Title
Cartesius and CTNET Integration and Field Operational Test

Authors
Rindt, Craig R.
McNally, Michael G.

Publication Date
2009

CALIFORNIA PATH PROGRAM
INSTITUTE OF TRANSPORTATION STUDIES
UNIVERSITY OF CALIFORNIA, BERKELEY

Cartesius and CTNET Integration and Field Operational Test

Craig R. Rindt, Michael G. McNally
University of California, Irvine

California PATH Research Report
UCB-ITS-PRR-2009-2

This work was performed as part of the California PATH Program of the University of California, in cooperation with the State of California Business, Transportation, and Housing Agency, Department of Transportation, and the United States Department of Transportation, Federal Highway Administration.

Final Report for Task Order 5324

January 2009

ISSN 1055-1425

Cartesius and CTNET Integration and Field Operational Test

Final Report for PATH TO 5324

Craig R. Rindt and Michael G. M^{c}Nally

Institute of Transportation Studies
University of California, Irvine
Irvine, CA 92697

December 19, 2008

Cartesius and CTNET Integration and Field Operational Test

Final Report for PATH TO 5324

Craig R. Rindt and Michael G. McNally

December 19, 2008

Abstract

This report describes the results of PATH Task Order 5324—the first year of a multi-year project to integrate the Cartesiusincident management system with Caltrans CTNET traffic signal management system. The results of this research are a set of software requirements for reimplementing the Cartesius to interoperate with CTNET. An analysis of the existing Cartesius prototype explains how the need to focus the system on deployment and technical shortcomings of the existing system justifies a reimplementation of the software. From here, we describe the problem to be solved by the new software implementation, including general use cases, the expected users, the systems that Cartesius will interoperate with, and the constraints that will be placed on the system. The problem statement is followed by a detailed discussion of the functional requirements, database requirements, the user interface requirements, and other external interface requirements. The report concludes with a discussion the reimplementation work to be completed under PATH Task Order 6324. This reimplementation will serve the more general purpose of making Cartesius capable of working with existing traffic management subsystems to provide multi-jurisdictional incident mitigation, thus improving its deployability and subsequent value for Caltrans.

Keywords: Cartesius, CTNET, traffic control, multi-jurisdictional incident management

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Purpose of this document

This document describes the results of PATH Task Order 5324, *Cartesius and CTNET: Integration and Field Operational Test*, the first year of a multi-year project to modify the Coordinated Adaptive Real-time Expert System for Incident management in Urban Systems (Cartesius) to work with the Caltrans Traffic Signal Management and Surveillance System (CTNET) and run a limited field operational test on the California ATMS Testbed (Testbed). At the time of this report, the second year of this project is ongoing under PATH Task Order 6324. The original scope of Task Order 5324 was modified during the project to improve synergy between a number of ongoing Caltrans projects. The modifications focused Task Order 5324 on defining the requirements for modifying Cartesius to meet the needs of Caltrans, in general, and to integrate the system with the CTNET system, in particular. PATH Task Order 6324 was re-scoped to implement a new version of Cartesius meeting these requirements.

To satisfy this modified scope, this document describes requirements for the reimplementation of the Cartesius in accordance with the findings of PATH Task Order 5313 (Rindt and McNally, 2007) that detail the challenges of deploying the Cartesius prototype developed as a doctoral thesis under Testbed funding (Logi, 1999).

Included in this document are descriptions of the following aspects of the system:

The functionality What is the software supposed to do?

External interfaces How does the software interact with users, hardware, and other software?

Performance How fast must the system perform its functions?

Attributes What are the security, compatibility, reliability and other non-functional requirements for the system?

Design constraints What standards must the system comply with? What resource limits and operating environments are anticipated? What implementation restrictions are in effect?

1.2 Scope of this document

This document specifies the requirements for the reimplementation of CARTESIUS incident management system to overcome various identified shortcomings of the current CARTESIUS prototype. It is intended to be used as the input to the software design and implementation to be carried out under Task Order 6324.

This documentation was produced by the authors using input from three primary sources.

Existing CARTESIUS prototype The existing CARTESIUS prototype was developed by Dr. Filippo Logi for his doctoral thesis using the G2 expert system shell (Gensym, 1995). Dr. Logi's thesis (1999) and the source code for the prototype have both been major information sources for this document.

Caltrans Caltrans is the primary customer for CARTESIUS and the sole developer of CTNET. Input given by Caltrans personnel during technical meetings has been incorporated into this document.

UCI Researchers UCI researchers, including pre- and post-doctoral researchers and faculty members, are the system engineers and developers of CARTESIUS.

The end goal of this reimplementation is to produce a new version of CARTESIUS that incorporates the system's most important core functionality in a manner that

- is technically feasible given *reasonable* assumptions about the availability of data and algorithms,
- is institutionally feasible in the face of known policy restrictions and the likelihood of jurisdictional participation,
- minimizes the cost and licensing requirements, and

- maximizes the system's potential to be deployed by Caltrans and municipal agencies.

To achieve this goal, UCI researchers' experience with the original prototype was integrated with input from Caltrans personnel regarding Caltrans needs.

The remainder of this document is structured as follows. Chapter 2 provides a broad description of the system including its intended functions and typical users. Chapter 3 defines specific functional requirements for the system. Chapter 4 describes other non-functional requirements. Chapter 5 concludes by offering a number of operational scenarios for the system to illustrate its intended use in more detail.

1.3 Overview

Non-recurrent congestion presents a difficult problem for existing localized automated traffic control systems such as the CTNET. These systems operate with control parameters that are fine-tuned to meet the normal demands of recurrent congestion. Incidents in the system can create disturbances that are beyond the control of such localized systems, rendering them incapable of mitigating the resulting delays. This creates a compelling justification for interfacing the CTNET signal control subsystem with a global traffic management subsystem to develop a general corridor traffic management architecture that can dynamically respond to incidents in the system. Cartesius is such a tool. This research is part of a two year project to first integrate these Cartesius and CTNET projects to produce a functioning traffic management system and then evaluate the integrated system under controlled conditions in a field operational test.

1.4 Business Context

Development of Cartesius is supported by Caltrans, whose mission in this context is to "optimize transportation system throughput and provide dependable travel times." To achieve this mission, Caltrans has funded the Testbed at the University of California, Irvine. The Testbed has produced a range of research products, including the existing Cartesius prototype that is the focus of the current effort. Caltrans seeks to realize the value of its investment in research and has funded this product by way of the California Partners for Advanced Transit and Highways (PATH) program.

The ultimate goal of this effort is to deploy Cartesius as one component of the broader traffic management portfolio available to Caltrans. To that end, the

next chapter provides a general description of CARTESIUS as it relates to Caltrans operations.

Chapter 2

General Description

2.1 Product Functions

CARTESIUS is composed of multiple interacting, real-time decision-support systems (agents) for transportation management center (TMC) operators that are able to perform cooperative reasoning and resolve conflicts, for the analysis of non-recurring congestion and the formulation of suitable integrated control responses. The agents support incident management operations for various transportation subsystems that are divided based upon jurisdictional responsibility. In the ideal case, a particular Caltrans district would have a single agent to manage its freeway and highway network, as would each of the municipalities in a region to manage their respective arterial networks. Each agent interacts with a human operator, is able to receive real-time traffic and control data, and provides the operator with control recommendations in response to the occurrence of incidents. Where technically feasible, CARTESIUS can directly update controller settings in the field at the direction of the operator by interfacing with relevant control subsystems.

The multi-agent approach adopted by CARTESIUS reflects the spatial and administrative organization of traffic management agencies. The agent collective works to provide a coordinated solution that attempts to satisfy all parties, preserves their own levels of authority, and reflects the inherent distribution of the decision-making power.

The original version of CARTESIUS placed nearly an exclusive emphasis on analytical correctness of the system. The needs of the user were a secondary consideration. The emphasis of the current effort is to maintain analytical correctness while serving the needs of the end-user more directly—in this case, TMC operators.

The approach to achieving this goal is to reimplement the existing CARTESIUS

prototype. A reimplementation is necessary for two main reasons. First, there is a general need to refocus its functionality to better meet the needs of the customer. The existing CARTESIUS prototype was designed to serve a research end. Namely, it was developed to demonstrate and evaluate a theoretical approach to managing traffic in multi-jurisdictional environments. The results of this research were positive in that CARTESIUS was shown to be a viable approach to multi-jurisdictional traffic management that could produce net reductions in travel delays during the onset of non-recurrent congestion assuming that particular types of control actions are available to the system (such as information provision to cause traffic diversion and traffic signal retiming to handle increased demands), and that those control actions produce measurable effects (such as particular rates of diversion).

Second, there are numerous technical problems with this prototype in the context of full-scale real-world deployments:

- The Graphical User Interface (GUI) is restrictive, awkward, and fragile.
- The multi-agent aspects are restrictive, hardcoded (2 agents only), and use a proprietary communications protocol that limits interaction with external traffic management systems.
- The knowledge base is difficult to update.
- Many of the algorithms are hardcoded or limited to very restrictive cases.
- The architecture used for connecting to external systems (G2's GSI interface) lacks the flexibility of modern middleware and has been deprecated by the vendor.
- The proprietary system (G2) upon which CARTESIUS is built has costly per seat licensing.
- G2's strengths don't match up well with the algorithmic nature of CARTESIUS.

2.2 User Characteristics

Anticipated users of CARTESIUS are operators in Caltrans or municipal TMCs. These individuals are likely to be technicians or engineers who have some proficiency in the use of analytical software and conventional traffic analysis techniques.

2.3 User Problem Statement

The task of the TMC operator is to manage the impact of events that affect the performance of the transportation system. In this context, performance is defined in terms of low-level operational measures such as throughput, average speeds, delay, or travel time, and not in terms of broader measures such as emissions (though the former may act as a proxy for the latter). The responsibilities of the operator in particular situations will vary. Generally, incident response involves coordinating emergency, operational, and maintenance activities in descending order of importance.

For the purposes of the CARTESIUS software, the relevant role of the TMC operator is to coordinate the operational response by applying various traffic management strategies that include informing drivers through various traffic information systems and adjusting the parameters of the myriad traffic control devices active in the operator's jurisdiction. The actions available to an operator for a particular incident may vary greatly depending on the situation for a number of reasons including the following.

- Technical limitations: the jurisdiction may have little or no ability to dynamically change traffic control settings or to sense conditions and notify travelers in real-time. Further, ignorance of other jurisdictions' actions can lead to local responses that are globally incompatible and therefore further degrade the system's performance.

- Institutional: the jurisdiction may have policies in effect that restrict available responses.

- Circumstantial limitations: there may be no effective response to particular incidents because of the nature of the event and/or the topology of the transportation infrastructure.

Quickly reasoning about this collection of possible actions and constraints to produce a coherent response strategy is a difficult task and it is unreasonable to expect an operator to effectively manage one or more incidents in real-time.

2.4 User Objectives

To make CARTESIUS a useful product it must help TMC operators perform their primary task as outlined above.

- CARTESIUS will initially be used as a supplemental incident management tool providing analytical advice about incident response strategies to operators.
- CARTESIUS could potentially act as the primary incident management interface if it were properly integrated with TMC processes. Such complete integration is *not*, however, a focus of the current effort.
- It must have a GIS-enabled GUI that presents information about the transportation system clearly—similar, perhaps, to Caltrans existing PeMS system. In some contexts, CARTESIUS might be viewed as a real-time PeMS system and a version might be developed that integrates as a plug-in to PeMS—with PeMS acting as the primary Caltrans interface for traffic management.
- CARTESIUS must provide the operator with information about the current state of the system.
- CARTESIUS must provide the operator with information about current problems in the system—including high-level characterizations of these problems displayed in a manner that is both easy to understand and analytically useful. Information must be displayed using commonly understood techniques that are possibly enhanced with new methods.
- CARTESIUS must interact with remote TMCs and other data sources to gather information both about the state of the system and about actions that may be taken by other jurisdictions. This interaction may be with another CARTESIUS agent, or it may be with an independent traffic management software system.
- CARTESIUS must interface with available analytical components (such as a simulation model) to predict potential outcomes of incident responses.
- CARTESIUS should provide the operator with an interface to estimated and predicted system states.
- CARTESIUS should only rely on *currently available* analytical components to improve the characterization of the system (in terms of both capacity and demand).
- CARTESIUS should incorporate knowledge about the capabilities of existing control subsystems into its response plan formulation.

- CARTESIUS should use existing communications and traffic management subsystems to obtain data about the state of the traffic network
- CARTESIUS should provide an interface for analyzing the predictions of the models it uses versus actual behavior in the system.
- The model analysis component should include the ability to visualize deviations between model predictions and actual measurements.
- The model analysis component could provide the ability to calibrate underlying models based upon measurements.
- CARTESIUS could interface with emergency response systems to provide routing guidance to responders.
- CARTESIUS could interface with maintenance management systems to obtain information about planned lane closures and account for them automatically.

2.5 Related System Information

CARTESIUS is not an isolated system. In typical operation it will interact with many systems to gather information about the current and predicted state of the traffic system. Each of these are considered in the sections below.

2.5.1 Traffic Management Systems

CTNET

CTNET is a distributed software system for integrated management of traffic signals that is widely deployed by Caltrans and some municipalities across the state. CTNET allows operators to remotely manage, view, and log real-time traffic signal field data. The system uses a modular client/server architecture in which a CTNET CommServer acts as a proxy server for one or more Traffic responsive field masters (TRFM) (see figure 2.1). The masters, in turn, control a subset of traffic signals, which can be managed by either Caltrans C8, version 4 software (on type 170 controllers) or traffic signal control program (TSCP) version 1.02 software (on type 2070 controllers). The TRFMs support both synchronized traffic responsive and time of day traffic signal coordination schemes.

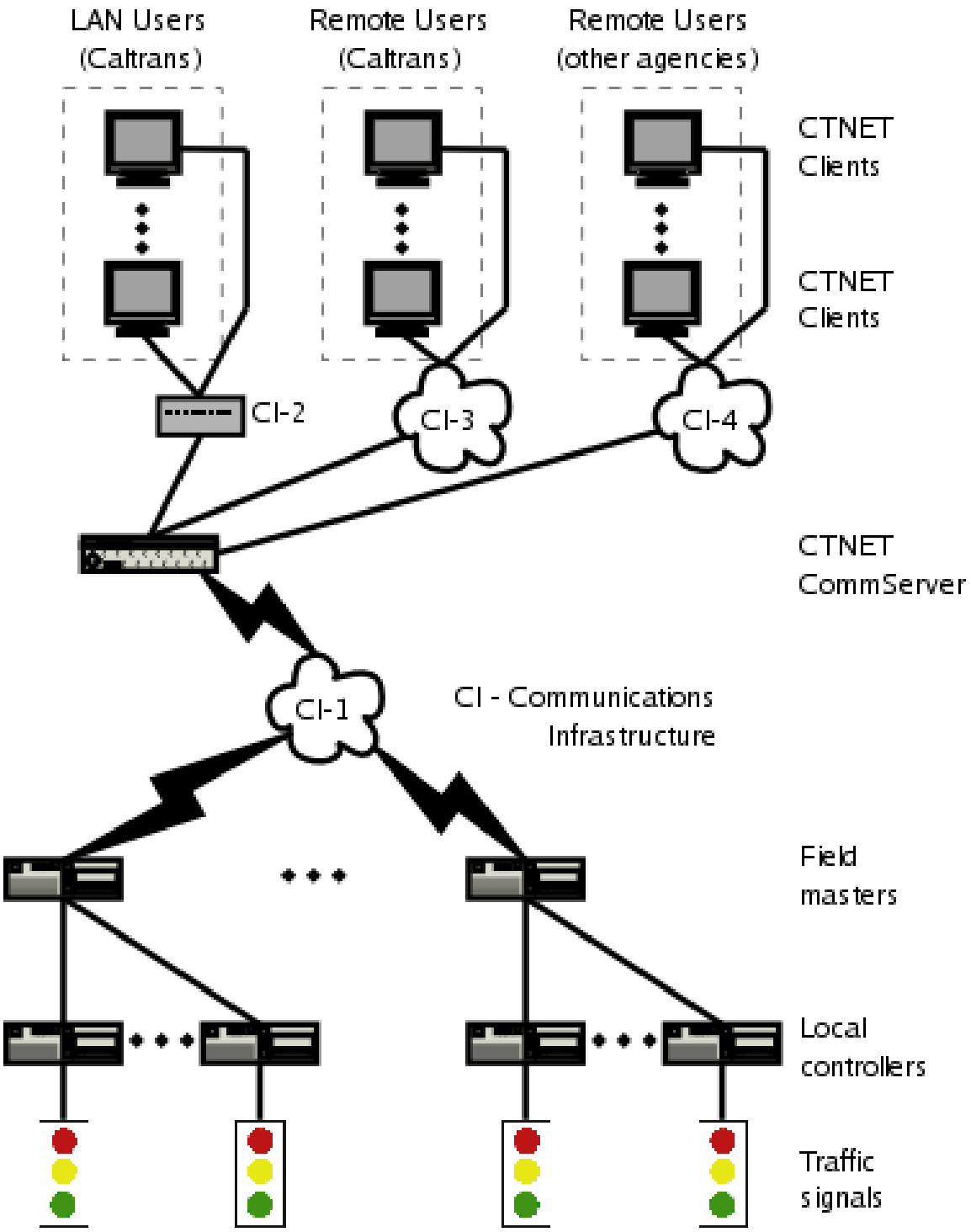

Figure 2.1: The CTNET client/server architecture.

CTNET clients connect to the CommServer to gain access to monitoring and management functions provided by the CommServer. Access privileges are controlled on a per-user basis, allowing the system to limit specific functions to authorized users only. The existing CTNET client provides a user interface which uses TIGER-line data to display geographically accurate maps of the managed traffic signals. An open (non-proprietary) communications protocol is used between CTNET clients and the CTNET server so that third party clients can be developed to work with the system.

By itself, CTNET does not provide automated, globally coordinated incident response. The existing CTNET client software allows TMC operators to manually alter timing plans to implement control responses determined outside the CTNET architecture. These responses may come from any source, but are most likely to be derived on the fly by TMC operators with expert knowledge of the system. Thus, CTNET could benefit from interaction with the global incident management capabilities that are fundamental to CARTESIUS. In this role, CTNET would serve as a traffic management subsystem that supplies CARTESIUS agents with real-time traffic data from arterial system detectors as well as the ability to change signal timing plans as part of a coordinated incident response strategy.

Caltrans District 12 Real-time Data Intertie

The Testbed maintains a real-time data intertie with Caltrans District 12 (D12) that will act as the main source of live data for the development of the software. We anticipate that similar systems will be available for future deployments. The intertie has been configured based on a system architecture that, to the extent possible, renders the UCI ATMS Testbed Intertie to function in a dedicated, independent environment that is functionally isolated from D12's production environment. Specific care has been given to adopting a design that will ensure that the UCI ATMS Testbed Intertie remains operational, with minimal revision, as the D12 ATMS is upgraded or is otherwise changed. Figure 2.2 shows the current configuration.

Under the current Testbed integration with the D12 ATMS, the Testbed Labs at UCI receives the same type of data feed from the D12 Caltrans District 12 Front End Processor (FEP) that the D12 ATMS receives. This data stream is a real-time data feed of 30-second "poll-data" from all of D12's SATMS 170 controllers. The network connection between D12 FEP and the UCI Testbed private network is restricted by the D12 firewall to permit transmission only from the D12 FEP to the UCI Testbed Labs; transmissions to the D12 FEP from the UCI Testbed Labs are strictly prohibited. The real-time data feed from the D12 FEP is received at UCI

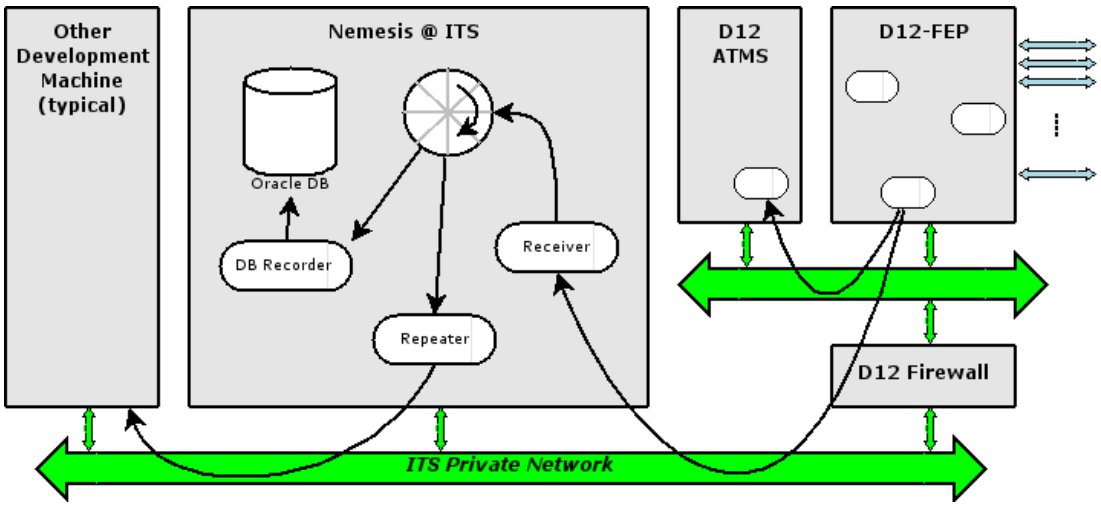

Figure 2.2: The D12 real-time data Intertie

.

Testbed Labs and stored in a database that is available to CARTESIUS.

City of Irvine Real-time Data Intertie

The City of Irvine (COI) uses a Siemens ACTRA Central Traffic Control System and eighteen 2070NL intersection controllers with SE-PAC firmware that was installed along an 18-intersection adaptive control testbed in the COI that supplies real-time traffic data to the UCI ATMS Testbed laboratories utilizing single mode fiber optic communication operating over 4 communication channels. The Eagle ACTRA system provides detection and adaptive control capabilities utilizing existing Ethernet communications.

The communications architecture for the complete system is designed to isolate the Testbed functions from the City of Irvine's traffic system, thus allowing CARTESIUS to interact with the traffic system without the possibility of disrupting normal city operations. This is achieved through custom software, the input acquisition software (IAS), which enables transmission of all detector inputs and signal displays to the UCI Testbed laboratories. This connection provides CARTESIUS with live sensor data from a critical portion of the City of Irvine network that is necessary to support the state monitoring and estimation critical to CARTESIUS functions.

2.5.2 Event Notification

CARTESIUS is an event management system and therefore depends on receiving event notifications. Two existing data sources are currently available for this purpose.

California Highway Patrol Computer-Aided Dispatch System

The California Highway Patrol (CHP) Computer-Aided Dispatch (CAD) system offers detailed information about on-going incidents throughout the state of California. The Testbed has access to the XML media feed from the CAD system which broadcasts the time and location of active incidents as well as detailed free-form logs of the CHP's incident response, edited to remove sensitive information.

D12 TMC Activity Log

The D12 TMC maintains an in-house log that details the activity of its personnel. These entries include coded details of when the TMC begins to manage an incident, when the incident is verified, when personnel arrive on-scene, and when the incident is cleared. Additional free-form details are also available regarding the state of traffic. Log entries cross reference CHP CAD IDs so that the databases can be integrated. CARTESIUS will have direction access to the activity log database for its D12 deployment. We anticipate that similar systems will be available at all Caltrans TMCs.

2.5.3 State Estimation and Prediction Systems

Demand Estimator

The logic of the analytical algorithms used by CARTESIUS to compute incident mitigation actions depends on detailed, path-specific estimates of the demand in the system at the onset of the prevailing incident(s). The original prototype relied on demand estimates developed using a custom implementation of the Dynasmart simulator, which in turn relied upon demand data obtained from a regional planning model. This approach was difficult to maintain and update as necessary. Furthermore, it was limited to modeling demand at particular time of the day and was not sensitive to measurements taken around the time of the incident.

Since the demand estimate is such a fundamental part of the CARTESIUS algorithm, it is essential that the demand estimate be improved. The requirements for this estimate are:

- that it produce estimates of the time-varying demands on specific paths through the network;
- that these estimates are produced on a time-scale consistent with CARTESIUS internal representation of demand—on the order of 5 to 30-minutes; and

- that these estimates are adjusted in quasi-real-time using available sensor data.

Two candidate path-based demand estimators developed by Nie et al. (2003, 2005) and Chootinan et al. (2005) should be considered and a generalized interface to such estimation modules should be developed to allow CARTESIUS to work with future developments in the field.

Capacity Estimator

As with the demand estimates above, CARTESIUS is dependent on estimates of roadway capacity at critical sections in order to develop mitigation strategies and the associated control actions. Real-time estimation of capacity is still an open and difficult problem in the transportation research literature Minderhoud et al. (1997). This is because the concept of capacity is difficult to define consistently—particularly for real-time operations.

A good starting point for this problem is the Performance Measurement System (PeMS) system. Chen (2003) proposed some methods for estimating real-time capacity using PeMS that could be adapted for use by CARTESIUS, or used directly by interfacing CARTESIUS with PeMS itself.

Note, however, that PeMS is for freeways and highways only. Estimation of surface street capacities is another problem that is typically dominated by estimation of intersection capacities. The CARTESIUS prototype contains detailed algorithms to estimate intersection capacities and delays based upon prevailing and forecast conditions. Because of their importance to CARTESIUS reasoning, it makes sense to leave this estimation process as part of the CARTESIUS core. Changes should be made to the architecture, however, to eventually allow these estimates to be modularized and separated from the algorithmic core.

As a fallback, CARTESIUS should implement a simple method for estimating capacity by using the number of lanes and the facility type to establish a baseline capacity. Fine-tuning of the capacity parameters for particular location can be performed during subsequent calibration steps.

2.6 General Constraints

2.6.1 Speed of strategy formulation

The reimplementation of CARTESIUS must meet some basic performance requirements. Generally, the collective of CARTESIUS agents must be able to develop and

implement incident response strategies with loose "real-time" guarantees on the order of 5 minutes. This constraint may later loosen or tighten depending on the full set of response actions implemented in the system. For instance if CARTESIUS were to deploy fine-grained control strategies such as re-timing of particular controllers, the real-time constraints would become more stringent.

2.6.2 Adherence to industry standards

As the National Intelligent Transportation Systems Architecture (NITSA) (Iteris, Inc, 2002) continues to mature, there will be an increasing need to map functionality to the NITSA's functional specification and to support its standards where applicable. Rindt (2005) compared the CARTESIUS prototype to the NITSA specification and found that CARTESIUS partially performed the functions of two different NITSA market packages. By providing the means for developing "integrated control strategies that enable integrated interjurisdictional traffic control" CARTESIUS performs some functions of a NITSA Regional Traffic Control System (ATMS07). The scope of ATMS07 is much broader than CARTESIUS was intended to handle and therefore CARTESIUS does not provide a general solution to ATMS07. Furthermore, CARTESIUS performs some functions that are not part of ATMS07, namely its primary role as an incident management system that better matches the NITSA's Traffic Incident Management System (ATMS08) market package. CARTESIUS shortcoming in this context is that it does not perform some key incident management functions such as emergency response and maintenance management. On the other hand, because it coordinates multi-jurisdictional response, CARTESIUS provides functions that are beyond the ATMS08 specification.

Despite the somewhat faulty mapping between CARTESIUS's functions and the NITSA specification, Rindt (2005) found that CARTESIUS is best viewed as a partial ATMS08 solution that incorporates features of ATMS07. If the system is successfully deployed, additional functions can be added as they are deemed necessary.

Given these conclusions, an effort should be made to use available NITSA communications standards for flows in the architecture. An example of this is the center to center communications standard under development.

In addition to the NITSA architecture, California, in general, and Caltrans, in particular, have internal standards and application deployments that CARTESIUS should use, where possible. For instance, CARTESIUS should interface with Caltrans' CTNET arterial to access data and control subnetworks managed by CTNET.

2.6.3 Limited hardware requirements

Each CARTESIUS agent should require only the computing power available on typical scientific workstation such as a 3-GHz Pentium 4 processor or the equivalent.

2.6.4 Open software licensing

The existing prototype's dependence on expensive commercial software with a custom development dialect was identified as one of the major barriers to continued CARTESIUS development. As such, the new implementation should meet the following requirements.

- CARTESIUS should rely on open source software as much as possible to limit licensing restrictions that might increase the cost of deployments.
- CARTESIUS should be implemented using commonly understood programming languages and established, well supported software libraries.
- CARTESIUS should run on multiple architectures and operating systems.

Chapter 3

Specific Requirements

This chapter details the specific requirements for the CARTESIUS software. These requirements are broken down into four general areas as shown in figure 3.1. The *Functional Requirements* describe the characteristics of the core algorithms and processing carried out by CARTESIUS. Generally, these describe what a given CARTESIUS agent will do once it is set up to receive information from external sources including incident diagnosis and solution formulation. Those external sources are broken down into two categories, each having its own set of requirements. The *User Interface Requirements* describe how CARTESIUS must interact with TMC operators. The *External Interface Requirements* describe how CARTESIUS must interact with components of the transportation system and various models that produce the data necessary for the CARTESIUS core to perform its functions. This section is of particular interest for this project because it details how CARTESIUS must interact with CTNET. Finally, the *Database Requirements* define how CARTESIUS should store the data it uses and the outputs of its actions in order to satisfy the other requirements described.

3.1 Core Functional Requirements

The functional requirements for the system are divided into four primary functions, whose relationships are shown in figure 3.2. The *System Monitoring* function is responsible for tracking the state of the system to identify when disruptions occur that CARTESIUS should attempt to manage. The *Event Analysis* function must produce a problem characterization that analytically describes the disruptions and serves as CARTESIUS's representation of all the problems in the system. When the active problem characterization is modified the *Problem Response* function finds a problem

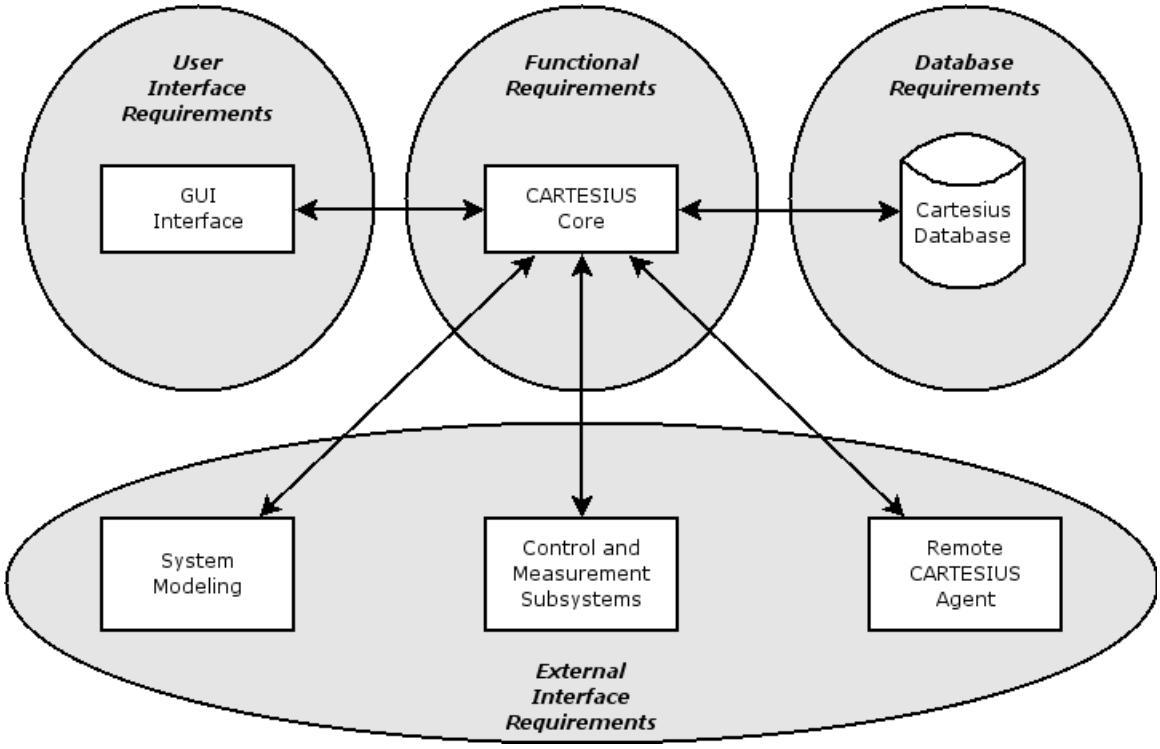

Figure 3.1: The requirement categories used in this specification. The categories are shown as gray boxes.

response using the control actions available in the knowledge base. The *Model Calibration* function should compare model predictions to observed outcomes to identify deviations that should be considered by operators to improve the performance of the system. We consider the requirements for each of these functions in detail below.

3.1.1 System Monitoring

CARTESIUS is a real-time control system and therefore requires access to the state of the system as well as to forecasts of future states of the system. This is divided into two sub-functions: state measurement and state forecasting. The state measurement function provides CARTESIUS with access to state measurements obtained from field devices. The state forecasting function provides CARTESIUS with access to estimates of the future state of the system based upon historical data. The specific requirements for these two monitoring functions are detailed below.

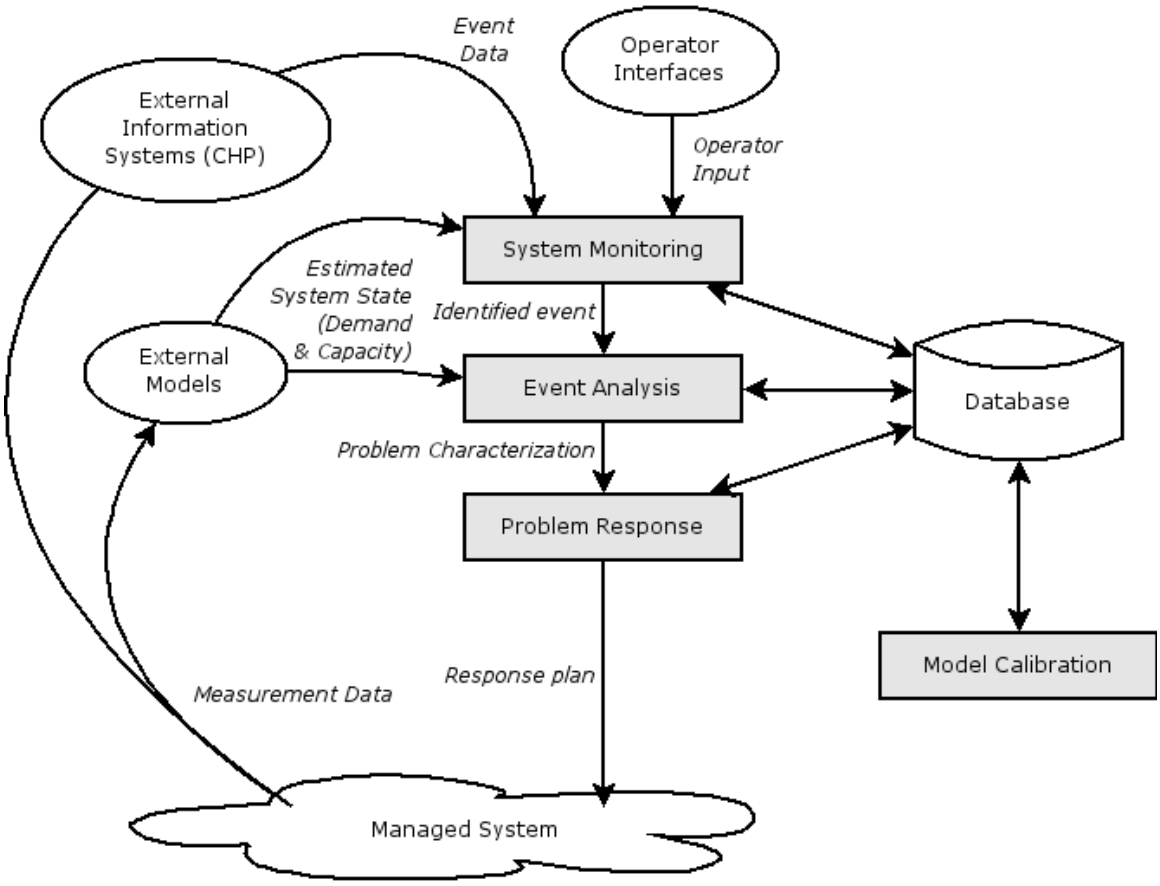

Figure 3.2: The functional requirement categories used in this specification. The categories are shown as gray boxes.

Measure system state

CARTESIUS has no direct responsibility for measuring the system state. It does, however, require access to real-time and historical measurements obtained from the system in order to support modeling the system. To the extent that the supporting services require this information to perform their tasks, such as external state estimation processes (see section 3.1.1), CARTESIUS should have access to the same measurement data, where possible, to provide a consistent view of the data used in strategy formation.

Through this function, CARTESIUS **must have access to speed and flow information for all parts of the network as well as active control settings for all control devices in the system**. Recognizing that this information may not be available in all systems, CARTESIUS **should be able to operate using historical data or planned (assumed) control settings**. FR 1 FR 2

Estimate/Forecast system state

CARTESIUS has no direct responsibility for estimating or forecasting the system state at the most fundamental levels: link-level capacities and path-level demands. It does, however, require access to estimates and forecasts made by external systems. The existence of such systems are a prerequisite for the successful operation of CARTESIUS. Minimally, CARTESIUS **must interface with external systems that can predict the demand on all paths through the managed system as well as the available capacity on all sections of the network**. These functions were part of the core of the original CARTESIUS implementation, but they are better handled by task-specific services that are dedicated to providing those functions (see also section 3.3.2 for the requirements for this external interface). CARTESIUS**'s core functions must use these externally obtained capacity and demand estimates to predict the impact of an event on traffic operations as described in section 3.1.2**.

FR 3

FR 4

Event tracking

CARTESIUS **must implement a function that can identify the onset of an event in the system**. An event is characterized as something that impacts one of the parameters that could affect system performance:

FR 5

1. System capacity: e.g., freeway lane closure, debris in a lane, etc.
2. System demand: a shift from one demand pattern to another, e.g., a special event.

This functionality is effectively an event detection system akin to the Automated Incident Detection (AID) algorithms researched heavily in the 1990's. Here, however, the identification is expected to rely on direct sources of information—e.g., the CHP's CAD system and various Caltrans systems as opposed to AID—to note the onset of an event that could have an impact on system performance.

At the system monitoring level, this event identification may be limited to finding a deviation from "normal operating conditions" defined by *expected capacities* and *expected demands*. We define *expected capacities* as the mean capacity of a facility in the absence of capacity reducing events (e.g., a lane closure). Ideally, capacities will be independent of time-of-day but, in practice, due to the nature of traffic flow, multiple capacities for a facility might exist depending on the evolution of traffic. As such, a peak-period capacity (after breakdown flow) might be less than an off peak capacity.

We define *expected origin-destination (O-D) demands* as the mean flow rate between all pairs of zones in the network representation of the transportation system. This produces a origin-destination (O-D) matrix representing demand. Because flow rates are known to vary by time-of-day, multiple O-D matrices are necessary to represent the dynamics of demand. A secondary representation of *expected* demand is also used by CARTESIUS to describe traffic demand in terms of *path demand*. We defined *expected path demands* as the mean flow rate on all used paths between all pairs of zones in the network representation of the transportation system. The use of paths simplifies the estimation of event impacts as well as the reasoning about alternative responses. Use of path demands requires computation and enumeration of all path flows, which can generally only be achieved via simulation.

A deviation may be considered an event when capacity and/or demand estimates exceed certain thresholds. These may be defined system-wide, or they may apply to specific facilities (in the case of capacities) or specific demand flow components (in the case of O-D and/or path demands).

Generally, an event will either be associated with a known cause or its cause will be unknown. **In any case, the system monitoring function must send any identified event to components that have asked for notification**—including the event analysis module (section 3.1.2). **FR 6**

3.1.2 Event Analysis

The primary task of the event analysis function is to update the problem characterization based upon analysis of prevailing events. The event analysis function in the CARTESIUS core involves two main steps: event characterization and event diagnosis, whose requirements are described below.

Event characterization

This event characterization function in CARTESIUS **must identify the cause of the problem in order for CARTESIUS to determine a course of action**. The reason for this is that CARTESIUS relies upon domain-specific knowledge that is conditioned on the cause of a disruption to the system. **FR 7**

The specifics of this reasoning process are outlined in (Logi, 1999). Generally, the determination of the cause relies heavily on operator input, which includes the operator providing information regarding the severity and expected duration of the disruption. For the purpose of this specification, **the outcome of the event char-** **FR 8**

acterization process must produce one of two results: it will either identify a cause or it will not.

Cause known

The CARTESIUS agent must continue processing to diagnose the event if the characterization process successfully determines a cause and its characteristics (see section 3.1.2). **FR 9**

No event presence

If the characterization process fails to determine a cause and its characteristics, this implies a shortcoming in the characterization process whereby the system cannot determine a course of action for a given valid input. The possible reasons for this are either that the external event notification system is over-reporting possible events or that CARTESIUS's knowledge base is insufficient to handle all inputs. In either case, **a failure to identify an event's cause should be logged along with the associated inputs so that the system can be analyzed and improved to handle a similar case in the future**. **FR 10**

Event diagnosis

The event diagnosis function must generate for a given event characterization the final data object that will be used by the event response functions (see section 3.1.3). The diagnostic function is another core CARTESIUS feature whose logic is described in Logi (1999). The resulting diagnosis contains the following information regarding the event's impact on traffic operations: **FR 11**

- the type of the incident (serves as a key into the knowledge base)
- the change in system capacity caused by the event
- the change in path-level demands caused by the event

In performing this function, CARTESIUS may need access to information from CARTESIUS agents representing neighboring jurisdictions whose operational status and control decisions may impact the local jurisdiction's operations. In particular, the existence of a problem in a neighboring jurisdiction can lead to traffic spillback that crosses jurisdictional boundaries. In this case, CARTESIUS needs to recognize that

the cause of the problem in the first jurisdiction is due to an event in the neighboring jurisdiction. Thus, **the CARTESIUS diagnostic function must consider the possibility of spillback from the neighboring jurisdictions** (this is dependent on the existence of distributed problem solving interfaces for CARTESIUS agents, see section 3.3.1). **FR 12**

Once the diagnosis is completed, CARTESIUS **must pass its event diagnosis as a problem characterization to neighboring jurisdictions so that they can take action if necessary (cooperatively or unilaterally)**. Then, CARTESIUS **must update its active problem characterization reflecting the new event diagnosis**; **FR 13** **FR 14**

3.1.3 Event Response

The event response function is triggered when there is a change to the system's problem characterization, which comes from the event analysis function (section 3.1.2). A change may be the addition or deletion of an event from the active event set, or it may be a change in the characteristics of a particular event in that set—for instance a modification in the expected duration of a lane closure that is input by the operator.

Determine response set

Problem solving in the multi-agent CARTESIUS system is a multi-stage process structured around the partitioning of global problems into sub-problems local to each jurisdiction, which are solved independently. These "partial solutions" are then shared with neighboring jurisdictions as constraints associated with each possible local solution. All agents exhaustively search the combinations of partial solutions to determine globally consistent solutions. This set of globally consistent solutions is then used as the basis for an automated or manual selection process across all jurisdictions. Again, the details of the algorithms involved are described in Logi (1999) and Rindt and McNally (2007). Each of the related sub-functions is discussed in more detail below.

Identify local response set

CARTESIUS **must identify the local response set by consulting jurisdiction-specific knowledge that defines available control actions that can be used to mitigate the operational problems caused by active events in the system**. As such, CARTESIUS **must access a knowledge database of control actions for the jurisdiction**. The requirements for this knowledge database are **FR 15** **FR 16**

defined in section 3.4. Given this information the response algorithm performs a best first search using a heuristic that converts each control action into a set of operational impacts. Again, **the estimated impacts for each control action must be part of the jurisdiction-specific knowledge base**. The result of this process is a set of leaf nodes in the search tree that define a feasible local response set for the identified conditions. **FR 17**

CARTESIUS **must communicate the feasible local response set to any neighboring jurisdictions to incorporate as constraints into their local decision processes**. **This communication must be performed in a way that does not render the local** CARTESIUS **unresponsive to the operator**. **FR 18** **FR 19**

Incorporate global response constraints

Similarly, CARTESIUS **must incorporate any constraints broadcast by remote agents based upon their local response sets**. This functionality requires the existence of a distributed problem solving information sharing interface for CARTESIUS agents (see section 3.3.1). In order to achieve synchronization between agents responding to a given problem, the agent collective may implement a messaging dialect to formally establish conditions under which data will be shared, including time limits for sharing that information. This dialect is a replacement for the protocol in the existing prototype that was built using G2's GSI infrastructure. **The** CARTESIUS **agent must wait for any remote constraints until any synchronization conditions are met, after which it will modify its response search tree as described in Logi (1999)**. **FR 20** **FR 21**

Identify final response set

CARTESIUS **must determine the complete set of feasible global responses by exhaustively searching the modified tree until all constraints are incorporated**. These represent a globally consistent set of possible alternative responses according to all participating jurisdictions. **FR 22**

Select response set

The set of globally feasible responses must be presented to the operator. Additional processing may be necessary for the operator to make an informed decision. **FR 23**

Filter candidates

Each remaining feasible global response is scored using CARTESIUS's domain-specific heuristics to imply an ordering of responses to the operator. CARTESIUS **should support a filtering function that can remove certain response options based upon attributes of the responses (see requirement 3.2.1**. For instance, it may be unacceptable to change the settings of a particular traffic signal on a particular day, or to route traffic onto particular streets. **FR 24**

Evaluate candidates

The operator may need to further evaluate all remaining feasible responses in more detail. CARTESIUS **should be able to send any potential response to an external simulation evaluator to produce detailed measure of effectiveness (MOE) forecasts**. **FR 25**

Choose candidates

Once all filtering and evaluation is complete, the final response must be chosen by CARTESIUS. CARTESIUS **should provide support for the specification of policies that allow the system to automatically choose an action in the absence of operator input**. An example of a default policy would be to not take any action. **FR 26** **FR 27**

Local choice

Though CARTESIUS is designed to be a cooperative system, CARTESIUS **should allow the operator to unilaterally select any globally consistent response via the GUI** (see section 3.2). The interface should order the list based upon any evaluations made and allow the operator to immediately select a response and implement it via connected control subsystems. Locally made choices should be broadcast to the CARTESIUS agents of all neighboring jurisdictions. **FR 28**

Global agreement

The local choice interface presented for the operator to make a local choice must also make possible for the operator to broadcast *suggested* response choices to neighboring jurisdictions. Similarly, **the local choice interface should receive and display any suggested response choice sent** **FR 29** **FR 30**

by neighboring agents. In particular, the interface should show whether all agents have agreed upon the same suggested response strategy, which is dependent on the mechanisms for distributed problem solving information sharing discussed in section 3.3.1. This information supports choice of a globally efficient solution to a given problem state.

Implement response

CARTESIUS **must be able to implement any collection of control actions that are available in a given response, but the actual mechanics of setting that control must be performed by external control subsystems**, such as CTNET. **The** CARTESIUS **core algorithms should be shielded from implementation details as much as possible**. For instance, all control actions should be characterized via common interfaces that return characterizations of the effects of implementing particular control actions, but hide implementation details from the core algorithm.

FR 31

FR 32

3.1.4 Model Calibration

CARTESIUS **could include functionality to simplify model calibration by providing an interface for after-the-fact comparisons of model predictions to observations of the system**. While this is a low priority feature, the possibility of such functionality should be considered during design and implementation.

FR 33

3.2 User interface requirements

This section describes how a single CARTESIUS agent is required to interact with the operator. To overcome a significant design problem with the original prototype **the** CARTESIUS **agent must separate user interface functions from the analytical code**. The user-interface will therefore be provided by a separate module or process that interacts with the core analytical code. The nature of the operator's tasks mean that **the primary user interface must be implemented as a GUI**. The GUI will provide a means for the operator to rapidly assess the state of the system and reason about possible mitigation strategies. The following sections detail the specific requirements for this interface.

IR 1

IR 2

Figure 3.3: The key features of the main user interface.

3.2.1 Heavy GUI client

A heavy client must be provided that mimics the majority original functions of the CARTESIUS prototype. The core features are detailed below. **IR 3**

Map display

The map display must provide high-level contextual information regarding the state of the transportation system. It consists of a layered mapping system that allows the operator to interact with various map features related to the system to obtain information. **IR 4**

The information provided by the various layers will be linked to the analysis display (see requirement 3.2.1) of the GUI that specifies the active dataset to be displayed.

Map interface

The map interface should display detailed map data that provides useful contextual information to the operator. This is GIS data that probably isn't useful for analyzing the system, but aids the operator in identifying the locations of various problems in the transportation system. **IR 5**

The map interface could display low level vector data, or may come in the form of preprocessed images as used by Google maps.

Network interface

The map interface should include a network layer that visually displays the underlying active state of the analytical network model by changing the colors of the links in the network based upon user-selected parameters. A minimal implementation should include speed, some measure of density, and flow rates or volumes. Additionally, the network interface should be capable of showing historical and/or predicted flow rates for components of the network. **The user should be able to click on individual components of the network to obtain additional contextual information**. Supplemental information might include a summary of all data available for that link or node. **IR 6** **IR 7**

Device interface

The map interface must include a device layer that visually displays the known devices in the infrastructure including loop detectors, traffic signals, ramp meters, and changeable message signs. Generally, data about specific devices should be limited to only those devices managed by the agent's jurisdiction. **Clicking individual device icons should provide additional information specific to that device**. For instance, a loop detector might display current traffic state information and also provide an interface to get historical information for the loop.

IR 8

IR 9

Analysis interface

The analysis interface should provide the operator with a view of the combined problem and solution space being considered by the system.

IR 10

This interface must be consistent with the CARTESIUS approach to representing the problem space as a hierarchical tree where nodes represent problem states and edges represent control actions that lead to new estimated problem states.

IR 11

The characteristics of the analysis interface will depend on the implementation of the underlying analytical model(s).

Solution selection

CARTESIUS should allow the user to interact with analysis interface to view all candidate solutions developed by the system, possibly the ability to filter out candidate solutions based upon particular criteria (see requirement 3.2.1). **The analysis interface should also allow the user to activate a particular solution for exploration**. This activated solution will determine the contextual content shown in the map display. The most likely approach to this interface will be a clickable tree graph that allows the user to select a given node (representing a problem) and to view the estimated network conditions associated with that problem state. The state estimate will come from any number of sources that should minimally include:

IR 12

IR 13

- Direct measurement of the current state (this is only applicable to the root node that represents the current problem state).

- Internal CARTESIUS state estimation routines relying on historical demand patterns and macroscopic network performance models.
- External state estimation modules, such a traffic simulation (e.g., Paramics), that can provide time varying estimates given an initial snapshot and anticipated control actions.

This functionality will make CARTESIUS a unique TMC application that provides a single interface to all analysis and estimation models available at the TMC.

Solution display

Selection of a problem node in the analysis pane should affect the map display. The effects should be reflected in the network display on the map by changing the underlying network and device display datasets used by the map display. Additionally, **the interface must provide a map-based representation of the problem characterization associated with the solution node**. IR 14 IR 15

The display of the problem characterization should be capable of representing critical sections and the demand/supply imbalance estimated by CARTESIUS. IR 16

3.2.2 Thin web client

CARTESIUS may be integrated with existing traffic management and analysis systems. Such systems include PeMS and other web-based technologies. To support this possibility, **A basic** CARTESIUS **web client could be developed that shows the status of a jurisdiction's** CARTESIUS **agent.** While this is a feature considered for later development, it is a likely evolutionary path for CARTESIUS. The requirements for a thin web client are similar to those for the heavy client. This possible evolution should be considered during the design and implementation of CARTESIUS in Task Order 6324, but not as a binding requirement. IR 17

3.3 External interface requirements

3.3.1 Distributed Problem Solving Interfaces

CARTESIUS partitions traffic management on a jurisdictional basis to isolate the management of problems to a single jurisdiction in the system as long the effects of those

problems are contained in one jurisdiction. The existence of spillback effects across jurisdictional boundaries, and the fact that some control strategies (such as diversion) can place operational burdens on portions of the network outside the jurisdiction of the diverting agency, require a means for sharing information between jurisdictions regarding such effects. It is here that CARTESIUS distributed problem solving (DPS) philosophy provides benefits by defining the specific types of information that need to be shared across jurisdictions in order to ensure that the collective of local responses to a given problem state produce a global solution that is acceptable to all jurisdictions.

The CARTESIUS adaptation of Functionally Accurate/Cooperative (FA/C) methods (Lesser and Corkill, 1981) to this problem defines two distinct stages during problem solving when distributed agents should exchange information: problem diagnosis and problem solution. The CARTESIUS approach also defines what type of high-level information should be exchanged at those points. In each case, the general approach is to locally perform detailed analysis, then to create and post an abstraction of that analysis that can be used by other agents that then leverage the abstraction to refine their analysis. The process can be iterative and each instance must be carefully analyzed to ensure tractability and the avoidance of race conditions in the distributed system.

Diagnostic Information Sharing

The characterization of traffic problems is complicated in a distributed system because of the potential relationship between problems as discussed above. Since such propagation can cross jurisdictional boundaries, the local problem characterization process described above must be augmented to handle information regarding the possible relationship between problems across jurisdictional boundaries.

CARTESIUS handles this problem by identifying all spillback effects and posting their characteristics to other agents for them to identify possible links between them. This information sharing allows agents whose jurisdictions are involved to compute *derives from* relationships between problems that cross these boundaries and to share those determinations with the agent collective.

An external interface for diagnostic information sharing must be provided. This interface must allow for diagnostic information sharing that allows for non-blocking information propagation between agents regarding conditions that could potentially produce spillback effects across jurisdictions. **ER 1**

Problem Solving Information Sharing

The nature of partially decoupled DPS for complex, large-scale systems invariably leads to uncertainty and imperfect knowledge that makes the use of heuristic algorithms the only tractable approach. Any exact or heuristic optimization algorithm requires specification of an objective that can be evaluated in reasonable time (as dictated by the constraints of the algorithm and available computing power). Furthermore, the algorithm needs a means for determining how to reach the objective (i.e., a search direction).

CARTESIUS uses a problem solving heuristic that seeks to both avoid the spread of congestion that may lead to oversaturation and secondarily, to achieve a balanced ratio between network capacity and traffic demand. The CARTESIUS traffic management agent uses problem solving algorithm that incrementally searches a space of problem mitigation strategies, each of which can be *translated* by a corresponding control action or set of control actions. Thus, a strategy to *reduce critical section demand through diversion* might be realized by setting a collection of changable message sign (CMS) to reroute traffic around the problem. The expansion of the solution search tree through strategy and control actions generates new nodes representing an estimate of the state of the system if the control actions on the path from the root node to the target node are applied.

The selection of given strategy or the specific control actions that translate that strategy may require the satisfaction of particular conditions that define dynamic constraints that must be met for those strategies or actions to achieve their anticipated effects. These constraints may only apply to the local jurisdiction, but often, as in the case of diversion strategies requiring sufficient network capacity, they may propagate to remote jurisdictions.

CARTESIUS propagates such constraints as conditions that the remote agent must meet. The remote agent translates these conditions into strategies that have the goal of meeting the condition. These strategies are used to expand the search tree by branching from existing nodes to generate new portions of the search tree that will satisfy the conditions broadcast by other agents.

The process continues until no more conditional strategies are being broadcast by any agents. This indicates that the global search has been exhausted and the existing candidate solutions in the tree represent global collective of candidate solutions under consideration, including the extent to which each solution meets any broadcast constraints. The task of selecting a single global solution from among the set of candidates is one of removing solutions that are inconsistent with local or remote

constraints. Since each agent has the same list of the available feasible solutions, it is now possible for the agent collective to jointly select a single global solution consisting of a combination of locally acceptable control actions.

An external interface consistent with CARTESIUS core logic for DPS must be provided. The DPS information sharing interface must offer non-blocking operation. Furthermore, **The DPS information sharing interface must support a protocol for information sharing that allows the agent collective to determine whether or not all agents have exhausted their feasible search space, and therefore will not be propagating further constraints across jurisdictions**.

ER 2

ER 3

ER 4

3.3.2 External Data Interfaces

As discussed in section 3.1, The CARTESIUS core interface with external systems to obtain various data about the state of the system over time. For CARTESIUS to effectively perform its functions, these data must be available on demand or be streamed to CARTESIUS in near-real time.

The specific data required by CARTESIUS from these subsystems is described by the core functional requirements in section 3.1. They include data from:

- Event/incident notification systems
- Measurement subsystems
  - Current flow, occupancy, and possibly speeds
- Control subsystems
  - Traffic signals
  - Ramp meters
  - CMSs

In addition to these direct interactions with components of the system, CARTESIUS relies on support from external models of the system as well as having access to historical information regarding the state of the system.

The requirements for each of these external interface classes are described in the following subsections.

Event/incident notification systems

CARTESIUS core behavior is triggered by the identification of events in the managed traffic network. Therefore, CARTESIUS **must interface with available event notification systems to obtain information regarding new and evolving events in the system**. For this project, CARTESIUS **must interface with the CHP CAD system** (see section 2.5.2). CARTESIUS can also make use of jurisdiction-specific event notification systems. For instance, **The D12 agent should also interface with the D12 TMC activity log (see section 2.5.2) to get more detailed information regarding incidents in that jurisdiction**. **ER 5** **ER 6** **ER 7**

Control and Measurement Subsystems

In modern traffic control infrastructure, control and measurement functions are typically provided by integrated systems that are specific to the jurisdiction. Generally, CARTESIUS **must interface with all traffic control subsystems used by a particular jurisdiction to measure and control traffic**. To the extent possible, **the CARTESIUS control and measurement interfaces must import external information into data objects used to implement the core CARTESIUS logic**. Similarly, **the CARTESIUS control interfaces must be able to translate control settings from data objects used to implement the core CARTESIUS logic**. **ER 8** **ER 9** **ER 10**

For this project, two agents will be built reflecting the Caltrans D12 jurisdiction and the City of Irvine jurisdiction.

D12 agent external interfaces

The CARTESIUS D12 agent must interface with all measurement and control functions provided by the Caltrans D12 Real-time data intertie (see section 2.5.1). **ER 11**

City of Irvine agent external interfaces

The CARTESIUS Caltrans City of Irvine agent must interface with all measurement and control functions provided by the CTNET traffic signal management system (see section 2.5.1). Note that the CTNET system for the City of Irvine will operate by interfacing with the City of Irvine Real-time Data Intertie (section 2.5.1). **ER 12**

System Modeling

Fundamentally, a CARTESIUS incident response is based upon finding a collection of response strategies with associated control actions that will balance prevailing demands with the available capacities. This requires that CARTESIUS has estimates of current (and future) demand and capacities in the system. The following subsections detail the external interface requirements for such estimates, and by proxy, the requirements of the external modules that provide those estimates.

Demand Estimates and Predictions

CARTESIUS **must obtain time-varying estimates of the traffic demand on all paths through the managed network**. These estimates should be made on time scales that are consistent with CARTESIUS core analytical models (e.g., 15-minute periods). The path-based demand estimates must refer to (or be convertable to) links in CARTESIUS's representation of the traffic network. **The demand should be represented as the total number of vehicle-trips using each path per time period**. **ER 13** **ER 14**

Demand estimates should be provided to CARTESIUS **whenever new estimates are available**. **ER 15**

CARTESIUS **should also be able to request that a new demand calculation be performed**. **ER 16**

Estimated Freeway Capacities

CARTESIUS must estimate the capacity of freeway sections, including those that have been affected by an incident. These can be provided by the operator as part of the incident reporting process. However, CARTESIUS **might obtain estimates of roadway capacity from external models or databases that have performed such estimates**. For instance, the PeMS database provides capacity analysis that may be of use to CARTESIUS. **ER 17**

Estimated Intersection Capacities

CARTESIUS must also estimate the capacity of intersection approaches in the managed network. These estimates are complicated by the fact that approach capacities are dependent on the flow rates and control settings of all approaches at an intersection. Since CARTESIUS may consider signal timings as a possible control action, CARTESIUS must be capable of estimating intersection capacities internally. In some

cases, however, CARTESIUS may rely on external control systems to adapt to variations in demand. In such cases, CARTESIUS **may need to communicate with external control or model systems in order to estimate available capacities**. ER 18

Historical Data

While not critical for CARTESIUS functions, CARTESIUS **should have access to the historical data from the system for all inputs required by** CARTESIUS **in real-time operation**. These data should include historical sensor, demand, and event data. In most cases, these historical data will be maintained by external systems. Where such data are *not* maintained by external systems, CARTESIUS should archive the data it uses in its own data store (see section 3.4 for related logical database requirements). ER 19

3.4 Logical database requirements

CARTESIUS **must use a database to store all possible data related to its operation**. This includes all configuration, user details, domain knowledge, and the results of any analytical processing. DR 1

All information stored must be kept until it is explicitly removed from the database. **The data must also remain consistent throughout its lifetime such that removal of particular entities from the database, such as a user, will also remove any static and dynamic information that is exclusively associated with that entity**. Such a removal may be linked to data that is not exclusive to a particular entity—such as a user being associated with particular actions taken by CARTESIUS. **Deletions of linked data should not be allowed and the system should report an error condition rather than carry out the operation so that consistency can be maintained**. DR 2 DR 3 DR 4

The details of the data model will be finalized during the design and implementation stages. However, the following are some general requirements for particular data objects that will be fundamental to the CARTESIUS implementation.

CARTESIUS **must maintain an internal data model for representing traffic networks**. To ensure that the resolution of this network model is consistent with available analytical tools, CARTESIUS **network data model should be compatible with analytical representations of current microsimulation models**. This will simplify network transformations from and to external data sources. DR 5 DR 6

CARTESIUS **must maintain an internal data model for representing traffic** DR 7

signal timing plans on the 2070 controllers. Similarly, CARTESIUS **must maintain an internal data model for representing ramp meter control settings consistent with those used in D12 operations**. This will allow CARTESIUS to implement control actions on the controllers used in the Testbed. **DR 8**

Chapter 4

Other Non-functional Attributes

Specifies any other particular non-functional attributes required by the system. Examples are provided below.

4.1 Security

Because of the sensitive nature of traffic management for both the general safety of the population and for more strategic questions of homeland security, CARTESIUS must include robust security measures at all levels. This minimally include:

- User authentication for any access to live system data.
- Layered security using a role-based system that limits access to sensitive data or control functions to sub classes of users.
- Full logging of all actions taken to support security auditing.

4.2 Binary Compatibility

There are no binary compatibility requirements for the CARTESIUS reimplementation.

4.3 Reliability

The system should be at least as reliable in all functions as the existing CARTESIUS prototype. This includes replicating the functional completeness of the original prototype as outlined in Logi and Ritchie (1997) and Logi et al. (2001). The user interfaces of the system must be sufficiently stable to allow for production use in

limited field operational test settings. This level of reliability should be the first step toward production-quality reliability of the system, the establishment of which will require later testing in production settings.

Chapter 5

Conclusion and Future Work

PATH Task Order 5324 is the first year of a multi-year effort to integrate the Cartesius incident management system with Caltrans CTNET traffic signal management system to produce a functioning traffic management system for arterial signals that can interoperate with the traffic control systems of adjacent jurisdictions to implement coordinated response strategies to incidents. Research to date has extensively studied both Cartesius and CTNET, and produced this requirements document for reimplementing Cartesius in order to alleviate the most significant barriers to deploying the original Cartesius implementation in conjunction with CTNET.

The re-implementation of Cartesius and related CTNET integration is being carried out under Task Order 6324. Based on feedback from the Caltrans customers for the product of this research, the work plan for Task Order 6324 has been revised from the originally planned full field operational test that served as the basis for both Task Orders 5324 and 6324. This new work plan includes development and testing of the new Cartesius software with a specific focus producing a deployable integration with CTNET. The following sections describe the plans for linking using CTNET as a control action provider for the Cartesius response algorithm and summarize the overall integration and evaluation plan for the remaining research.

5.1 Cartesius Response Formation and Strategy Translation

Cartesius response formation is the product of a heuristic search through the problem space defined by estimated traffic conditions and estimated incident impacts in terms of demand and capacity disruptions. The search is driven by an agency-specific

set of response strategies and related control actions that can be used to implement those strategies. These agency-specific configurations comprise a knowledge base that constrains the possible solution space to solutions that are consistent with local traffic control policies. Any evaluation of the CARTESIUS/CTNET system will be governed by this knowledge base and the particular set of possible strategies and control actions available in the knowledge base. For the remaining research under Task Order 6324, two types of control actions will be added to the knowledge base for use in the I-405 Corridor. The first will use predetermined timing patterns accessible from the controllers using CTNET to provide coordinated signalization strategies to the search algorithm. This will permit CARTESIUS to interoperate effectively with typical installations currently in use. We will also consider adding a control action to the knowledge base that uses the adaptive control system being developed in PATH Task Order 6323, *Optimal Control for Corridor Networks: A Mathematical Logic-Based Modeling and Solution*. This control subsystem solves the corridor ramp-metering and traffic signal control problem using a multi-objective formulation that produces solutions for optimal control that specify the set of non-dominated corridor control options. CARTESIUS can explore this set using its search algorithm to find the solution that is acceptable to all participating agencies.

5.2 Path to Deployment

The integration of CARTESIUS and CTNET offers a significant step toward deploying true corridor control of non-recurrent congestion. The use of existing timing patterns provided by CTNET improves the deployability of CARTESIUS by using control actions that have already been vetted by traffic engineers rather than trying to dynamically determine new strategies. Consequently, a completion of the final year of this project will move CARTESIUS significantly closer to potential real-world deployment. Toward this ultimate end, the CARTESIUS/CTNET evaluation is being conducted on the I-405 corridor network in the City of Irvine. This location has been chosen because we have at least limited authority to conduct tests involving closed-loop control. On the arterial, we have installed a system of Type 2070 controllers at all signalized intersections that operate independently from the local COI system. Research already completed as part of this project has connected these controllers to CTNET; a secondary system based on state-of-the-art Siemens ACTRA Central Traffic Control System with custom-designed Input Acquisition Software is in place as a backup, should the CTNET configuration prove problematic. Software

has been developed, and laboratory tested, that permits real-time adaptive control of Caltrans D12 ramp meters in the study area (a feature not currently possible under Caltrans D12 ATMS). We have established real-time communication with these control devices and also receive real-time raw data streams from loop detectors within the study area. We have memoranda of understanding with both the COI and Caltrans D12 that permit the research team to conduct closed-loop control experiments in the study area.

As part of Task Order 6324, we will first evaluate this joint deployment of Cartesius and CTNET using our Paramics simulation of the corridor using custom plugins already written as part of this research to allow the simulation to interoperate with CTNET, and ultimately Cartesius—this configuration is shown in 5.1. Here, links to the simulated control and sensor systems are provided by Testbed plugins. Freeway data coming from the Paramics ATMS plugin flows into the Testbed database (a mock version in this simulated implementation) through direct communication with the plugins. This configuration mimics the architecture of the Real-time Data Intertie described in section 2.5.1. Similarly, traffic signal data flows to and from the AB3418e Paramics plugin to the CTNET CommServer. A custom CTNET client connects to the CommServer over the CTNET Comm Protocol and pushes arterial traffic signal data into the database. Cartesius uses a traffic measurement data access object (DAO) to obtain real-time sensor data for monitoring the system. A demand model also uses this DAO to get measurements to update the system's estimate of prevailing demands, which Cartesius accesses via the Demand DAO.

When events dictate a Cartesius response, the agent collective selects a global response strategy from available control actions. In this simulated mode, these control actions are pushed to the field controllers using the Signal Control, Ramp, and CMS DAOs and traffic monitoring continues as described above.

The system will also be connected in a read-only manner to real-world data feeds to evaluate its recommendations as compared with actual operations. This configuration is shown in figure 5.2. The only differences in real-world operation are in the lower half of the figure. Here, freeway data is sent to the Testbed from the D12 FEP and pushed into the Testbed database by the common ITS data processor as described in section 2.5.1. Arterial sensor data has two paths into the system. The existing path described in section 2.5.1, uses the custom input acquisition software (IAS) software to obtain arterial sensor data, which is read by the ITS data processor and sent to the Testbed database. This data is also bridged to the CTNET CommServer so that CTNET can access IAS systems. Where field controllers deploy the the

Caltrans TSCP program, these controllers can communicate directly with the CTNET CommServer. Beyond these low-level communications, all processing in the real-world mode of the system continues as described above. In the final analysis of the real-world operation, we will assess the degree to which constraining the control actions available to CARTESIUS (e.g, by using only pre-vetted signal timing patterns stored in CTNET) impacts the quality of the system's response to incidents by comparing recommended actions with those actions taken by the TMC.

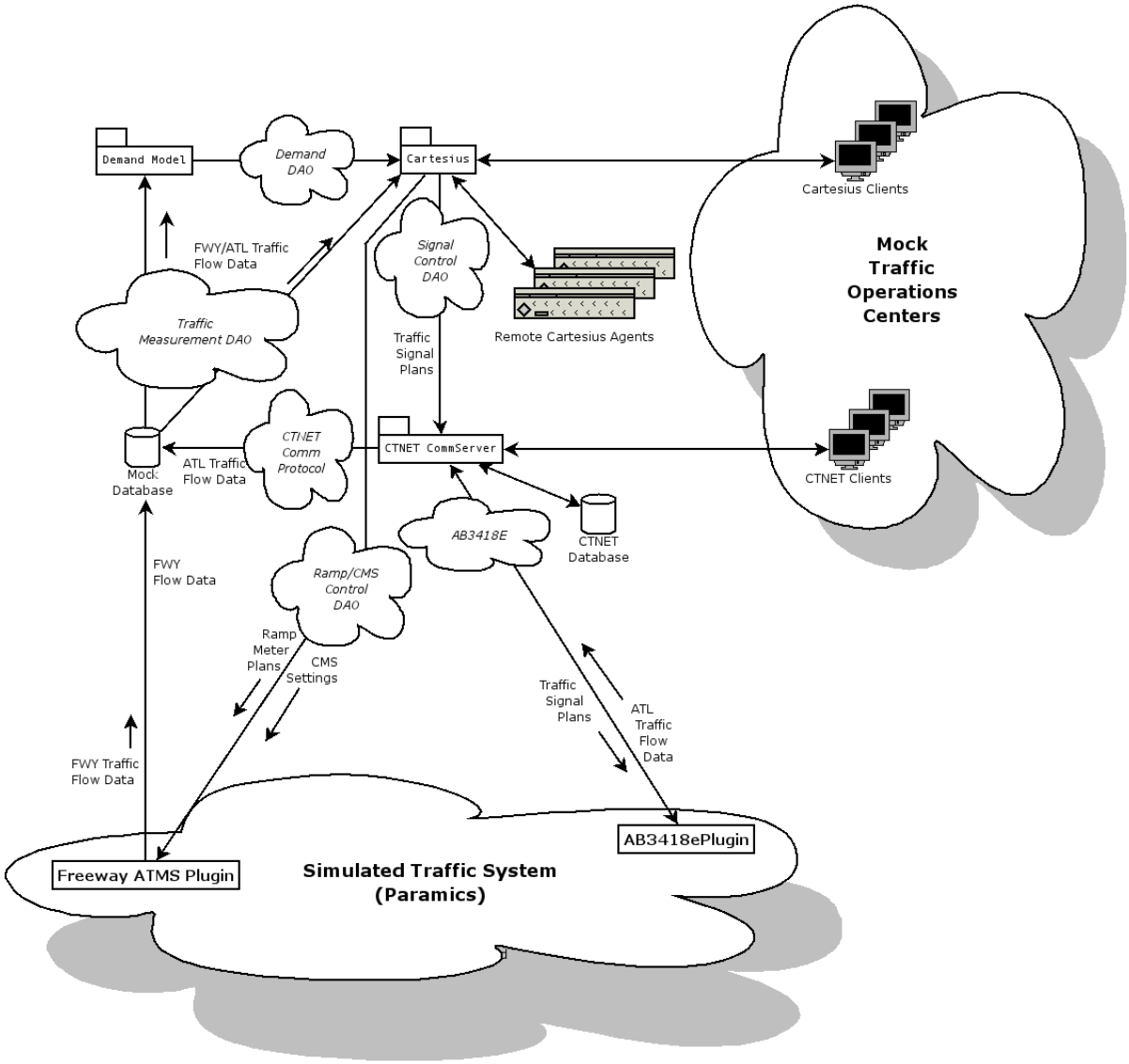

Figure 5.1: Simulation data flows

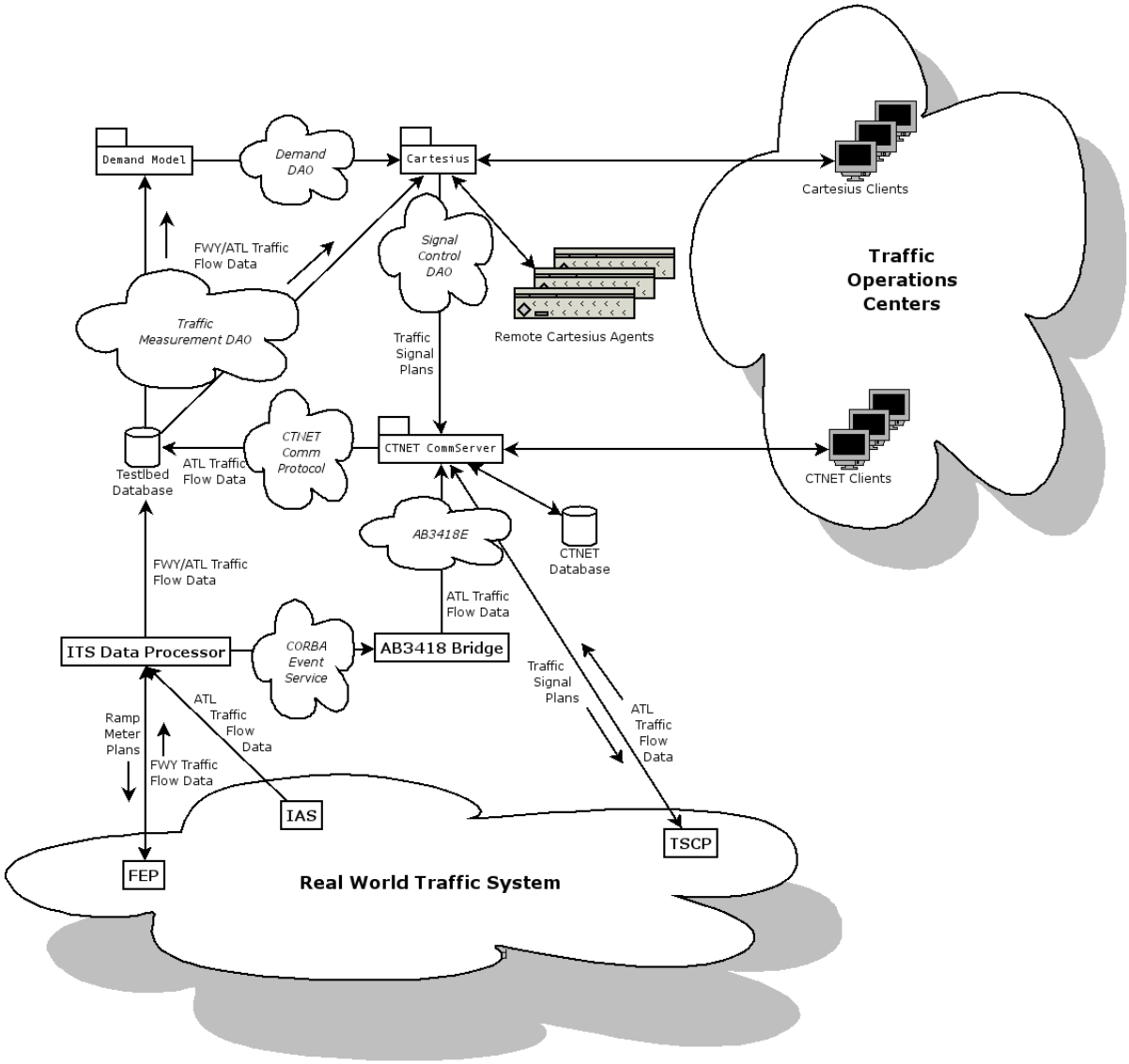

Figure 5.2: Real-world data flows

References

Chen, C. (2003). Freeway Performance Measurement System (PeMS). Technical Report UCB-ITS-PRR-2003-22, California PATH.

Chootinan, P., Chen, A., and Recker, W. W. (2005). Improved path flow estimator for estimating origin-destination trip tables. *Transportation Research Record*, 1923.

Gensym (1995). *G2: Reference Manual. Version 4.0.* Gensym Corporation, Cambridge, MA.

Iteris, Inc (2002). National ITS architecture. Technical report, US Department of Transportation. URL `http://itsarch.iteris.com/itsarch/`.

Lesser, V. R. and Corkill, D. D. (1981). Functionally Accurate, Cooperative distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):81–96.

Logi, F. (1999). *CARTESIUS: A Cooperative Approach to Real-time Decision Support for Multijurisdictional Traffic Congestion Management.* PhD thesis, University of California, Irvine.

Logi, F., Rindt, C. R., McNally, M. G., and Ritchie, S. G. (2001). Advancd transportation management system test bed for evaluation of interjurisdictional traffic management strategies. *Transportation Research Record*, 1748:125–131.

Logi, F. and Ritchie, S. G. (1997). Verification, validation and evaluation of TCM, a knowledge-based system for traffic congestion management. Technical report, Institute of Transportation Studies, University of Califorina, Irvine.

Minderhoud, M. M., Botma, H., and Bovy, P. H. L. (1997). Assessment of roadway capacity estimation methods. *Transportation Research Record*, 1572.

Nie, Y., Zhang, H. M., and Recker, W. W. (2003). Equilibrium-based o-d estimation using a constrained generalized least squares path flow estimator. Technical Report UCD-ITS-Zhang-2003-4, Institute of Transportation Studies, UC Davis.

Nie, Y., Zhang, H. M., and Recker, W. W. (2005). Inferring origin-destination trip matrices with a decoupled gls path flow estimator. *Transportation Research*, 39B:497–518.

Rindt, C. R. (2005). A preliminary comparison of cartesius and CTNET. Technical report, University of California, Irvine. URL `http://parsons.its.uci.edu/projects/cartesius/ctnet-integration/reports%/cartesius-ctnet-integration.pdf`.

Rindt, C. R. and McNally, M. G. (2007). Field deployment and operational test of an agent-based, multi-jurisdictional traffic management system. Technical Report UCB-ITS-PWP-2007-1, University of California Irvine.