maps by 50%	closed
267	IRIS Client	Synchronization issue in client on startup.	closed
268	CMS	OtherSystem message is categorized as CAWS De- ployed	closed
269	IRIS Reports	Duplicate message logging due to page on-time	closed
270	CMS	Sensorserver is returning the wrong error message when a phone line is busy	closed
271	IRIS Client	The 'blank' and 'send' buttons are intermittently not activating when pressed	closed
272	CMS	Terminal server modem connection problem in sensorserver	closed
273	CMS	Ability to schedule CMS messages with a specifi- able frequency	new
274	CMS	Monitoring and control of portable CMS	new
275	Maps	Add a more detailed street map for D5 IRIS	new
276	CMS	Enable the message duration combobox	new
277	IRIS Reports	Add ability to send single page flashing messages	new
278	IRIS	Add center-to-center (C2C) functionality to IRIS	new
279	CAWS	Can't override CAWS blank message	closed
280	CMS	Add ability to perform periodic 'get message' re- quest for dial-up modems	new
281	CMS	Add a confirmation dialog-box after the user presses the Send button	closed

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
282	Traffic	Add support for microwave traffic detectors for D1	new
283	IRIS	Update to IRIS 3.94, 3.95, 3.96	closed
284	CMS	Message author name not updating correctly in CMS Chooser	closed
285	CMS	Accurate memory of CMS messages in sensorserver	new
286	User Accounts	Edit User List	closed
287	Maps	Add KML output for additional field elements	new
289	CMS	Ability to specify all CMS configuration information via the IRIS client	Reopened
290	CMS	Rapid flashing of DMS page preview	closed
291	CMS	Communication problem with new CMS V60	closed
292	General	D1 IRIS server move	closed
293	CMS	Connection timeout on CMS # V27	closed
294	IRIS	Null handling conformity	closed
295	IRIS Client	Correct font combobox behavior	closed
296	IRIS Client	Correct quick message library combobox behavior	closed
298	IRIS Client	IRIS client froze and can't close	new
299	IRIS	Release 9.0.6	closed
300	User Accounts	Create IRIS account for John Castro	closed
301	IRIS Client	CMS message lines being incorrectly trimmed	closed
302	IRIS	Release 9.0.5	closed
303	IRIS UI	IRIS logout and log back in	new
304	CMS	CMS missing in Chooser catogries	new
305	IRIS	Changing The IP address for IRIS server	new
306	User Accounts	Creat New IRIS user John A Ragusa	Design
307	User Accounts	Create new IRIS user Laura C Williams	Design
308	IRIS Client	The IRIS client needs to fit on a non-huge screen	new
309	CMS	CMS reinitialization message is not clearing cached message	closed
310	CMS	CMS message bitmap corruption issue	new
311	CMS	Users should not be able to specify a page on-time less than the minimum allowed	Development
312	User Accounts	Create new user	Open
313	Maps	Map not visible when the user changes to Cameras tab	new
314	CMS	CMS 'get message' button behavior is incorrect	closed
315	CMS	Vertical CMS message centering is incorrect.	closed
316	IRIS Reports	Update D10 CAWS reports	closed
317	IRIS	Install A complete backup system for IRIS - D10	Development
318	IRIS	Admins can easily lock themselves out when modifying permissions	Open
319	IRIS Client	Need appropriate error message for users without any roles	new
320	IRIS Client	IRIS client freezes if a wrong user name / password was entered	new
321	IRIS Client	Client login failure with EOF message	new
322	CMS	Vertical centering regression	Development

Continued on Next Page...

Table A.1 – Continued

<b>Ticket</b>	<b>Component</b>	<b>Description</b>	<b>Status</b>
323	IRIS	Merge w/ Mn/DOT releases for R9.0.7	new
324	IRIS Client	Make spreadsheet type view of CMS editable and sortable	new
325	CMS	Sensorserver threading issue for modem CMS	closed
326	CCTV	Operator can't use camera PTZ, presets, zoom	closed
327	Maps	Can't see CMS map icons if logged in as 'view-only' user	new
328	CAWS	Incorrect CMS blanking behavior for failed CMS	new
329	CMS	Correct response to detection of power cycle events	Design
330	CCTV	Camera Control seems to work intermittently	new
331	CMS	Client crashes sometimes during multi-selection	new
332	Maps	Pink Screen Map	new
333	Maps	No CMS Icons	closed
334	General	Release 9.0.7 composite ticket	new
335	General	Cumulative communication statistics is not running	new



## Appendix B

# IRIS Installation Verification Procedure

The verification procedure below is used by a developer or system administrator on the production server after installing an IRIS release and before executing verification tests. For information on the larger test procedure, see Section 3.

*These instructions are intended for use by developers or system administrators who understand IRIS and the operating system. If a step does not make complete sense, then stop and find someone that can explain it.*

1. Verify the new release on the development machine:
  - (a) Start the iris client.
  - (b) Verify release number and date have been incremented: Help, About.
  - (c) Verify the CAWS driver is turned off or on (as desired): System Attribute Editor.
  - (d) Verify: travel time messages are turned on or off as desired.
  - (e) Verify: IRIS user permissions are as expected for all users.
  - (f) Verify: each functional area: traffic, CMS, incidents, video.
  - (g) Verify: check the client and server logs, for warning messages for missing system attributes and other warning messages.
2. Verify the new release on the production machine after installation:
  - Verify installation file dates and versions:
    - Verify symbolic links point to the correct directory version for: client, server, sensorserver, trafserver.
    - `ls -lt /usr/share/java/tms/irisserver`
    - `ls -lt /usr/share/java/tms/sensorserver`
    - `ls -lt /usr/share/java/tms/trafserver`
    - `ls -lt /var/www/html/iris-client`

- `ls -lt /var/www/html/iris-client/lib`
- Verify config files have correct IP address:
  - `grep -R '119.17' /usr/share/java/tms/*`
  - `grep -R '119.17' /var/www/html/iris-client`
  - `grep -R '117.19' /etc/iris/*`
  - `grep -R '117.19' /usr/share/java/tms/*`
  - `grep -R '117.19' /var/www/html/*`
- Verify cron jobs are running:
  - Have new cron jobs been created? If so, add them using: `crontab -e`
  - Verify cron jobs are specified: `crontab -l`
  - `ls -lt /var/www/html/signscope.html`
  - `ls -lt /var/www/html/cmsreport.txt`
- Verify the database:
  - Verify the old database has been backed up: `ls -lt /var/lib/pgsql`
  - `grep -R '119.17' /var/lib/pgsql/data/pg_hba.conf`
  - `/sbin/service postgresql status`
- Verify the servers:
  - Verify servers started: `iris, sensorserver, trafserver, drvsrv`
  - Verify the iris server log file looks good: `more /var/log/tms/iris.stderr`
  - Verify the sensorserver log file looks good: `more /var/log/tms/sensorserver.log.0`
  - Verify the trafserver log file looks good: `more /var/log/tms/trafserver.log.0`
- Verify the IRIS client:
  - Start IRIS client via `jnlp` and log in.
  - Verify the release number and date in `help / about`.
  - Verify CMS are active or deployed.
  - Verify Travel time messages are turned on or off as desired.
  - Verify user permissions are reasonable for all users.
  - Verify the IRIS client contains features in the new release.
  - Verify each functional area: traffic, CMS, video cameras, incidents, CMS message logging, SignScope.
  - Verify video is viewable for each camera
  - Verify real-time traffic data is being received: each VDS is not gray.
- Verify system start-up configuration is correct:
  - Restart the IRIS server.
  - Verify each server starts: `trafserver, sensorserver, IRIS, the database`
  - Start an IRIS client and perform a `get-status` request to verify the network is configured correctly.